Solving Industrial Scheduling Problems with Constraint Programming

Philippe Laborie
IBM Analytics, Decision Optimization

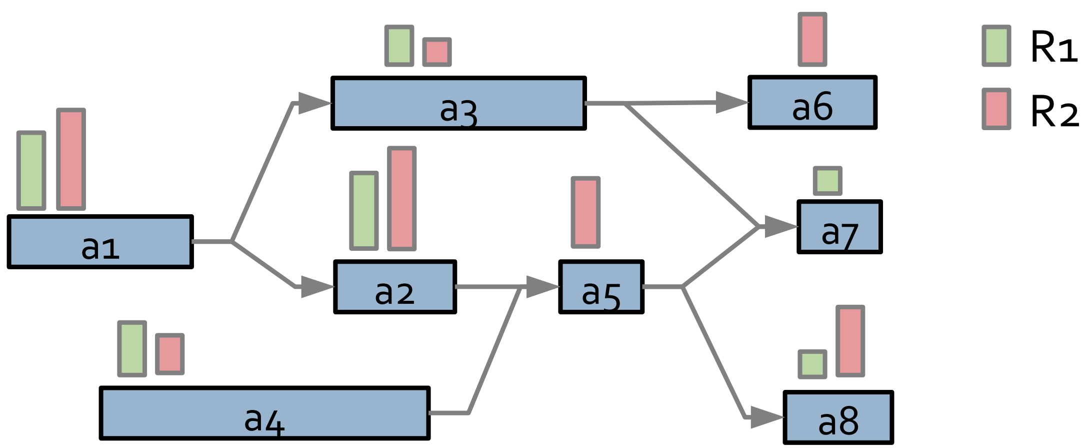
Sept. 23, 2015

Outline

- Scheduling problems: from theory to practice
- CP extensions for scheduling
- Simplifying model design & problem resolution

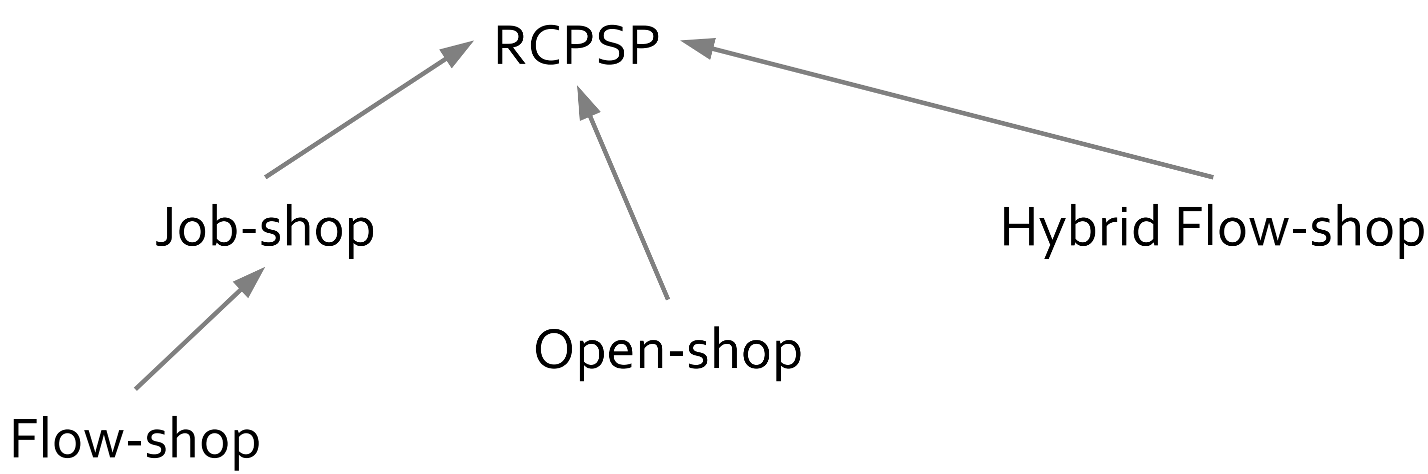
Scheduling problems: from **theory** to practice

