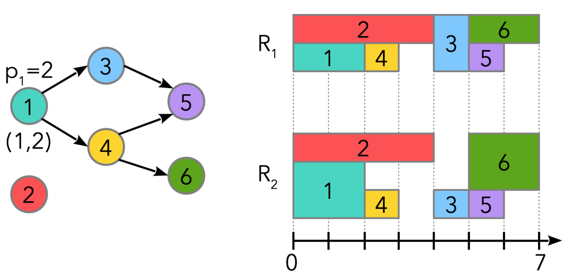
CP Optimizer

A generic optimization engine for solving industrial scheduling problems

Philippe Laborie, IBM
laborie@fr.ibm.com



What is CP Optimizer ?

- Historically developed since 2007 by ILOG, now IBM
- Our team has 20+ years of experience in designing combinatorial optimization tools for **real-life industrial problems**, and particularly **scheduling** problems
- #1 objective of CP Optimizer : **lower the barrier to entry for efficiently solving industrial scheduling problems**
- In particular, no need to be an OR, AI or algorithmist expert

What is CP Optimizer ?

Model & run approach:

- User focuses on a **declarative mathematical model** of the problem using the classical ingredients of combinatorial optimization: variables, constraints, expressions, objective function
- Resolution is performed by an **automated search** algorithm with the following properties: complete, deterministic, anytime, efficient, robust, continuously improving ...

Scheduling is about **time** ...

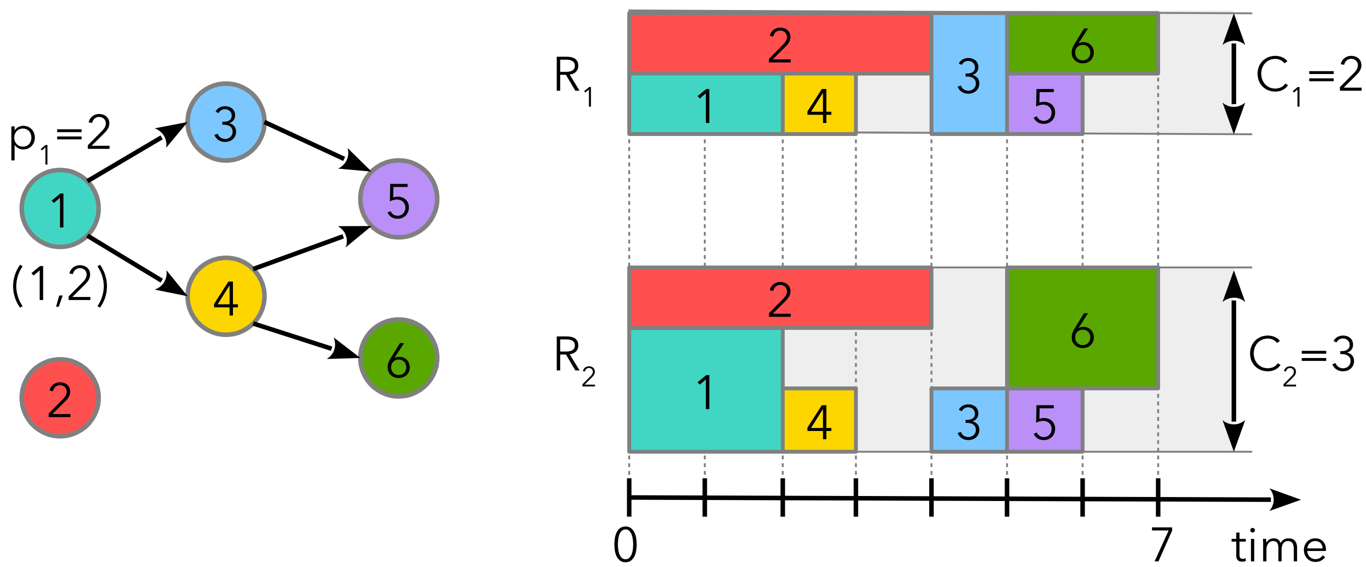
- Most existing framework in combinatorial optimization (MILP, classical CP) only deal with numerical values ($x \in \mathbb{R}$, $x \in \mathbb{Z}$)
- CP Optimizer introduces a set of other simple **mathematical** concepts that naturally emerge when dealing with **time**:
  - ***Intervals*** : $a = [s,e) = \{ x \in \mathbb{R} \mid s \leq x < e \}$
  - ***Functions*** : $f: \mathbb{R} \rightarrow \mathbb{Z}$
  - ***Permutations***
  - ***Optional interval*** : occurrence / non-occurrence of an event

What is CP Optimizer ?

