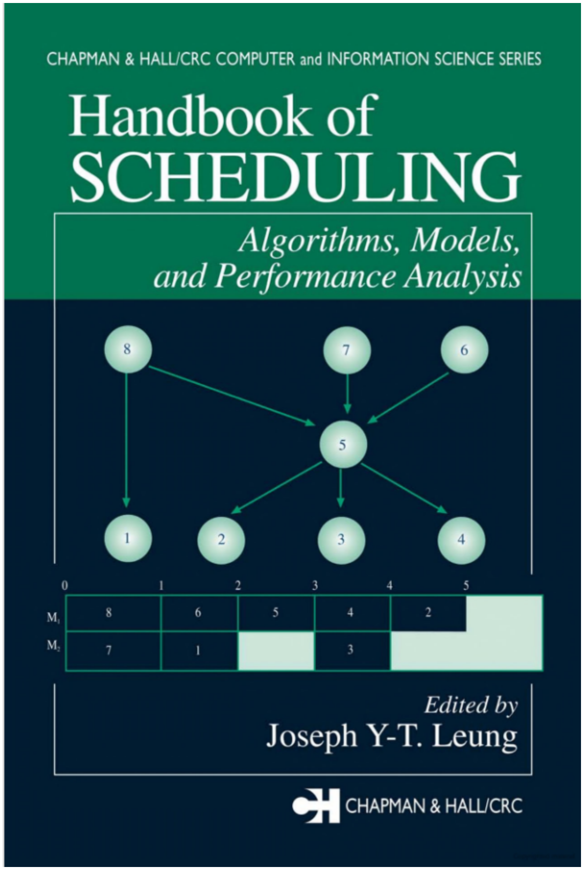
CP Optimizer pour la planification et l'ordonnancement

Philippe Laborie
IBM, IBM Data & AI
laborie@fr.ibm.com

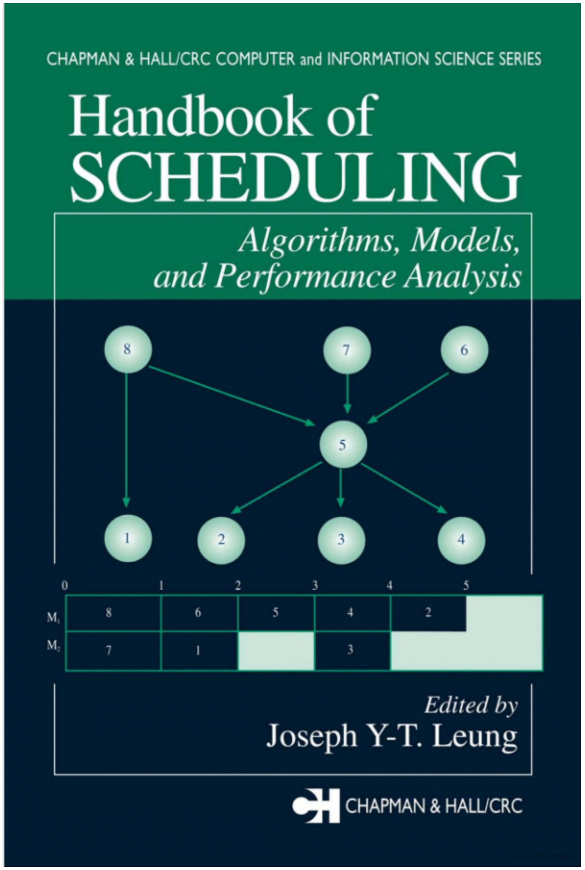
AfIA
Association française pour l'Intelligence Artificielle

What is scheduling ?

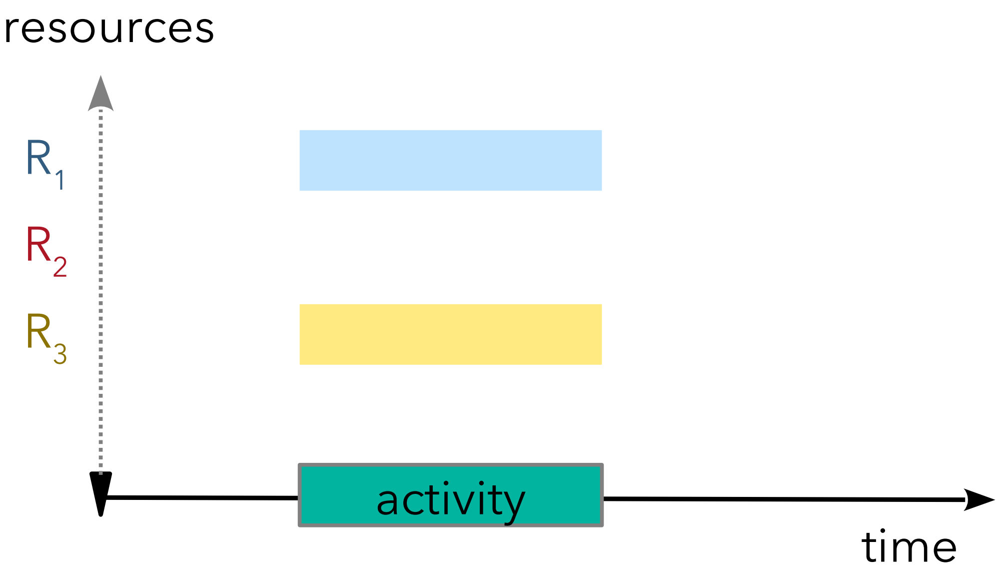


« *Scheduling* is concerned with the allocation of scarce **resources** to **activities** over **time** with the objective of **optimizing** one or more performance measures. »

What is scheduling ?