- **Theory**: e.g. RCPSP (Resource Constrained Project Scheduling)
  - Problem definition:
    - A set of n tasks A1,…,An with specified duration Di for task Ai
    - A set of precedence constraints between tasks (precedence graph)
    - A set of m resources R1,…,Rm with specified capacity Cj for resource Rj
    - Each task Ai requires some resources in a given quantity
    - Objective is to minimize project makespan
  - Classical benchmark: PSPLib = 30-120 tasks



Scheduling problems: from **theory** to practice

- **Theory**: e.g. RCPSP (Resource Constrained Project Scheduling)
  – From an academical point of view, this is a very generic problem that subsumes many other classical scheduling problems



Scheduling problems: from theory to **practice**

- Some real scheduling applications
  - Aerospace: Project scheduling, Aircraft assembly, Assembly line configuration, Satellite communication & observation, Rover activities …
  - Energy & Utilities: Nuclear plant outage scheduling, Maintenance, Production planning …
  - Mining: Open pit mining …
  - Logistics, Supply Chain: Vehicle routing, Bikes and car sharing …
  - Manufacturing: Production scheduling, Assembly lines, Factory configuration, Test scheduling …
  - Media & Communications: Advertizing & program scheduling, Event & personnel scheduling …
  - Travel & transportation: Airport scheduling (gates, landing, parking, …), Port scheduling (quay cranes …), Train scheduling …
  - Health: Employee, Patient scheduling, Pharmaceutical products tests …
  - Agriculture: Crop scheduling (harvesting …)
  - General audience: Personal schedule organizer, Theme park planner …

Scheduling problems: from theory to **practice**

- **Practice**: the problem looks like …

this …

or this …

Scheduling problems: from theory to **practice**

- **Practice**: the problem is **large**
  - Typical problems are larger than 1000 activities
  - E.g. we solved a 1.000.000 tasks RCPSP-like problem for an aircraft manufacturer

