Hybrid CP-SAT Solver
for Temporal Planning and Scheduling

Arthur Bit-Monnot

MAFTEC 2024

Aries

New experimental solver[1]

- for experimenting with hybrid SAT/CP solving
- targeting Temporal Planning

First milestone: a CP-SAT solver for Disjunctive Scheduling

- Core subproblem of temporal planning
- Not a strong suit of existing CP-SAT solvers

[1] MIT licensed, in Rust: `https://github.com/plaans/aries`

Case Study: Disjunctive Scheduling

Given a set of **tasks** subject to **precedence** and **no-overlap** constraints, find a schedule of **minimum duration**.

Most famous: **JobShop** Scheduling Problem

JobShop: CSP model

Constants:

- $d_i \in \mathbb{N}$: duration of the i^{th} task.

(Decision) Variables:

- $s_i \in \mathbb{N}$: start time of the i^{th} task.

Constraints:

- precedence: $s_i + d_i \leq s_j$ $\quad \forall i, j \text{ s.t. } job(i) = job(j) \wedge i < j$
- no-overlap: $s_i + d_i \leq s_j \ \vee \ s_j + d_j \leq s_i$ $\quad \forall i, j \text{ s.t. } machine(i) = machine(j)$

Objective (makespan)

- Minimize $\max_i s_i + d_i$

Aries: CP Solver Fundamentals

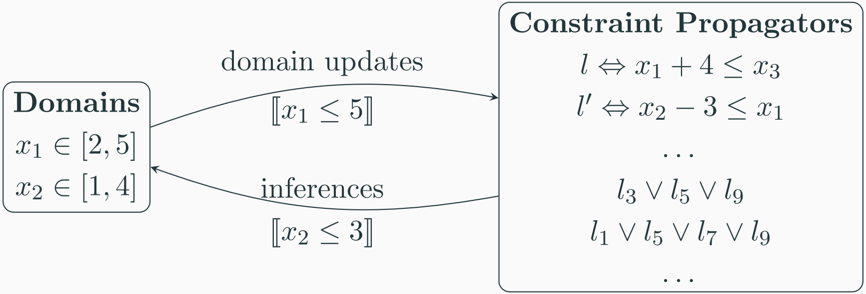
Domains

$x_1 \in [2, 5]$

$x_2 \in [1, 4]$

- **Variables**: (bounded) integers
- **Domains**: Lower & Upper Bounds
- **Search events**: Bound changes

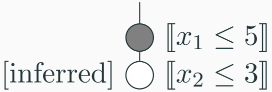
Aries: CP Solver Fundamentals



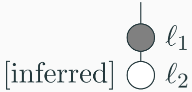
CP-SAT Search: Conflict Directed Clause Learning (CDCL)

$[\![x_1 \leq 5]\!]$

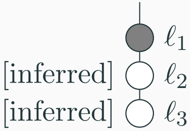
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



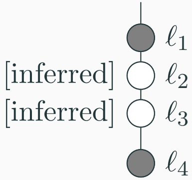
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



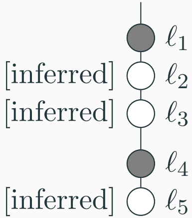
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



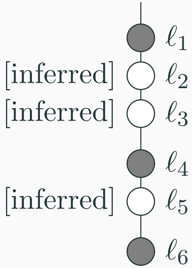
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



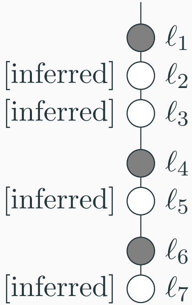
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



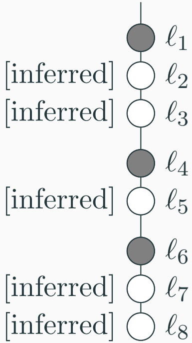
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



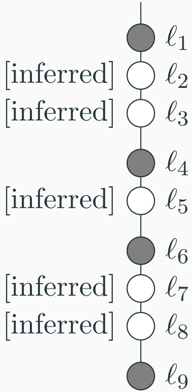
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



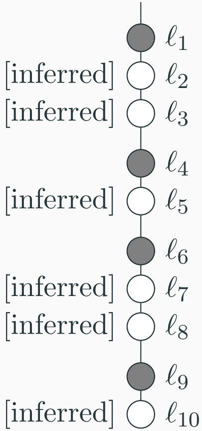
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



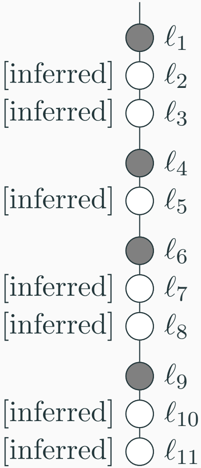
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