- What if we integrate these mathematical concepts in the model …
- And keep all the good ideas of well established frameworks like MILP:
  - Model & run
  - Exact algorithm
  - Input/output file format
  - Language versatility (C++, Python, Java, C#, OPL)
  - Modeling assistance (warnings, …)
  - Conflict refiner for explaining infeasibility
  - Warm-start
  - …
- That’s exactly what CP Optimizer is about !

Examples of scheduling problems

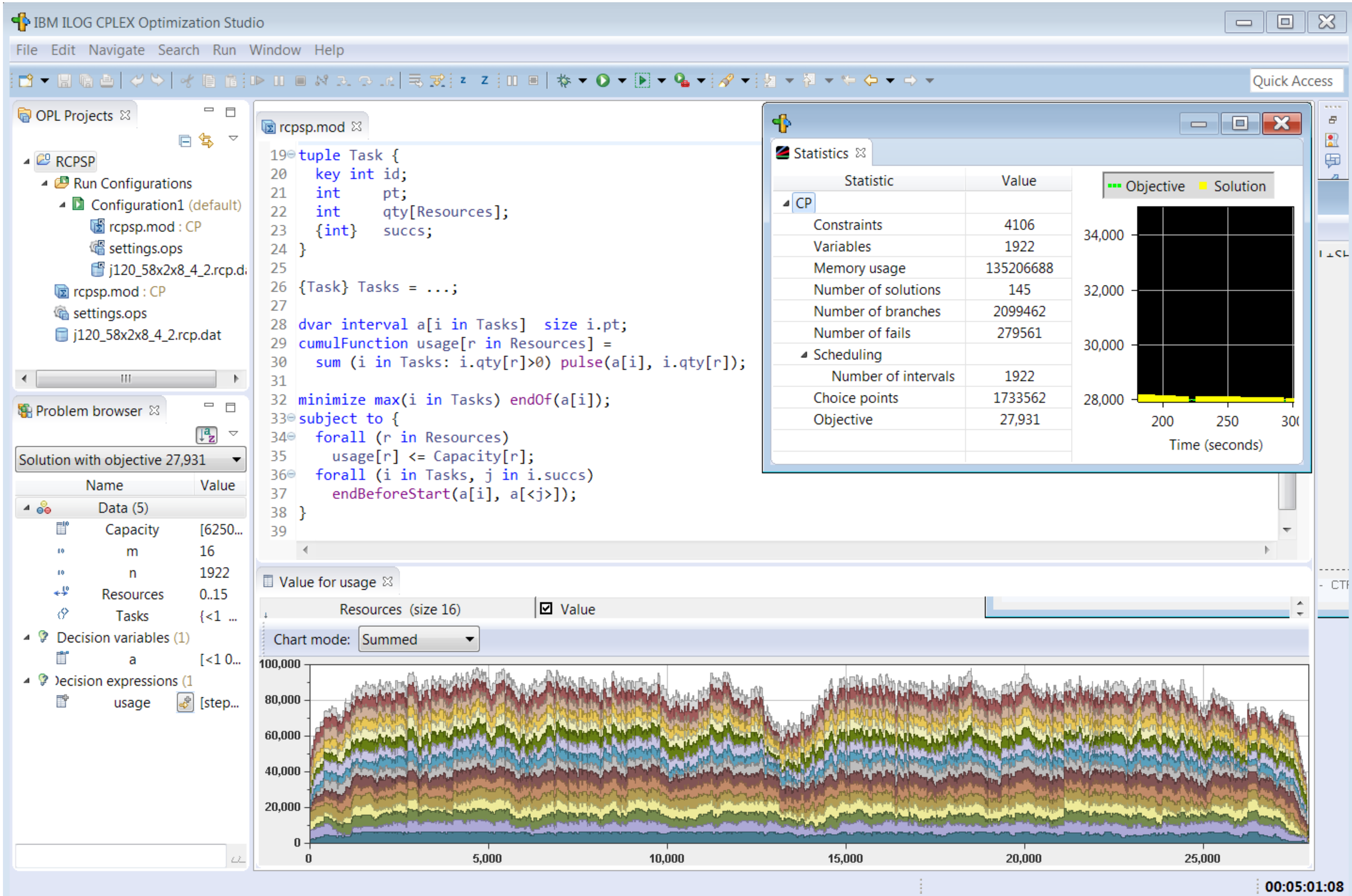
- Resource-Constrained Project Scheduling (RCPSP)
  - Notorious NP-Hard problem in combinatorial optimization (>5000 references on Google Scholar)
  - N tasks with precedence constraints
  - M resources of limited capacity
  - Minimize project makespan