Scheduling problems: from theory to **practice**

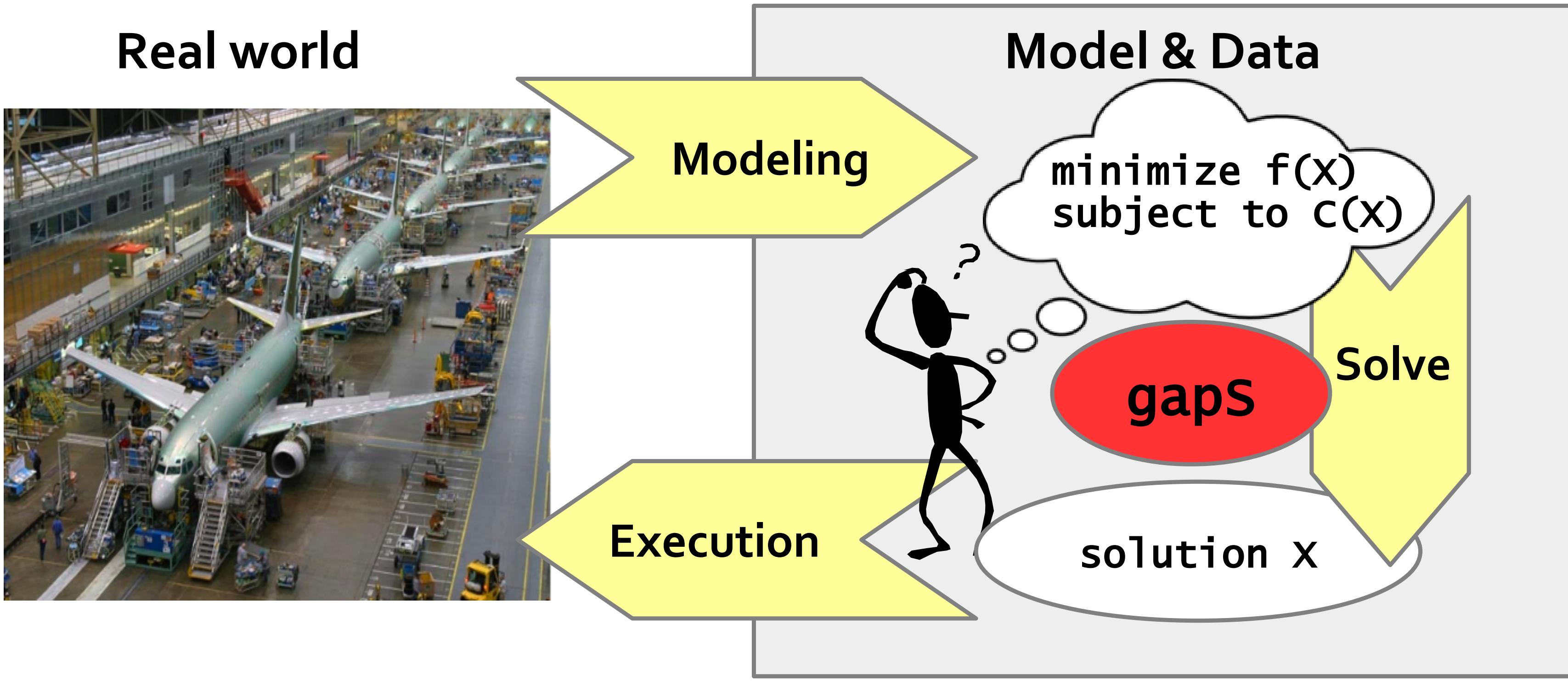
- **Practice**: the problem is **complex**
  - Heterogeneous types of **decisions**: start/end dates, resource allocation, mode selection, activity non-execution, activity durations, activity interruptions, resource quantities, producer/consumer matching, batch configuration, resource overconsumption, resource configuration, …
  - Complex **resources**: inventories, setup times and costs, calendars/shifts, moving resources, spatial constraints, synchronization constraints (e.g. conveyer belt), batch constraints
  - Complex **activities**: optional, interruptible, setups, maintenance, work breakdown structure
  - Complex **objective function**: earliness/tardiness, temporal preferences, resource related costs (many types!), most of the problems are multi-objectives

Scheduling problems: from theory to **practice**

- **Practice**: the problem is **not well defined**
  - Implicit constraints / objectives
  - Some aspects of the problem are critical, others can be:
    - Approximated
    - Relaxed
    - Over-constrained
    - Considered in a post-processing step
    - …
- **Practice**: the data is:
  - Hard to get (different data sources, confidentiality, …)
  - Uncertain / Imprecise / Incomplete / Incorrect
- Customers are expecting **good solutions** to **their problem**, they usually do not care about how the problem is modeled/solved