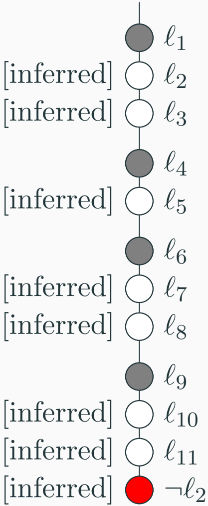
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



CP-SAT Search: Conflict Directed Clause Learning (CDCL)



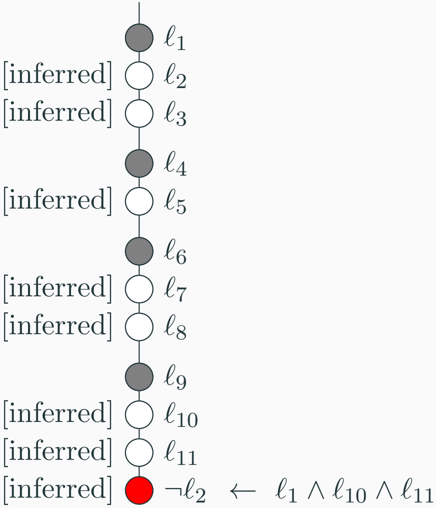
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



Conflict:

$$\ell_2 \land \neg\ell_2 \to \bot$$

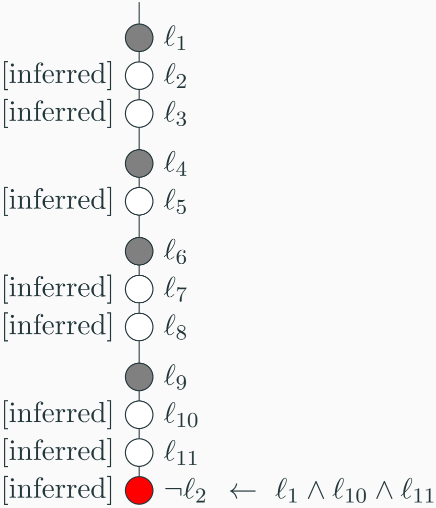
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



Conflict:

$$\ell_2 \wedge \neg\ell_2 \rightarrow \bot$$

CP-SAT Search: Conflict Directed Clause Learning (CDCL)

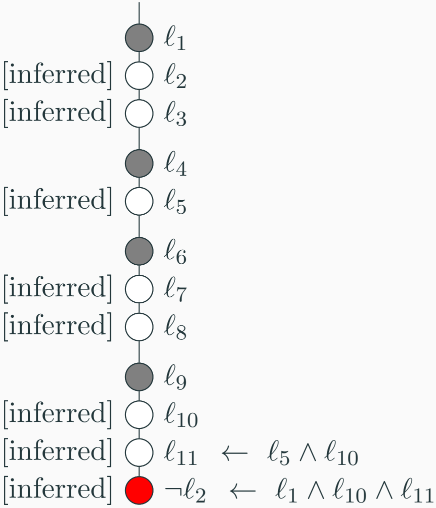


Conflict:

$$\ell_2 \wedge \neg \ell_2 \rightarrow \bot$$

$$\ell_2 \wedge (\ell_1 \wedge \ell_{10} \wedge \ell_{11}) \rightarrow \bot \qquad \text{(resolved } \neg \ell_2\text{)}$$

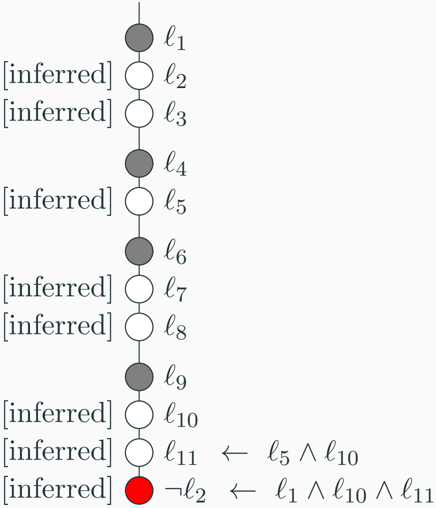
CP-SAT Search: Conflict Directed Clause Learning (CDCL)



Conflict:

$$\ell_2 \land \neg\ell_2 \rightarrow \bot$$

$$\ell_2 \land (\ell_1 \land \ell_{10} \land \ell_{11}) \rightarrow \bot \qquad \text{(resolved } \neg\ell_2\text{)}$$