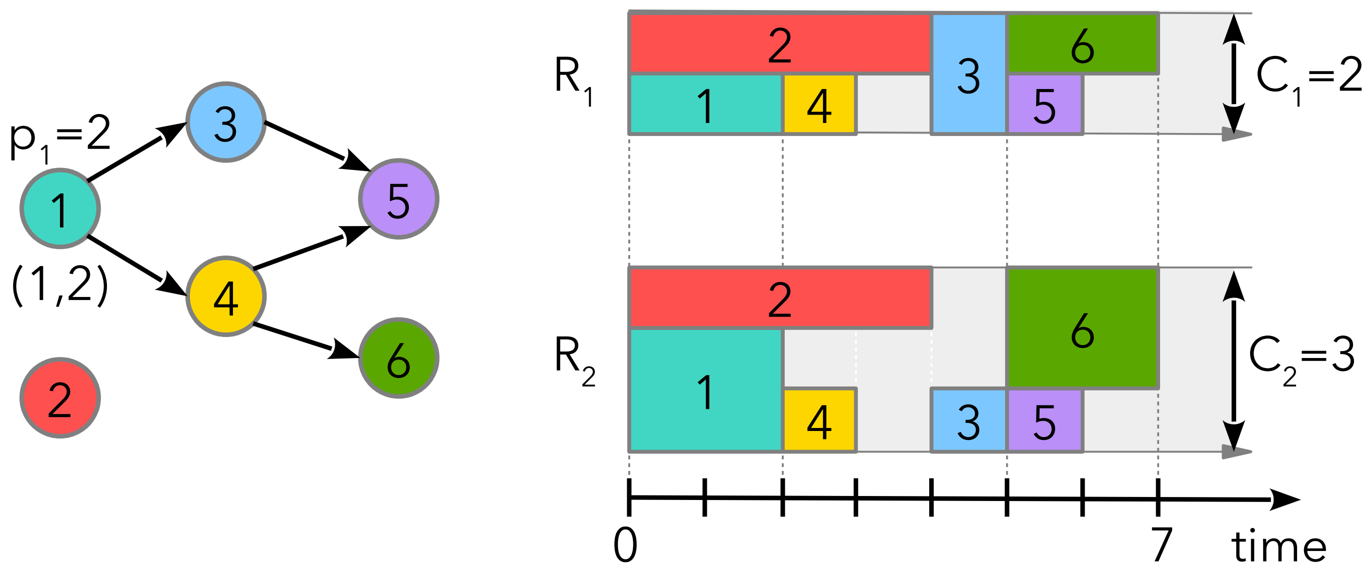


« *Scheduling* is concerned with the allocation of scarce **resources** to **activities** over **time** with the objective of **optimizing** one or more performance measures. »