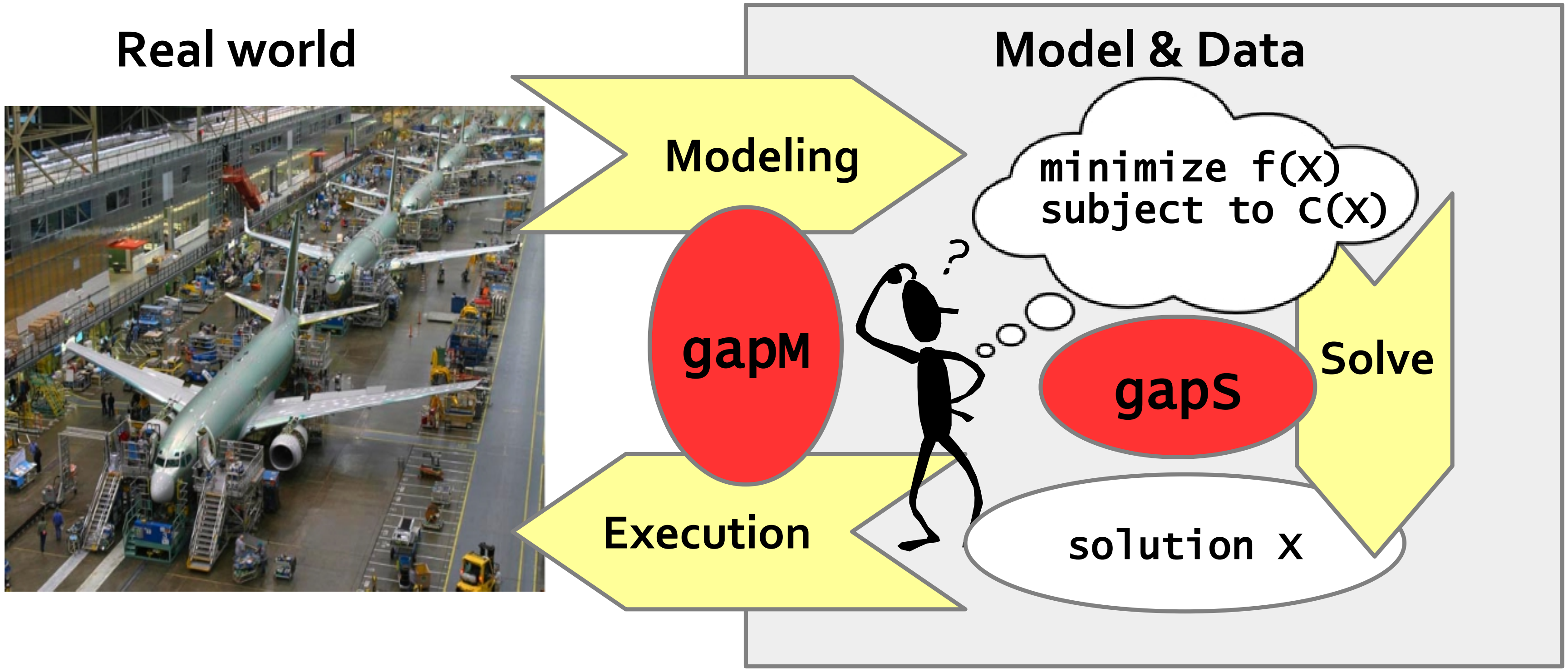
Scheduling problems: from theory to **practice**

- **Practice**: mind the **real gap**



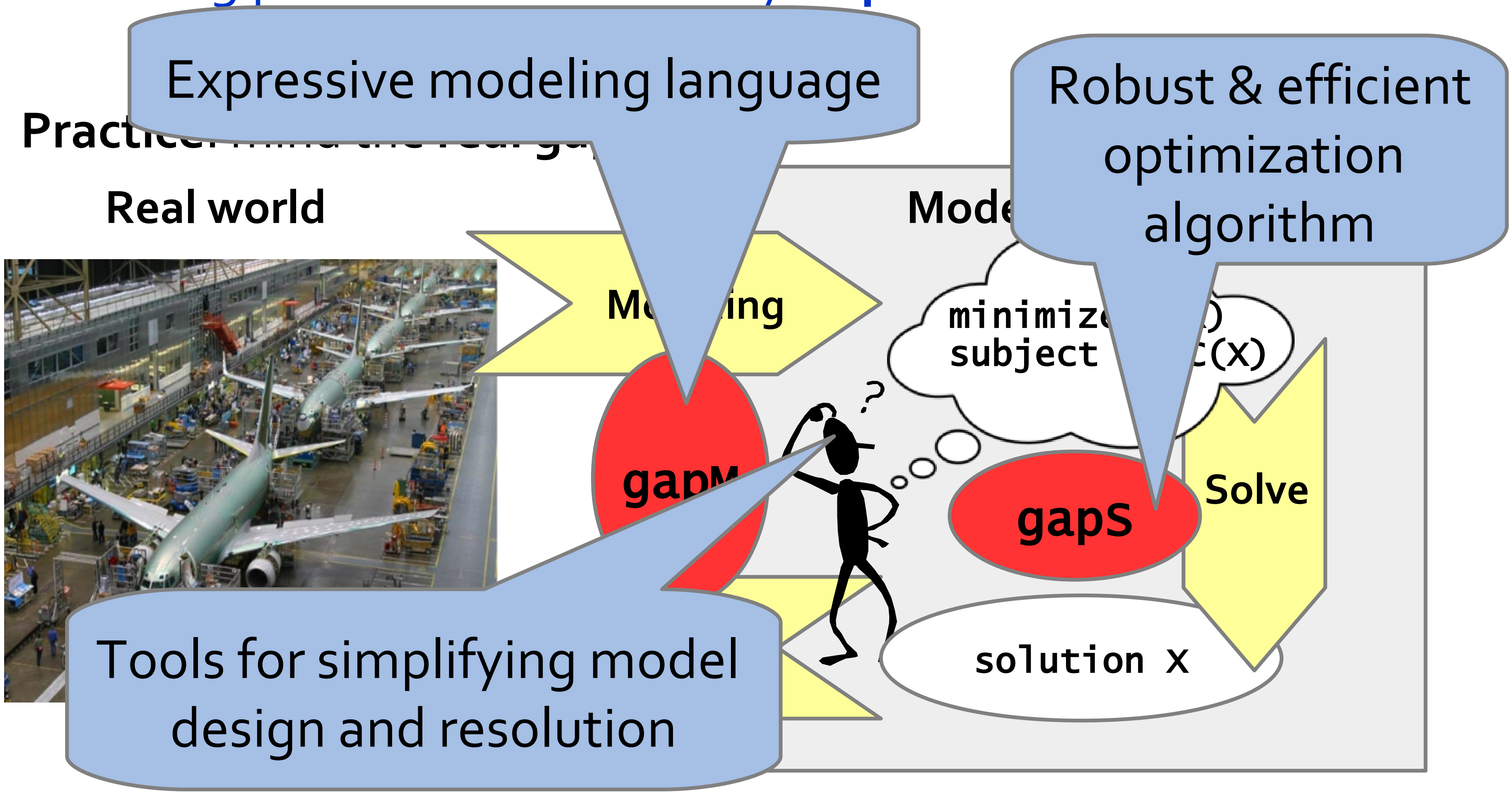
Scheduling problems: from theory to **practice**