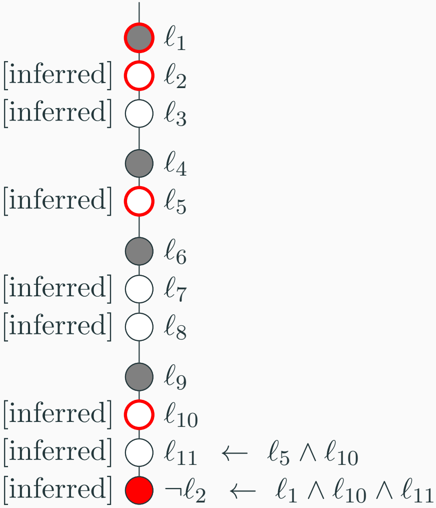
Conflict:

$$\ell_2 \wedge \neg\ell_2 \rightarrow \bot$$

$$\ell_2 \wedge (\ell_1 \wedge \ell_{10} \wedge \ell_{11}) \rightarrow \bot \qquad \text{(resolved } \neg\ell_2\text{)}$$

$$\ell_2 \wedge (\ell_1 \wedge \ell_{10} \wedge (\ell_5 \wedge \ell_{10}) \rightarrow \bot \qquad \text{(resolved } \ell_{11}\text{)}$$

CP-SAT Search: Conflict Directed Clause Learning (CDCL)



Conflict:

$$\ell_2 \wedge \neg\ell_2 \rightarrow \bot$$

$$\ell_2 \wedge (\ell_1 \wedge \ell_{10} \wedge \ell_{11}) \rightarrow \bot \qquad \text{(resolved } \neg\ell_2\text{)}$$

$$\ell_2 \wedge (\ell_1 \wedge \ell_{10} \wedge (\ell_5 \wedge \ell_{10}) \rightarrow \bot \qquad \text{(resolved } \ell_{11}\text{)}$$

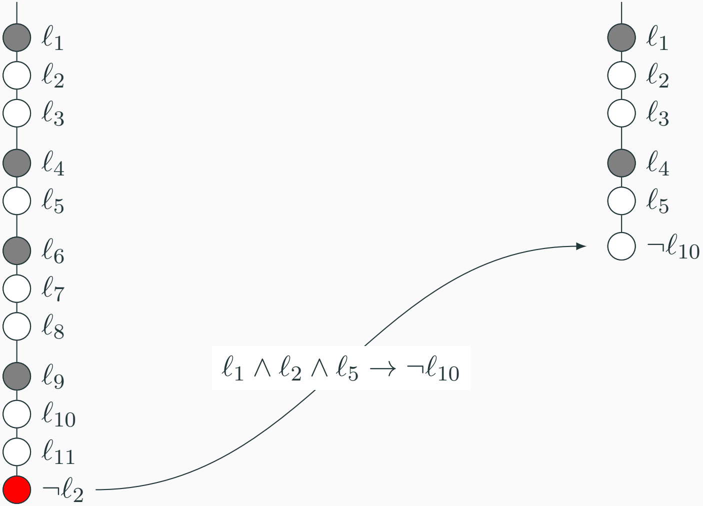
$$\ell_1 \wedge \ell_2 \wedge \ell_5 \wedge \ell_{10} \rightarrow \bot \qquad \text{(reorganized)}$$

ℓ_{10}: Unique Implication Point (UIP)

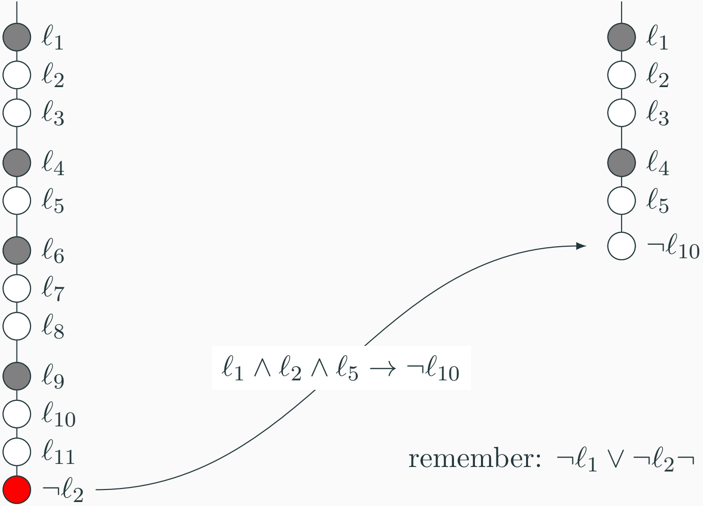
Asserting clause:

$$\ell_1 \wedge \ell_2 \wedge \ell_5 \rightarrow \neg\ell_{10}$$

Search: Backjumping

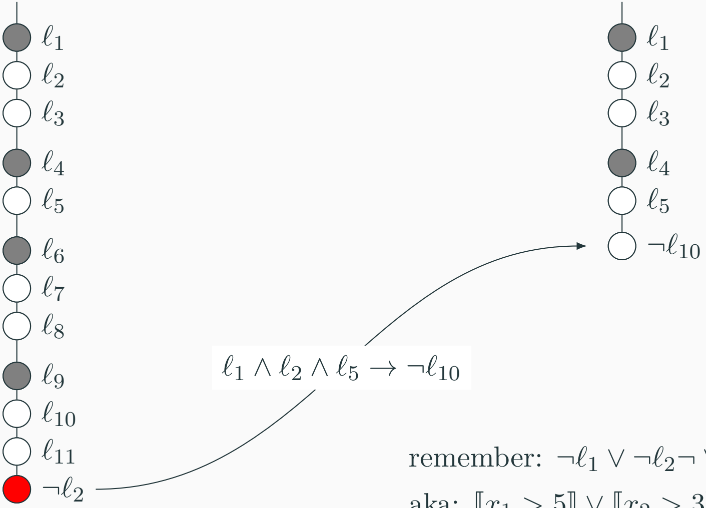


Search: Backjumping



remember: $\neg\ell_1 \vee \neg\ell_2\neg \vee \ell_5 \vee \neg\ell_{10}$

Search: Backjumping



remember: $\neg\ell_1 \vee \neg\ell_2 \neg \vee \ell_5 \vee \neg\ell_{10}$

aka: $[\![x_1 > 5]\!] \vee [\![x_2 > 3]\!] \vee [\![x_4 \leq 7]\!] \vee [\![x_2 > 1]\!]$

Bound Literals

Hybrid CP-SAT solvers:

- (lazily) create **boolean variable** for every **bound literal**
- **Maintain consistency** of boolean variables with the underlying integer variable

Bound Literals

Hybrid CP-SAT solvers:

- (lazily) create **boolean variable** for every **bound literal**
- **Maintain consistency** of boolean variables with the underlying integer variable

To handle: $[\![x_1 > 5]\!] \vee [\![x_2 > 3]\!] \vee [\![x_4 \leq 7]\!] \vee [\![x_2 > 1]\!]$

- Create boolean variables
  - $\ell_1 : [\![x_1 > 5]\!]$
  - $\ell_2 : [\![x_2 > 3]\!]$
  - $\ell_3 : [\![x_4 \leq 7]\!]$
  - $\ell_4 : [\![x_2 > 1]\!]$
- Post the disjunctive constraint
  - $\ell_1 \vee \ell_2 \vee \ell_3 \vee \ell_4$
- Maintain consistency of $\ell_1/x_1, \ell_2/x_2, \ell_3/x_4, \ell_4/x_2$

Bound Literals

Aries: Reasons **natively** on **bound literals**

$$[\![x_1 > 5]\!] \vee [\![x_2 > 3]\!] \vee [\![x_4 \leq 7]\!] \vee [\![x_2 > 1]\!]$$

is just a **regular disjunctive constraint** with no need for **intermediate boolean variables**

Bound Literals

Aries: Reasons **natively** on **bound literals**

$$[\![x_1 > 5]\!] \lor [\![x_2 > 3]\!] \lor [\![x_4 \leq 7]\!] \lor [\![x_2 > 1]\!]$$

is just a **regular disjunctive constraint** with no need for **intermediate boolean variables**

- No variable creation
- No synchronization costs

Bound Literals

Aries: Reasons **natively** on **bound literals**

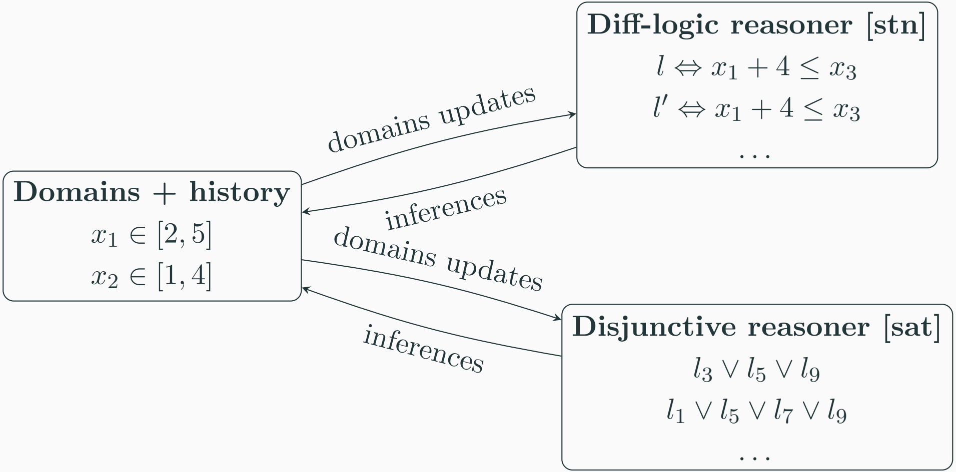
$$[\![x_1 > 5]\!] \vee [\![x_2 > 3]\!] \vee [\![x_4 \leq 7]\!] \vee [\![x_2 > 1]\!]$$

is just a **regular disjunctive constraint** with no need for **intermediate boolean variables**

- No variable creation
- No synchronization costs

Downside: pervasive change

Structure of Propagators: SMT-like ”reasoners”



~~Difference-Logic Reasoner (aka DTN propagator)~~ [2]

[2] SMT-like, organized propagation + few tricks

Search Strategy: Decision Variables

Which decision variables to branch on?