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

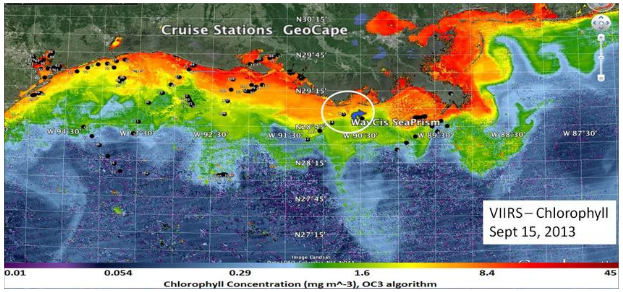


Examples of scheduling problems

- In the real life, scheduling problems are more complex

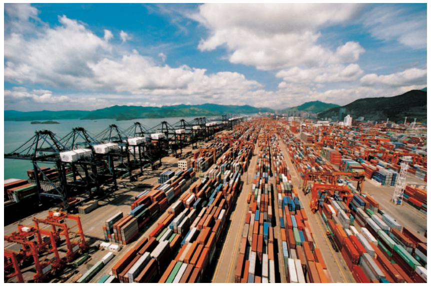




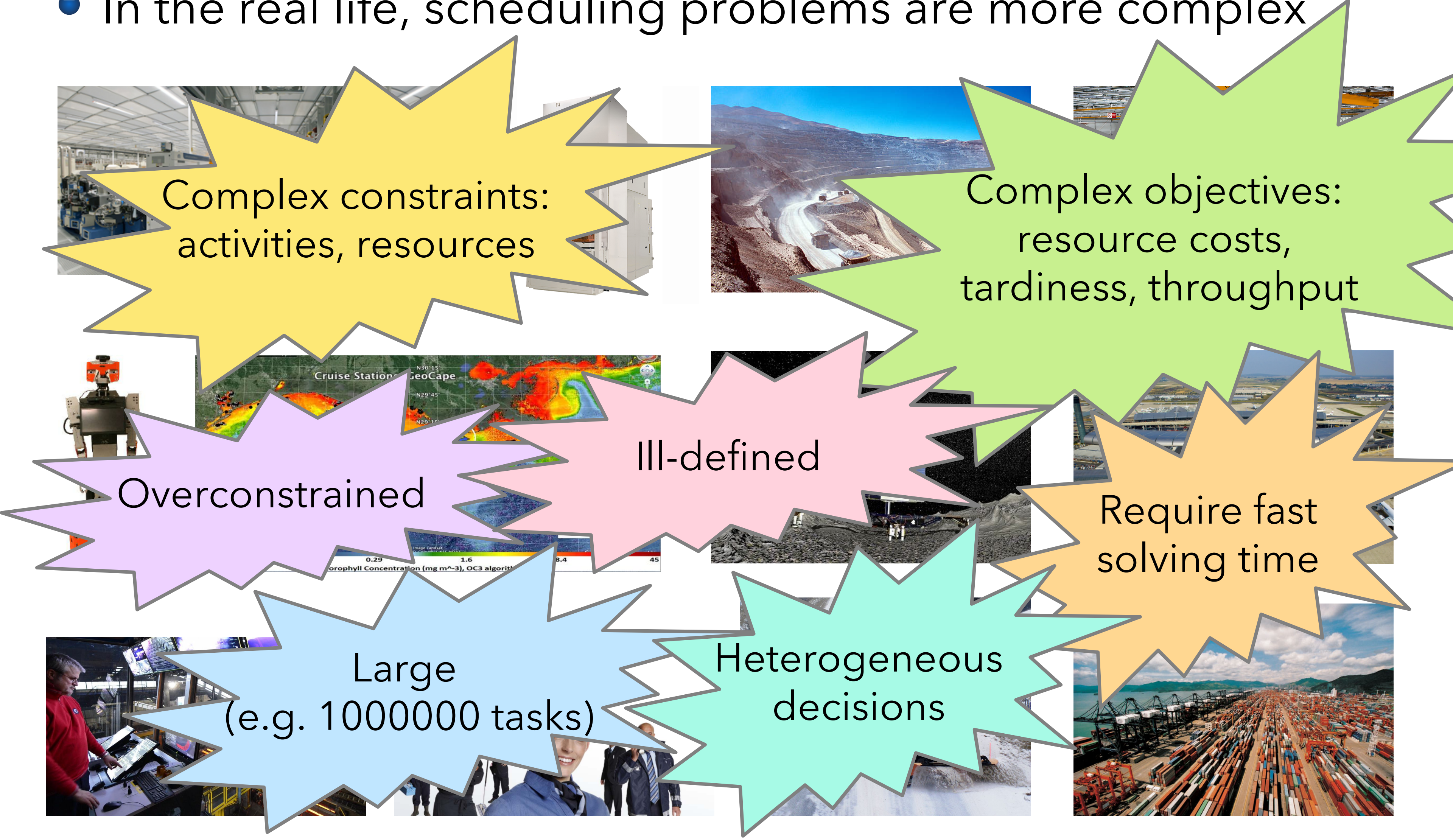




Examples of scheduling problems

- In the real life, scheduling problems are more complex



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**
- Targeted audience goes beyond CP experts:
  - OR experts
  - Data scientists
  - Software engineers

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 ...

What is CP Optimizer ?

Model & run approach: wait … this already exists: it looks like Mixed Integer Linear Programming (MILP) !

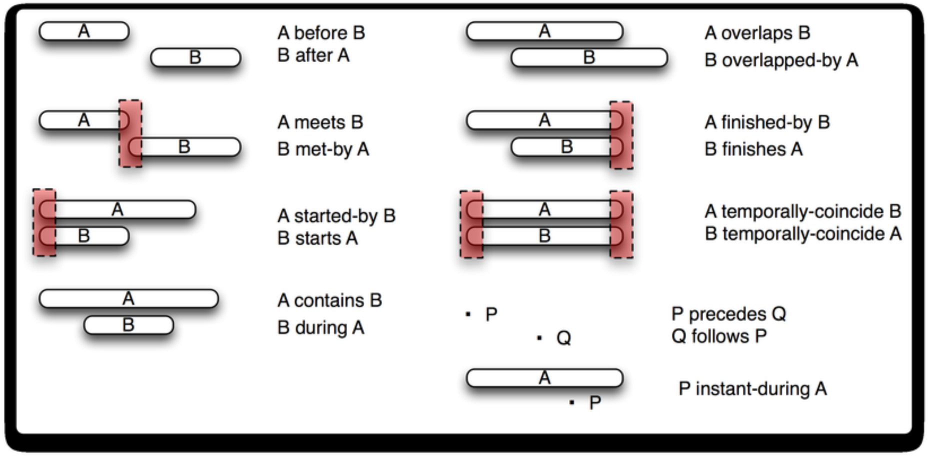
- Right: we borrowed a lot from the MILP paradigm when designing CP optimizer
- Wrong: MILP is usually not good for scheduling problems
  - Difficult to model them
  - Many modeling tricks exist but they are brittle
  - Large models (typically in n^2 or n^3 with problem size)
  - Poor performance, especially on large problems
  - Often it takes long even to get a feasible solution (not anytime)

What's wrong with MILP (and classical CP) for scheduling ?