CP Optimizer model for RCPSP

$$
\begin{array}{lll}
\min & \max_{i \in I} \ \text{endOf}(a_i) & \\
 & \text{endBeforeStart}(a_i, a_j) & \forall (i,j) \in E \\
 & \sum_{i \in I} \text{pulse}(a_i, d_{ir}) \leq D_r & \forall r \in R \\
 & a_i \ \text{interval of size } p_i & \forall i \in I
\end{array}
$$

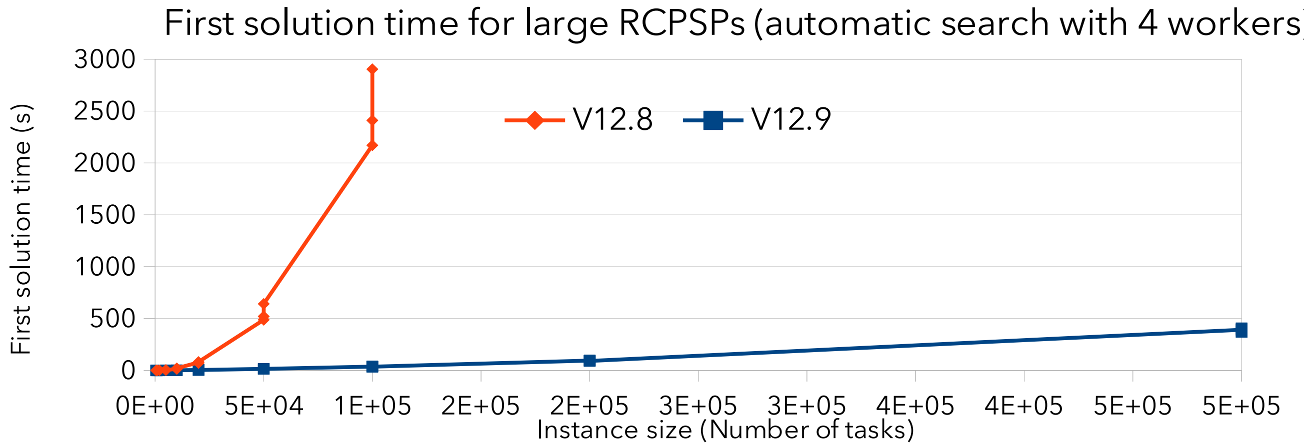
CP Optimizer model for RCPSP



Performance on classical scheduling benchmarks

- Results published in CPAIOR-2015 (using V12.6) on classical benchmarks:
  - Many instances **closed** or **improved** on : Job-shop, Job-shop with operators , Flexible job-shop, RCPSP, RCPSP with maximum delays, Multi-mode RCPSP, Multi-mode RCPSP with maximum delays, ...
  - These classical benchmarks are **small** compared to real industrial problems