- $t_i \in \mathbb{N}$: start times
- $o_{ij} \in \{0, 1\}$: task ordering
  - $[\![o_{ij} \geq 1]\!] \Leftrightarrow t_i + d_i \leq t_j$
  - $[\![o_{ij} \leq 0]\!] \Leftrightarrow t_j + d_j \leq t_i$

Search Strategy: Decision Variables

Which decision variables to branch on?

- $t_i \in \mathbb{N}$: start times
- $o_{ij} \in \{0, 1\}$: task ordering
  - $[\![o_{ij} \geq 1]\!] \Leftrightarrow t_i + d_i \leq t_j$
  - $[\![o_{ij} \leq 0]\!] \Leftrightarrow t_j + d_j \leq t_i$

Variable selection: Learning Rate Branching (LRB)

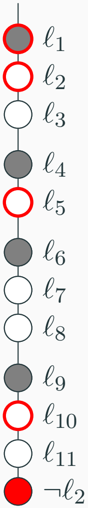
Key idea: maximize the number of conflicts per decision
$\hookrightarrow$ prefer literals that participate in conflicts

Variable selection: Learning Rate Branching (LRB)

Key idea: maximize the number of conflicts per decision
↪ prefer literals that participate in conflicts

Literals that participate in the conflict are:

- clause literals: $\ell_1, \ell_2, \ell_5, \ell_{10}$

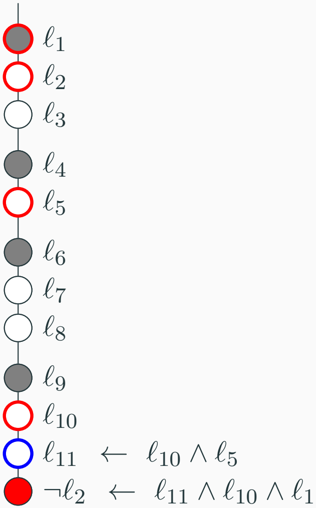


Variable selection: Learning Rate Branching (LRB)

Key idea: maximize the number of conflicts per decision
$\hookrightarrow$ prefer literals that participate in conflicts

Literals that participate in the conflict are:

- clause literals: $\ell_1, \ell_2, \ell_5, \ell_{10}$
- resolved literals: ℓ_2, ℓ_{11}
  (seen while building the clause)



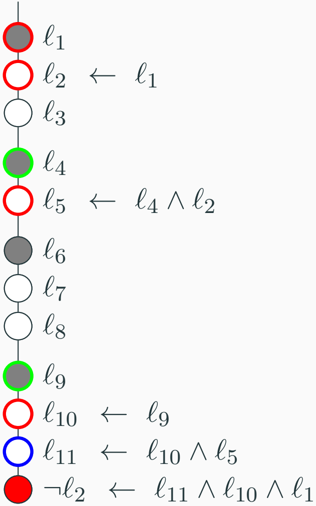
Variable selection: Learning Rate Branching (LRB)

Key idea: maximize the number of conflicts per decision

↪ prefer literals that participate in conflicts

Literals that participate in the conflict are:

- clause literals: $\ell_1, \ell_2, \ell_5, \ell_{10}$
- resolved literals: ℓ_2, ℓ_{11}
  (seen while building the clause)
- reasoned literals: ℓ_1, ℓ_4, ℓ_9
  (reason of literals in the clause)



Search Strategy: full picture

- **Variable selection: Learning Rate Branching (LRB)**
  - Favors conflictual variables
- **Value selection: solution guided**
  - take value from last solution
  - search focused in its neighborhood
- **Greedy Search initialization**
  - **until first conflict**, assign start-time variables
  - Choice: Earliest-starting time, min-slack

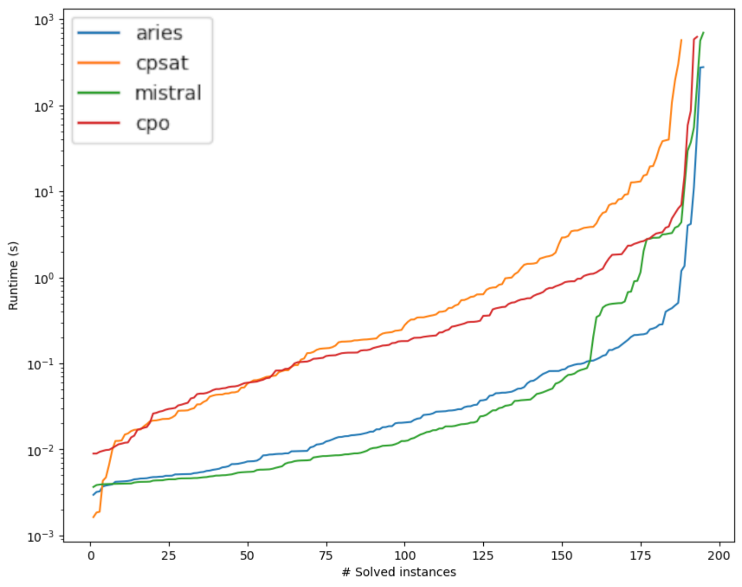
Performance Evaluation: Contenders

State-of-the-art CP solvers for disjunctive scheduling.