- **Practice**: mind the **real gap**



  - “**Essentially, all models are wrong, but some are useful**” – G. Box
  - The real gap you must care about is **gapM** ⊕ gapS

Scheduling problems: from theory to **practice**



– “**Essentially, all models are wrong, but some are useful**” – G. Box

– The real gap you must care about is **gapM ⊕ gapS**

CP extensions for scheduling

- Extension of classical CSP with a new type of decision variable: **optional interval variable** :

$$\text{Domain}(x) \subseteq \{\bot\} \cup \{\, [s,e) \mid s,e \in Z,\ s \leq e \,\}$$

Absent interval Interval of integers

- Introduction of mathematical notions such as **sequences** and **functions** to capture temporal dimension of scheduling problems

CP extensions for scheduling

- In scheduling models, **interval variables** usually represent an interval of time during which some property hold (e.g. an activity executes) and whose end-points (start/end) are decision variables of the problem.
- Examples:
  - A sub-project in a project, a task in a sub-project (Work Breakdown Structure)
  - A batch of operations
  - The setup of a tool on a machine
  - The moving of an item by a transportation device
  - The utilization interval of a resource
- Idea of the model (and search) is to avoid the enumeration of start/end values (continuous time)

CP extensions for scheduling

- An interval variable can be **optional** meaning that it is a decision to have it present or absent in a solution.
- Examples:
  – Unperformed tasks and optional sub-projects
  – Alternative resources, modes or recipes for processing an order, each mode specifying a particular combination of operational resources
  – Operations that can be processed in different temporal modes (e.g. series or parallel)
  – Activities that can be performed in an alternative set of batches or shifts

CP extensions for scheduling

- Example: RCPSP

```
dvar interval a[i in Tasks] size i.pt;

cumulFunction usage[r in Resources] =
  sum (i in Tasks: i.qty[r]>0) pulse(a[i], i.qty[r]);

minimize max(i in Tasks) endOf(a[i]);
subject to {
  forall (r in Resources)
    usage[r] <= Capacity[r];
  forall (i in Tasks, j in i.succs)
    endBeforeStart(a[i], a[<j>]);
}
```