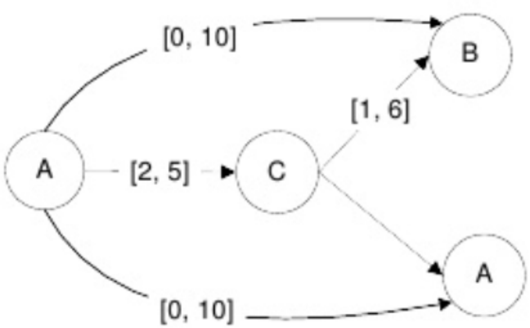
- It's missing the essential ingredient of scheduling: **time**
- Interestingly, **time** is a very relevant topic in AI
  - Temporal reasoning
  - Reasoning on action and change

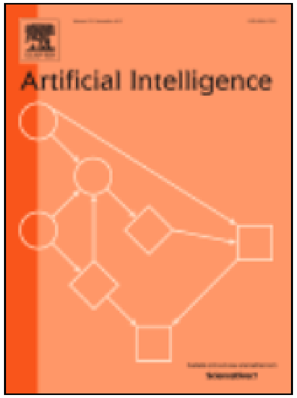
Time in AI: examples

- Allen's interval algebra (1983)



- Temporal constraint networks (1991)

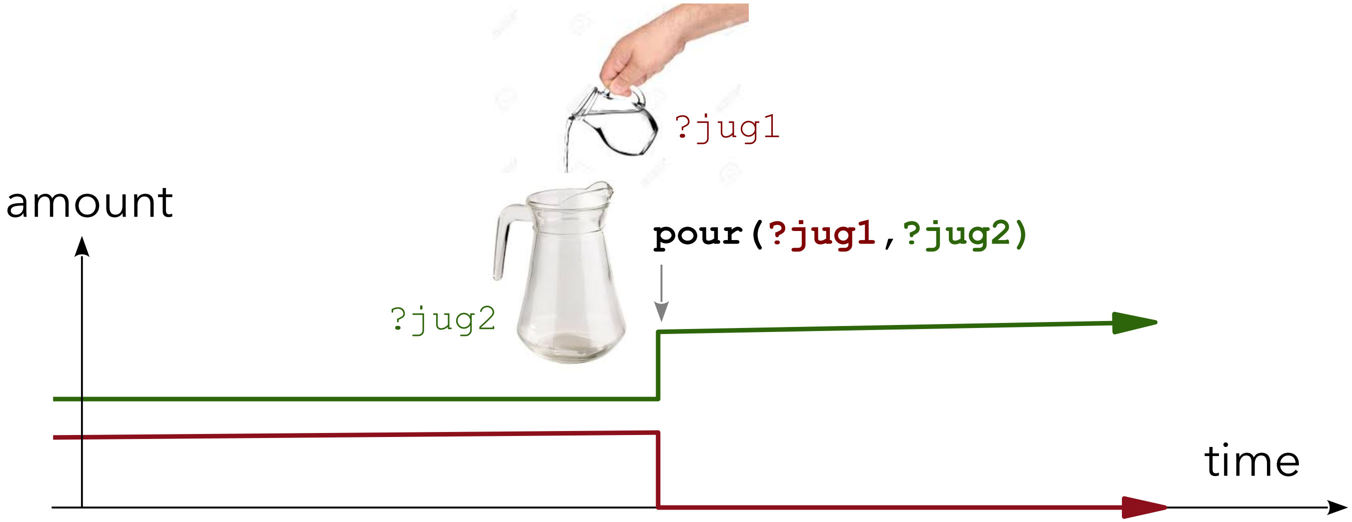




Time in AI: examples

- Temporal planning in PDDL 2.1 (2003)


```
(define (domain jug-pouring)
   (:requirements :typing :fluents)
   (:types jug)
   (:functions
      (amount ?j - jug)
      (capacity ?j - jug))
   (:action pour
      :parameters (?jug1 ?jug2 - jug)
      :precondition (>= (- (capacity ?jug2) (amount ?jug2)) (amount ?jug1))
      :effect (and (assign (amount ?jug1) 0)
                   (increase (amount ?jug2) (amount ?jug1)))
)
```