- MISTRAL: Pure CP for disjunctive scheduling
- CPOPTIMIZER: SOTA CP Scheduling solver
- CPSAT: SOTA Hybrid CP-SAT solver (OrTools)

Evaluated on common **OpenShop** and **JobShop** instances

OpenShop: runtimes (cactus plot)

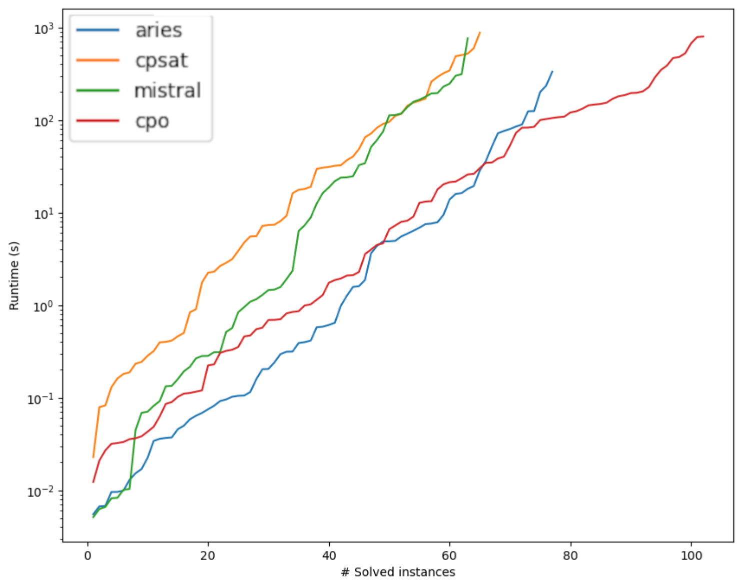


Runtime necessary to find proven optimal solutions.

Aries & Mistral:

- All problems solved
- Aries systematically faster when more than 1 second is needed

Jobshop: runtimes (cactus plot)



Runtime necessary to find proven optimal solutions.

ARIES & CPOPTIMIZER:

- Aries fails to prove optimality for the harder instances
- Solution Quality difference is of 0.2%
- Aries generally substantially faster

Conclusion

Aries on disjunctive scheduling **milestone**:

- simple and generic design
- state-of-the-art performance
- extensible CP core

Conclusion

Aries on disjunctive scheduling **milestone**:

- simple and generic design
- state-of-the-art performance
- extensible CP core

...on the road to temporal planning

- resource reasoning
- optional reasoning

Aries for Temporal Planning

Action Template:

$$
\begin{aligned}
&\mathsf{move}(r, a, b) \\
&\text{variables:} \quad r,\ a,\ b,\ t_{start},\ t_{end} \\
&\text{constraints:} \quad a \neq b \\
&\qquad t_{end} - t_{start} = \mathit{travel\text{-}time}(a, b) \\
&\text{conditions:} \quad [t_{start}]\ loc(r) = a \\
&\text{effects:} \quad]t_{start}, t_{end}]\ loc(r) \leftarrow b
\end{aligned}
$$

From actions to CSP

For each action template:

- generate a bounded set of *optional activities*
- each with its own parameters (decision variables)

$$move^1(r^1, a^1, b^1) : p^1$$
$$move^2(r^2, a^2, b^2) : p^2$$

- constraints ensure coherence of activities + goal achievement

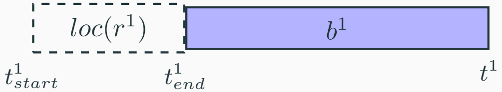
Problem as a dummy action

$$
\begin{aligned}
&\textit{Problem}\ (\mathcal{C}_0) \\
\text{variables:}\quad & t, \ell \\
\text{constraints:}\quad & t < 100 \\
& \ell = \textit{London} \lor \ell = \textit{Dublin} \\
\text{conditions:}\quad & [t, t]\ loc(Bob) = Toulouse \qquad \leftarrow \textbf{end condition} \\
\text{effects:}\quad & [0, 0]\ loc(Bob) \leftarrow Toulouse \qquad \leftarrow \textbf{init. state}
\end{aligned}
$$

Decomposition validity: effect token

Effect token

$$move^1(r^1, a^1, b^1)$$
$$\dots$$
$$\text{effects:} \quad]t^1_{start}, t^1_{end}]\, loc(r^1) \leftarrow b^1$$
$$\dots$$



Here: t^1 is a lower bound on the persistence of the effect.