- New benchmark on RCPSP (from 500 to 500.000 tasks)



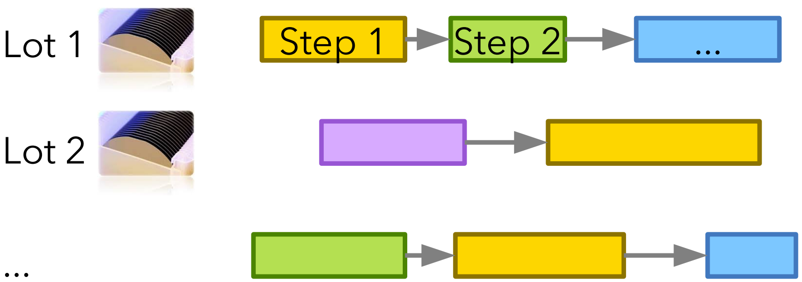
Examples of scheduling problems

- In the real life, scheduling problems are more complex
  - Complex constraints: activities, resources
  - Complex objectives: resource costs, tardiness, throughput
  - Ill-defined problems
  - Over-constrained problems (can’t schedule all activities)
  - Heterogeneous decisions (time, resource allocation, batching, …)
  - Large (e.g. 1.000.000 tasks)
  - Require fast solving time

CP Optimizer modeling concepts

- Allows easy modeling of:
  - Variable activity duration, partially preemptive tasks
  - Optional activities, oversubscribed problems
  - Hierarchical problems (Work Breakdown Structures)
  - Alternative resources and modes (MM-RCPSP)
  - Resource calendars and breaks
  - Cumulative resources, inventories, reservoirs
  - Parallel batches, activity incompatibilities
  - Unary resources with setup times and costs
  - Complex objective functions
- In CP Optimizer the size of the model in general grows linearly with the size of the problem instance

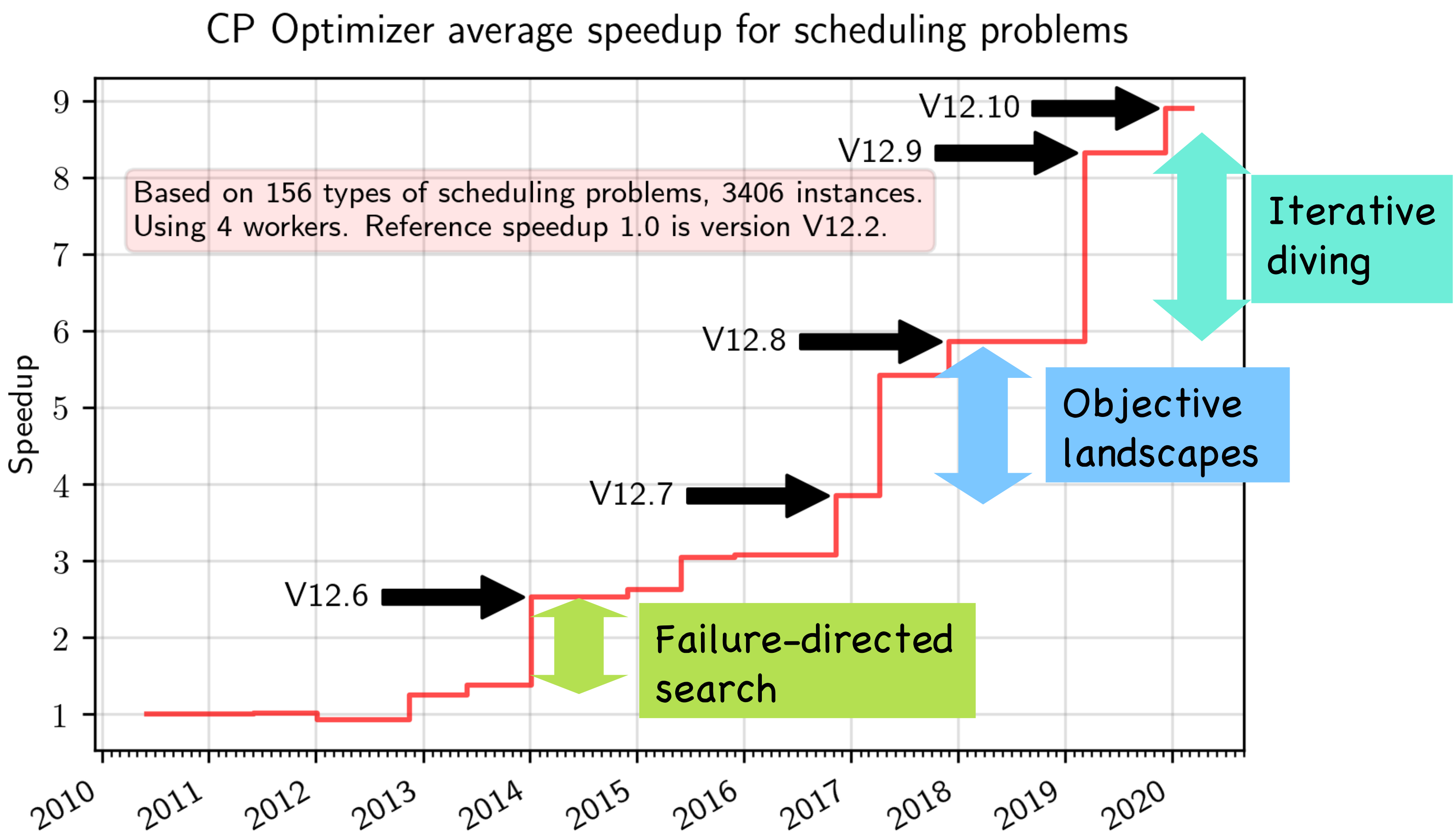
CP Optimizer model for semiconductor manufacturing