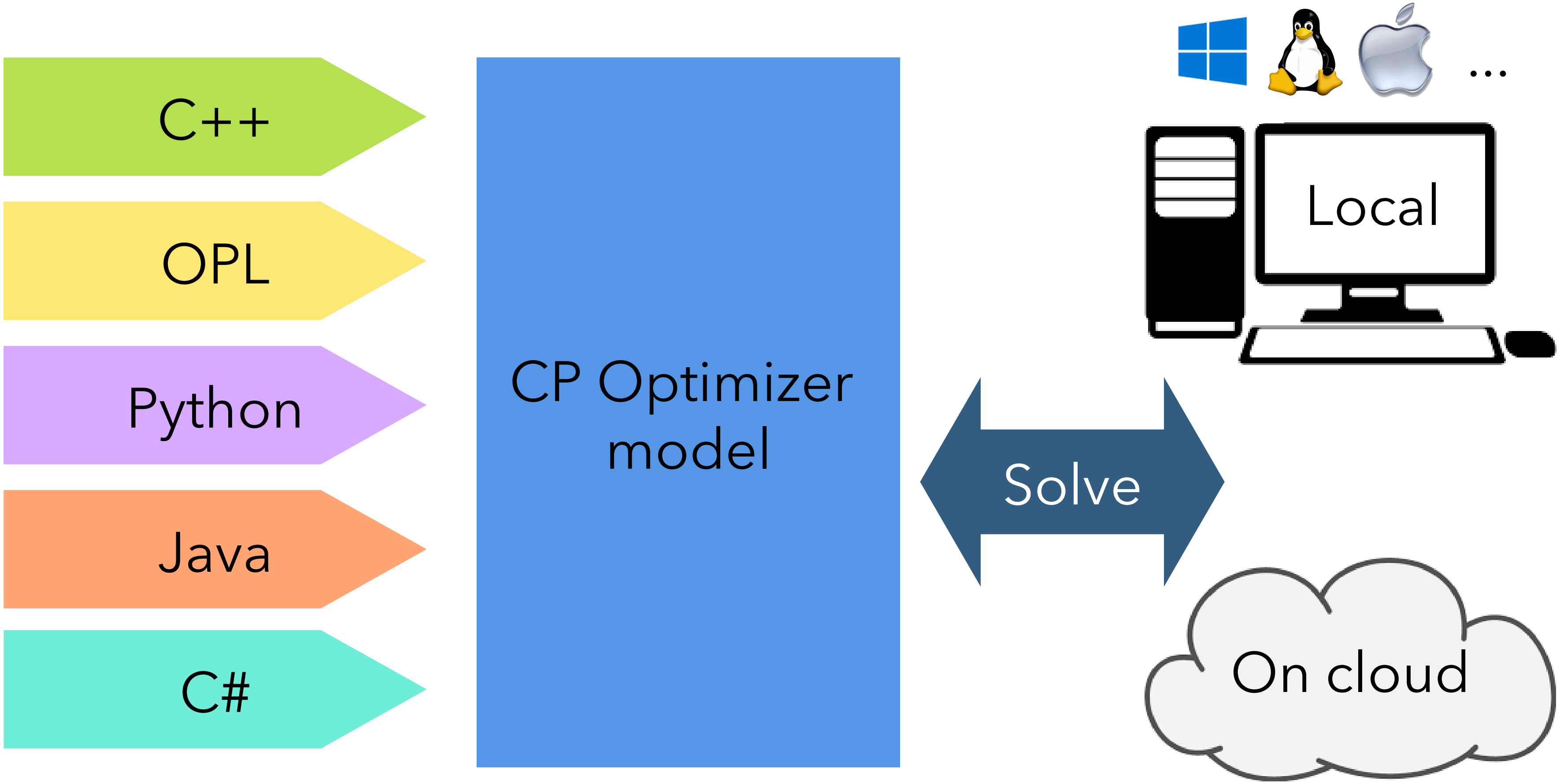
Time in AI: examples

- MILP (and classical CP) models only deal with numerical values ($x \in \mathbb{R}$, $x \in \mathbb{Z}$)
- A set of other simple mathematical concepts seem to 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
  - Occurrence / non-occurrence of an event : optional interval

What is CP Optimizer ?

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

Overview of CP Optimizer



Interval variables

- An **optional** interval variable has an additional possible value in its domain (absence value)
- Domain of values for an optional interval variable x:

$$\mathrm{Dom}(x) \subseteq \{\perp\} \cup \{\, [s,e) \mid s,e \in \mathbb{Z},\ s \leq e \,\}$$

Absent interval

Interval of integers
(when interval is **present**)

- Example: interval x optional in 1000..2000 size 10..20
- Constraints and expressions on interval variables specify how they handle the case of absent intervals (in general it is very intuitive)

CP Optimizer model for RCPSP

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

A set of decision variables of type *interval*

CP Optimizer model for RCPSP

A set of precedence constraints aggregated into a global temporal constraint network