Decomposition validity: condition token

Condition token

$$move^1(r^1, a^1, b^1) : p^1$$

$$\dots$$

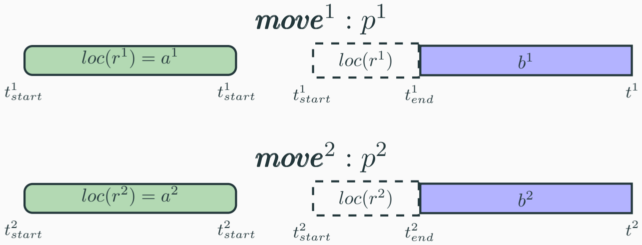
$$\text{conditions:} \quad [t^1_{start}]loc(r^1) = a^1$$

$$\dots$$

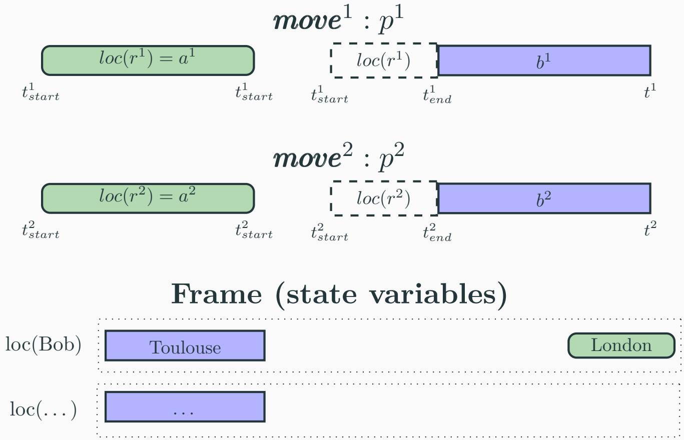
$loc(r^1) = a^1$

t^1_{start} t^1_{start}

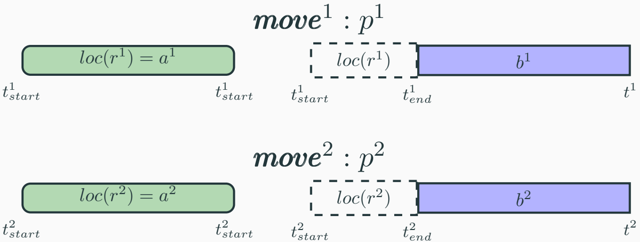
Planning Puzzle



Planning Puzzle



Planning Puzzle

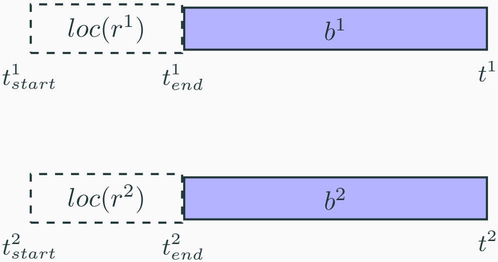


Frame (state variables)

Degrees of freedom: presence (p*) state variable (r*)
time (t*) value (a*/b*)

Rule 1: No overlapping effects (coherence)

No two overlapping effects



$$p^1 \land p^2 \implies t^1 \leq t^2_{start} \lor \\ t^2 \leq t^1_{start} \lor \\ r^1 \neq r^2$$

Rule 2: All conditions are *supported* (support)

For any condition C^1, there is an effect E^2 such that:

$$p^2$$

$$t^2_{end} \leq t^1_{start} \wedge t^1_{end} \leq t^2$$

$$r^1 = r^2$$

$$a^1 = b^2$$

("each green must be in a blue")

Bounded planning problem as CSP

CSP (X, C) where

- X = all variables in chronicles + chronicle presence
    - $\{ r^1, \ldots, p^1 \} \cup \{ r^2, \ldots, p^2 \} \cup \{ t, \ell \}$
- C = all constraints:
    - coherence
    - support
    - $\ldots$

Bounded planning problem as CSP

CSP (X, C) where

- X = all variables in chronicles + chronicle presence
  - $\{ r^1, \dots, p^1 \} \cup \{ r^2, \dots, p^2 \} \cup \{ t, \ell \}$
- C = all constraints:
  - coherence
  - support
  - ...
  - refinement (HTN) / symmetry breaking (generative)
  - resources (numeric)

$\Rightarrow$ handed over to Aries' CP/SAT solver

Aries planner

Planner features

- Generative or Hierarchical planning
- Durative actions
- Timed effects (TILs), timed goals (deadline)
- (Simple) Numeric Planning
- Optimization (length, costs, makespan, final value)

Integration in Unified Planning

- python library
- modeling/solving
- from AIPlan4EU project

Aries: at the state-of-the art?