S. Knopp et al. Modeling Maximum Time Lags in Complex Job-Shops with Batching in Semiconductor Manufacturing. PMS 2016.

```
using CP;
tuple Lot { key int id; int n; float w; int rd; int dd; }
tuple Stp { key Lot l; key int pos; int f; }
tuple Lag { Lot l; int pos1; int pos2; int a; int b; float c; }
tuple Mch { key int id; int capacity; }
tuple MchFml { Mch m; int f; int pt; }
tuple MchStp { Mch m; Stp s; int pt; }
tuple Setup { int f1; int f2; int dur; }

{Lot} Lots = ...;
{Stp} Stps = ...;
{Lag} Lags = ...;
{Mch} Mchs = ...;
{MchFml} MchFmls = ...;
{Setup} MchSetups[m in Mchs] = ...;

{MchStp} MchStps = {<c.m,s,c.pt> | s in Stps, c in MchFmls: c.f==s.f};

dvar interval lot[l in Lots] in l.rd..48*60;
dvar interval stp[s in Stps];
dvar interval mchStp[ms in MchStps] optional size ms.pt;

dvar int lag[Lags];

stateFunction batch[m in Mchs] with MchSetups[m];
cumulFunction load [m in Mchs] =
  sum(ms in MchStps: ms.m==m) pulse(mchStp[ms],ms.s.l.n);

minimize staticLex(
  sum(d in Lags) minl(d.c, d.c*maxl(0,lag[d]-d.a)^2/(d.b-d.a)^2),
  sum(l in Lots) l.w*maxl(0, endOf(lot[l])-l.dd));
subject to {
  forall(l in Lots)
    span(lot[l], all(s in Stps: s.l==l) stp[s]);
  forall(s in Stps) {
    alternative(stp[s], all(ms in MchStps: ms.s==s) mchStp[ms]);
    if (s.pos>1)
      endBeforeStart(stp[<s.l,s.pos-1>],stp[s]);
  }
  forall(ms in MchStps)
    alwaysEqual(batch[ms.m], mchStp[ms], ms.s.f, true, true);
  forall(m in Mchs)
    load[m] <= m.capacity;
  forall(d in Lags)
    endAtStart(stp[<d.l,d.pos1>], stp[<d.l,d.pos2>], lag[d]);
}
```