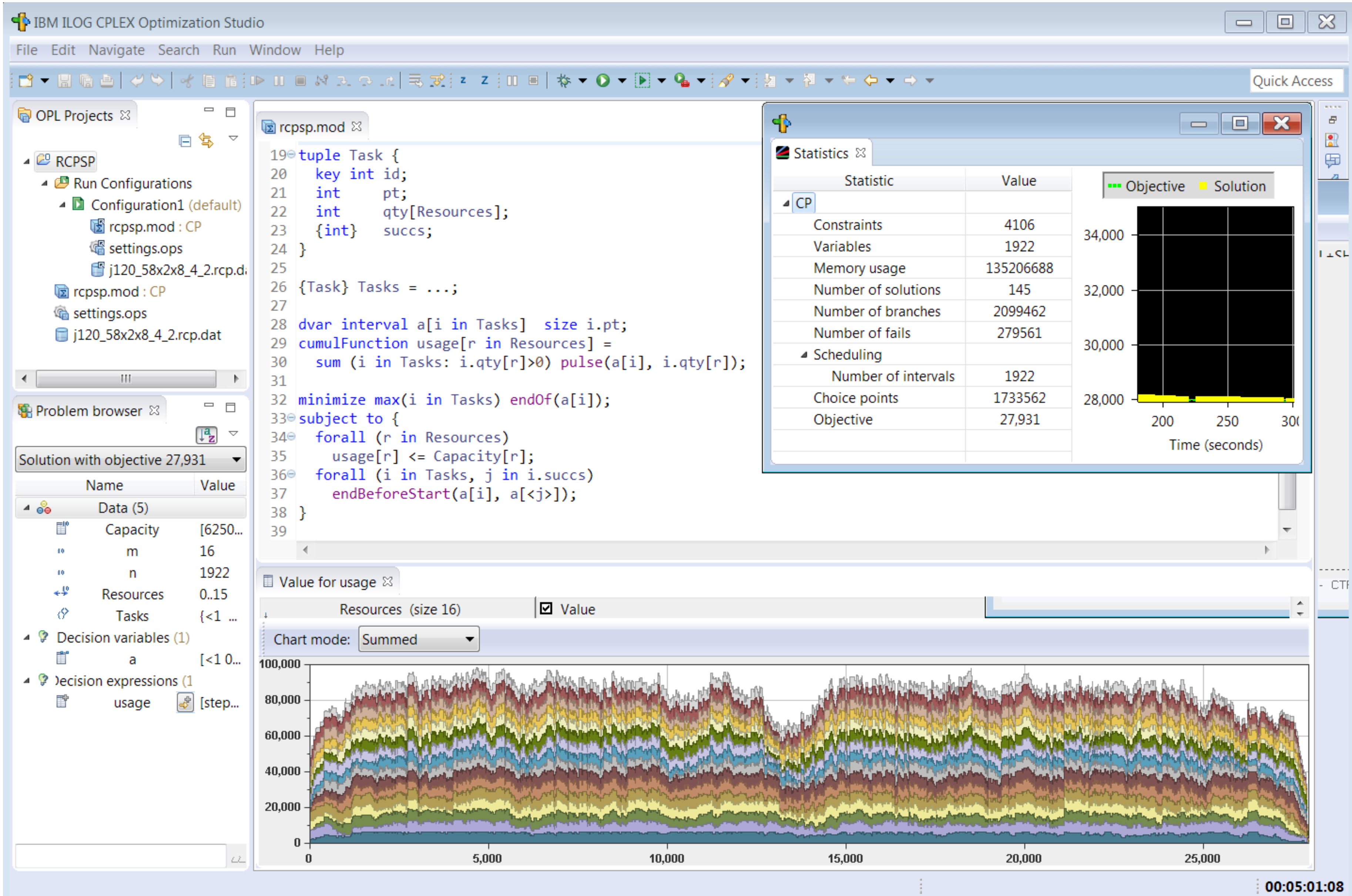
$$
\begin{array}{lll}
\min & \max_{i \in I} \ \mathrm{endOf}(a_i) & \\
& \mathrm{endBeforeStart}(a_i, a_j) & \forall (i,j) \in E \\
& \sum_{i \in I} \mathrm{pulse}(a_i, d_{ir}) \leq D_r & \forall r \in R \\
& a_i \text{ interval of size } p_i & \forall i \in I
\end{array}
$$

CP Optimizer model for RCPSP

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

A cumul **function** expression that can be constrained (here ≤ D_r)

CP Optimizer model for RCPSP

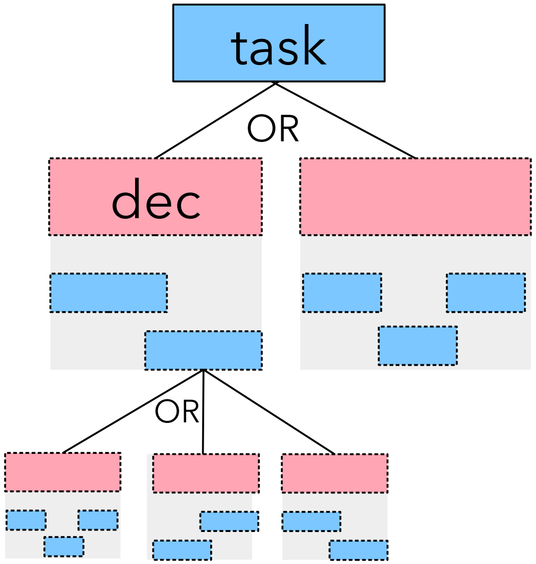


CP Optimizer modeling concepts

- 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
- Unlike MILP, in CP Optimizer the size of the model in general grows linearly with the size of the problem instance