CP extensions for scheduling

- Example: Job-shop scheduling problem

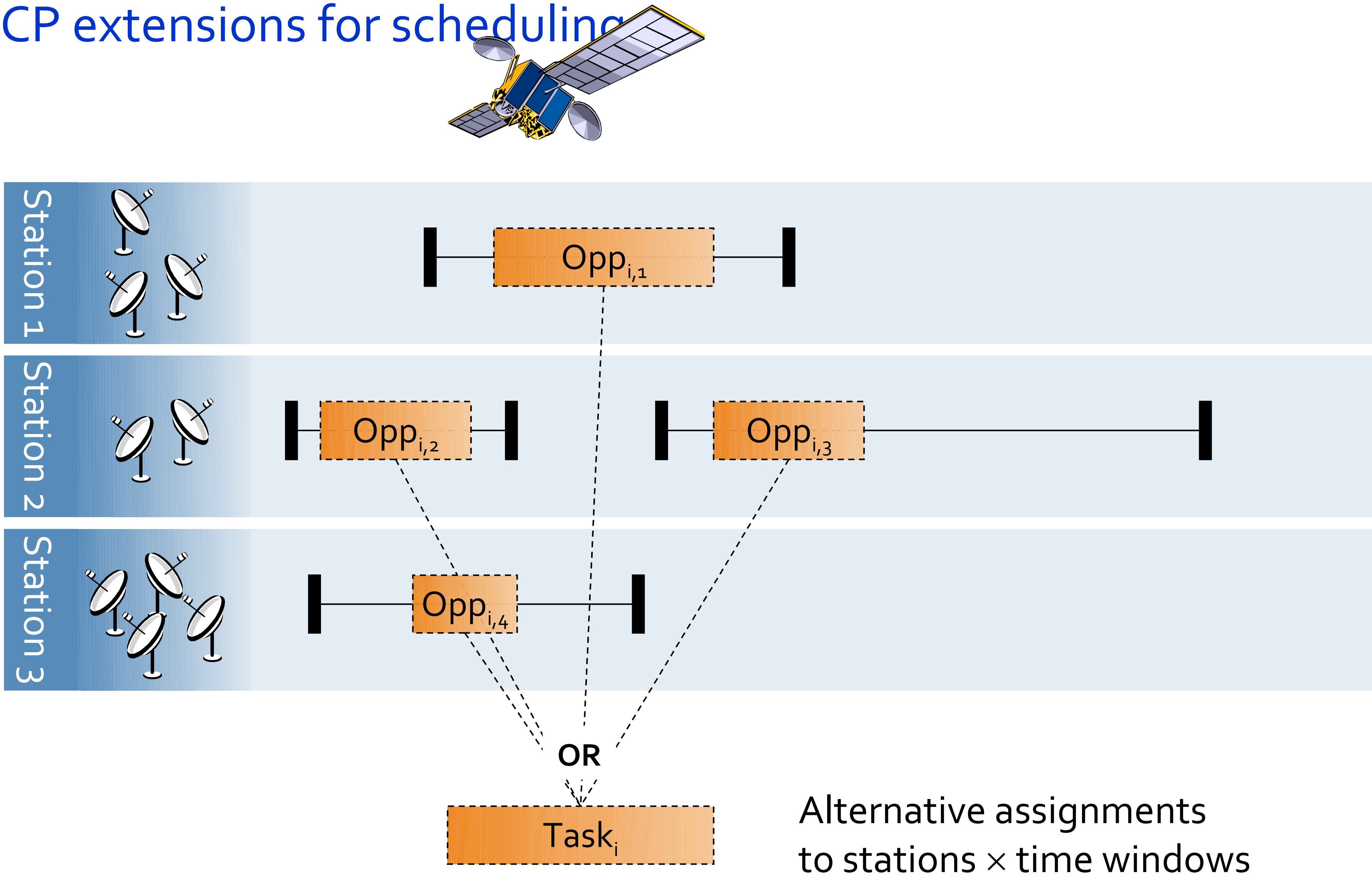
```
dvar interval op[j in Jobs][p in Pos] size Ops[j][p].pt;
dvar sequence mchs[m in Mchs] in
  all(j in Jobs, p in Pos : Ops[j][p].mch == m) op[j][p];

minimize max(j in Jobs) endOf(op[j][nbPos-1]);
subject to {
  forall (m in Mchs)
    noOverlap(mchs[m]);
  forall (j in Jobs, p in 1..nbPos-1)
    endBeforeStart(op[j][p-1], op[j][p]);
}
```