- When action space is constrained (hierarchical non-recursive)
- temporally expressive (deadlines, time-windows)
- when quality matters

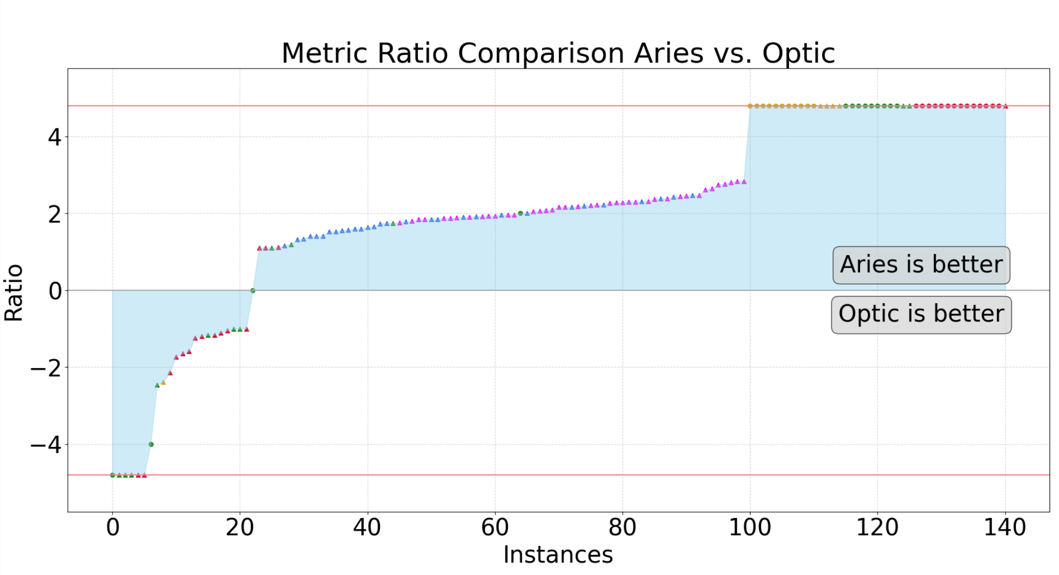
Aries: at the state-of-the art?

- When action space is constrained (hierarchical non-recursive)
- temporally expressive (deadlines, time-windows)
- when quality matters

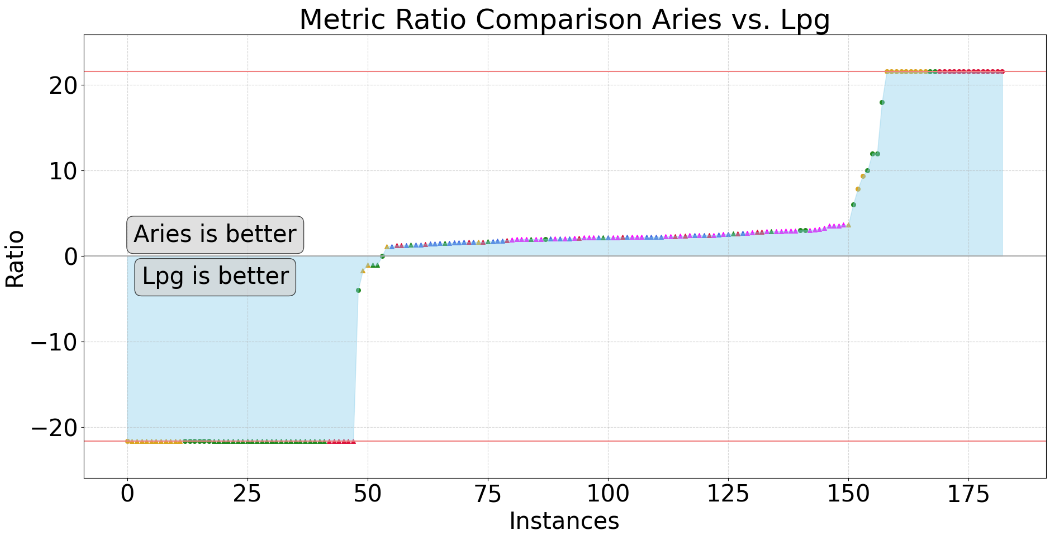
Hierarchical: no competitors Generative, compared to:

- Optic, (much) better coverage, higher quality
- LPG, lower coverage, higher quality

Aries vs Optic: solution quality (temporal+numeric problems)



Aries vs LPG: solution quality (temporal+numeric problems)



References

- Arthur Bit-Monnot. Enhancing Hybrid CP-SAT Search for Disjunctive Scheduling. ECAI 2023.
- Arthur Bit-Monnot. Experimenting with Lifted Plan-Space Planning as Scheduling: Aries in the 2023 IPC. 2023 International Planning Competition (2023)
- Roland Godet, Arthur Bit-Monnot. Chronicles for Representing Hierarchical Planning Problems with Time. ICAPS Hierarchical Planning Workshop (HPlan 2022),