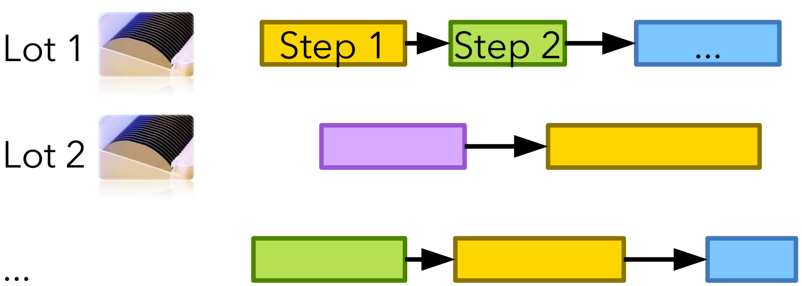
CP Optimizer model for Work-Breakdown Structures

```
using CP;
tuple Dec { int task; {int} subtasks; };
int n = ...;
int compulsory[1..n] = ...;
{Dec} Decs = ...;
int nbDecs[i in 1..n] = card( {d | d in Decs : d.task==i} );
int nbParents[i in 1..n] = card( {d | d in Decs : i in d.subtasks} );

dvar interval task[i in 1..n] optional;
dvar interval dec[d in Decs] optional;

constraints {
  forall(i in 1..n) {
    if (nbParents[i]==0 && 0<compulsory[i])
      presenceOf(task[i]);
    if (nbDecs[i]>0) {
      alternative(task[i], all(d in Decs: d.task==i) dec[d]);
      forall(d in Decs: d.task==i)
        span(dec[d], all(j in d.subtasks) task[j]);
    }
  }
  forall(d in Decs, j in d.subtasks: 0<compulsory[j])
    presenceOf(dec[d]) => presenceOf(task[j]);
}
```



Note the similarities with Hierarchical Task Network (HTN) in A.I. Planning

CP Optimizer model for semiconductor manufacturing



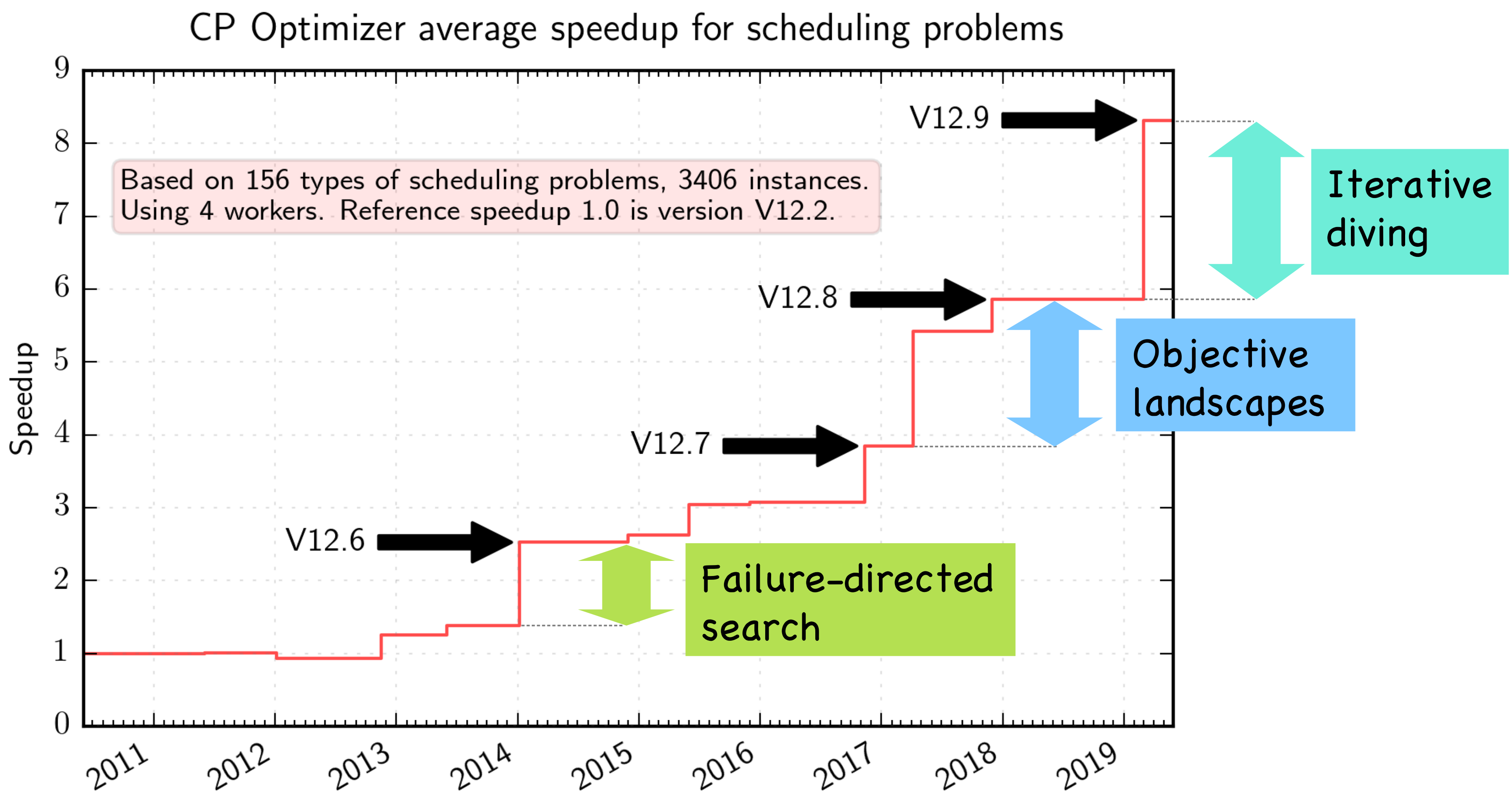
S. Knopp et al. Modeling Maximum Time Lags in Complex Job-Shops with Batching in Semiconductor Manufacturing. PMS 2016.

```
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]);
}
```