Performance evolution



Under the hood

Artificial Intelligence **Operations Research**

Heuristics Model presolve

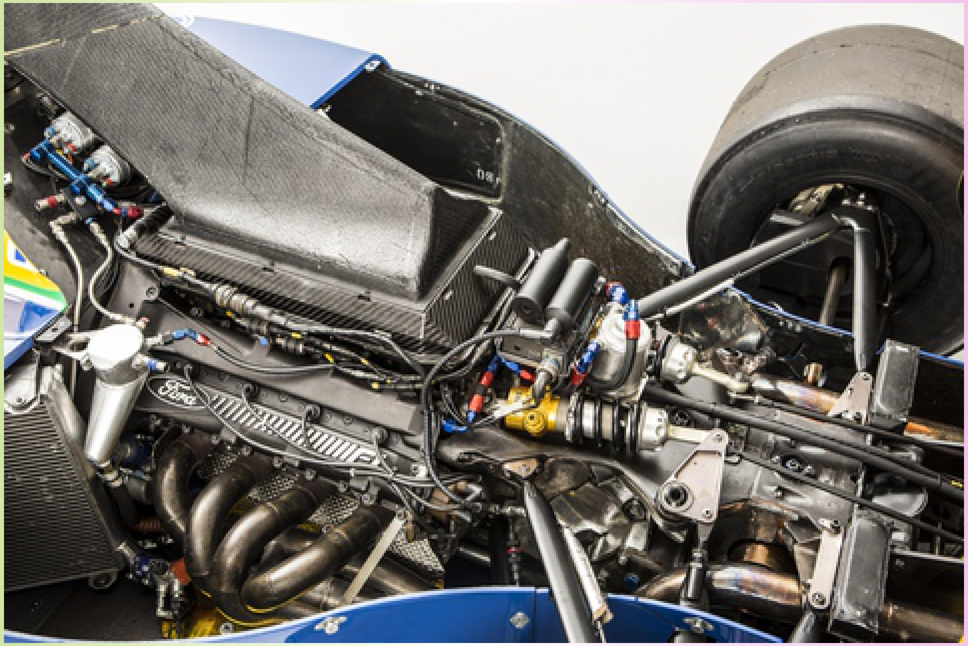
Constraint propagation

Learning

Temporal constraint networks

2-SAT networks

No-goods



Linear relaxations

Problem specific scheduling algorithms

Restarts Tree search

LNS Randomization

Some references

- P. Laborie, J. Rogerie. Reasoning with Conditional Time-Intervals. In: Proc. FLAIRS-2008, p555-560.
- P. Laborie, J. Rogerie, P. Shaw, P. Vilím. Reasoning with Conditional Time-Intervals. Part II: An Algebraical Model for Resources. In: Proc. FLAIRS-2009, p201-206.

Modeling concepts

- P. Laborie. CP Optimizer for detailed scheduling illustrated on three problems. In: Proc. CPAIOR-2009.
- P. Laborie, B. Messaoudi. New Results for the GEO-CAPE Observation Scheduling Problem. In Proc. ICAPS-2017.

Examples

- P. Laborie, D. Godard. Self-Adapting Large Neighborhood Search: Application to Single-Mode Scheduling Problems. In: Proc. MISTA-2007.
- P. Laborie, J. Rogerie. Temporal Linear Relaxation in IBM ILOG CP Optimizer. Journal of Scheduling 19(4), 391–400 (2016).
- P. Vilím. Timetable Edge Finding Filtering Algorithm for Discrete Cumulative Resources . In: Proc. CPAIOR-2011.
- P. Vilím, P. Laborie, P. Shaw. Failure-directed Search for Constraint-based Scheduling. In: Proc. CPAIOR-2015.

Search algorithm

- P. Laborie, J. Rogerie, P. Shaw, P. Vilím. IBM ILOG CP Optimizer for Scheduling. Constraints Journal (2018).

Overview

Conclusion

- Try CP Optimizer !
- Full-version of CP Optimizer is **free** for academics: **ibm.com/academic**