CP Optimizer automatic search - Performance

- Results published in CPAIOR-2015 (using V12.6)
  - Job-shop
    - 15 instances closed out of 48 open ones
  - Job-shop with operators
    - 208 instances closed out of 222 open ones
  - Flexible job-shop
    - 74 instances closed out of 107 open ones
  - RCPSP
    - 52 new lower bounds + 39 instances closed in 2019
  - RCPSP with maximum delays
    - 51 new lower bounds out of 58 small-medium instances
      + 372 bounds improved on large instances in 2019
  - Multi-mode RCPSP
    - 535 instances closed out of 552
  - Multi-mode RCPSP with maximum delays
    - All 85 open instances of the benchmark closed

Performance evolution



Under the hood

Artificial Intelligence **Operations Research**

Constraint propagation

Learning

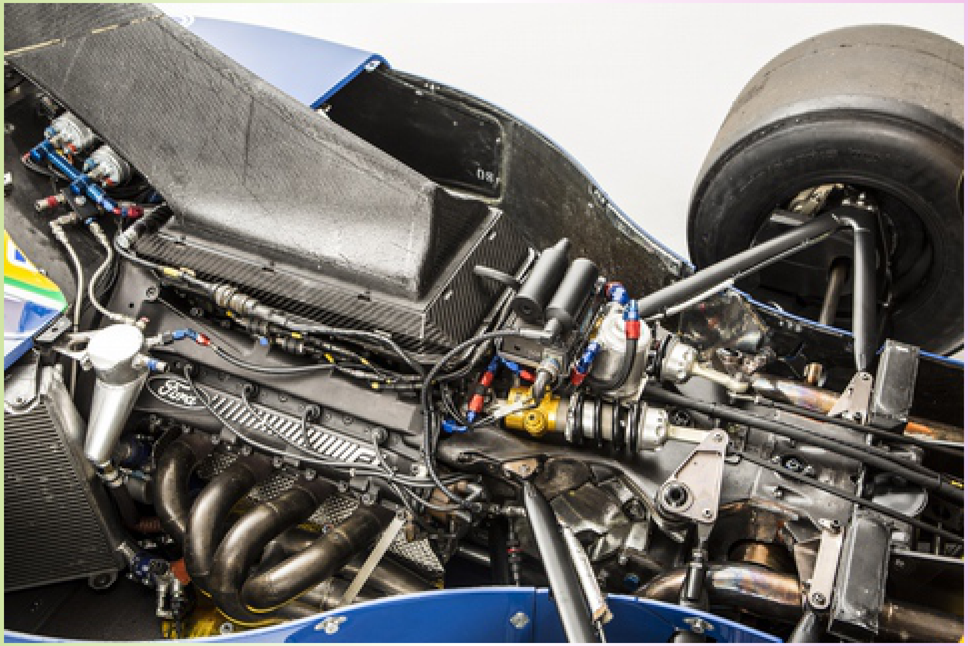
Temporal constraint networks

2-SAT networks

No-goods

Heuristics

Model presolve



Restarts

Tree search

LNS

Randomization

Linear relaxations

Problem specific scheduling algorithms

Conclusion

- CP Optimizer $\subset$ AI $\cup$ OR
- CP Optimizer $\neq$ CP
- CP Optimizer = Exact algorithm for optimality proofs + Fast heuristic for finding feasible solutions and optimizing them
- CP Optimizer ecosystem $\approx$ MILP ecosystem :
  Model & run paradigm
  Clean combinatorial optimization framework
  Language versatility: C++, Python, Java, C#, OPL
  Well documented improvements of automatic search
  I/O file format
  Modeling assistance
  Conflict refiner
  Warm-start

Most of CP Optimizer's ideas have been published !

Modeling concepts

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

Examples

- 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.
- P. Laborie. An Update on the Comparison of MIP, CP and Hybrid Approaches for Mixed Resource Allocation and Scheduling. In Proc. CPAIOR-2018.

Search algorithm

- P. Laborie, D. Godard. Self-Adapting Large Neighborhood Search: Application to Single-Mode Scheduling Problems. In: Proc. MISTA-2007.
- 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.
- P. Laborie, J. Rogerie. Temporal Linear Relaxation in IBM ILOG CP Optimizer. Journal of Scheduling, 19(4), 391–400 (2016).
- P. Laborie. Objective Landscapes for Constraint Programming. In Proc. CPAIOR-2018.

Overview

- P. Laborie, J. Rogerie, P. Shaw, P. Vilím. IBM ILOG CP Optimizer for Scheduling. Constraints Journal, 23(2), 210-250 (2018). http://ibm.biz/Constraints2018