CP extensions for scheduling

- Satellite Control Network scheduling problem [1]
- n communication tasks for Earth orbiting satellites must be scheduled on a total of 32 antennas spread across 13 ground-based tracking stations
- In the instances, n ranges from to 400 to 1300
- Objective: maximize the number of scheduled tasks

[1] Kramer & al.: Understanding Performance Trade-offs in Algorithms for Solving Oversubscribed Scheduling.

CP extensions for scheduling



Alternative assignments to stations × time windows (opportunities)

CP extensions for scheduling

```
using CP;

tuple Station {
  string name; // Ground station name
  int id;      // Ground station identifier
  int cap;     // Number of available antennas
}

tuple Opportunity {
  string task; // Task
  int station; // Ground station
  int smin;    // Start of visibility window of opportunity
  int dur;     // Task duration in this opportunity
  int emax;    // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks] optional;
dvar interval opp[o in Opportunities] optional in o.smin..o.emax size o.dur;

maximize sum(t in Tasks) presenceOf(task[t]);
subject to {
  forall(t in Tasks)
    opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);
  forall(s in Stations)
    numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o],1) <= s.cap;
}
```


Automatic Search

- Search algorithm is **Complete**
- Core CP techniques used as a building block:
  - Tree search (Depth First)
  - Constraint propagation
- But also:
  - Deterministic multicore parallelism
  - Model presolve
  - Algorithms portfolios
  - Machine learning
  - Restarting techniques
  - Large Neighborhood Search
  - No-good learning
  - Impact-based branching
  - Opportunistic probing
  - Dominance rules
  - LP-assisted heuristics
  - Randomization
  - Evolutionary algorithms

Automatic Search

- Very good performance results on academical problems
  - CP-AI-OR 2015:

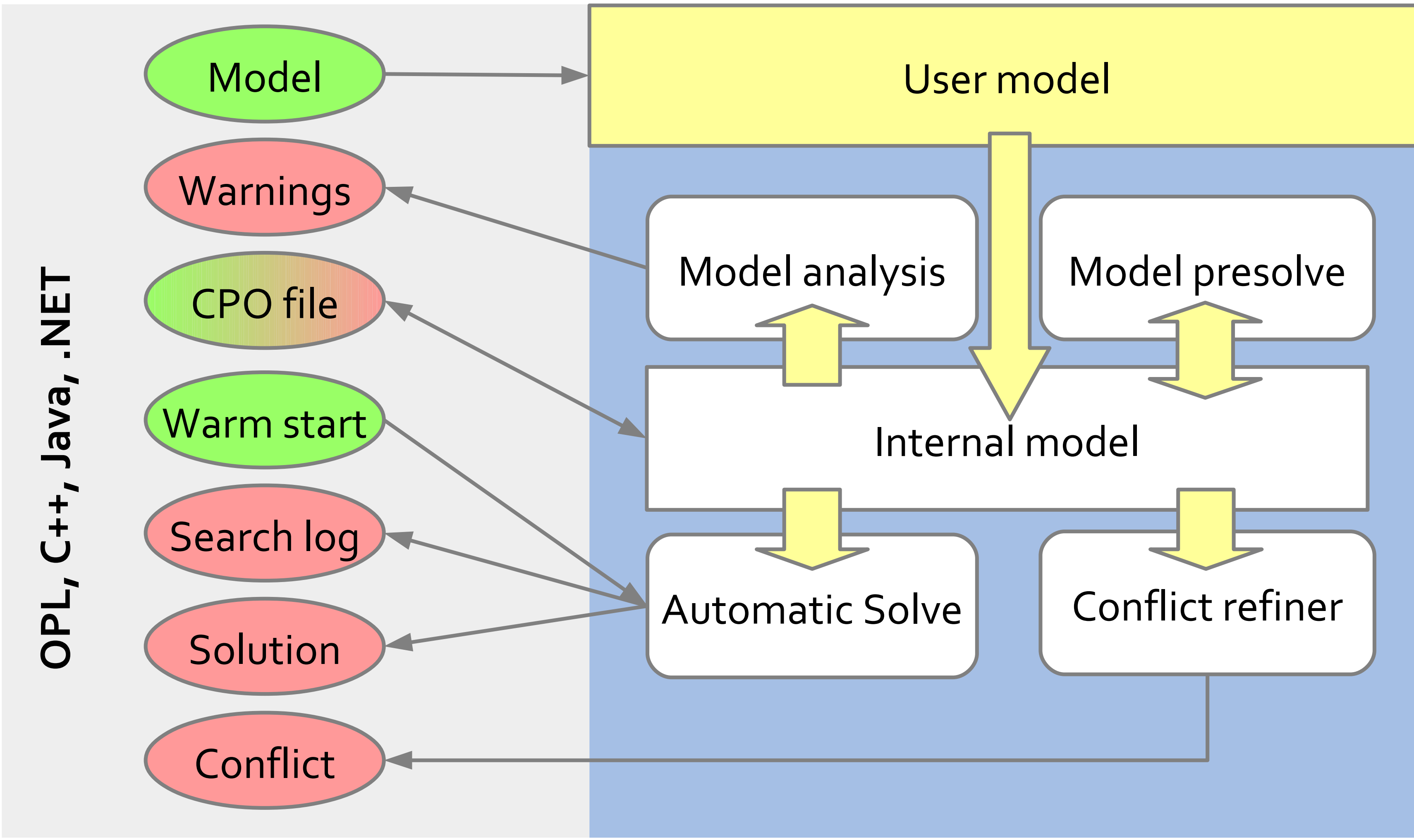
Benchmark set	Number of instances	Lower bound improvements	Upper bound improvements	Closed instances
JobShop	48	40	3	15
JobShopOperators	222	107	215	208
FlexibleJobShop	107	67	39	74
RCPSP	472	52	1	0
RCPSPMax	58	51	23	1
MultiModeRCPSP (j30)	552	No reference	3	535
MultiModeRCPSPMax	85	84	77	85

Table 1. Results summary

  - Winner of the CP-2015 Industrial Modelling Challenge with a very simple model: csplib.org/Problems/prob073/models/model.mod.html

- <u>More important</u>: very good performance on industrial problems

Simplifying model design & problem resolution



Tools: I/O format

- Objective:
  - Make it easier to understand the content of a model
  - Communicate a model to IBM support team regardless of the API used to build it (OPL, C++, Java, .NET)
- Structure of a .cpo file
  - Human readable
  - Flat (no cycle, forall statements)
  - No user defined data types
  - Internal information such as CPO version or platform used
  - Includes search parameter values
- Facilities
  - Export model before/instead of solve
  - Export model during solve (with current domains)
  - Import model instead of normal modeling

Tools: I/O format

```
// Interval-related variables:

"task(1)"  = intervalVar(optional);
"task(1A)" = intervalVar(optional);
…
"opp({1,1,62})"  = intervalVar(optional, start=62..intervalmax, end=0..99, size=25);
"opp({1A,1,32})" = intervalVar(optional, start=32..intervalmax, end=0..69, size=33);
…

// Objective:

maximize(sum([presenceOf("task(1)"), presenceOf("task(1A)"),  …]));
…
// Constraints:

alternative("task(1)", ["opp({1,1,62})"], 1);
…
pulse("opp({3,1,58})", 1) + pulse("opp({1,1,62})", 1) +  … <= 4;
…

parameters {
  LogVerbosity = Quiet;
}
```

Tools: model warnings

- Like a compiler, CP Optimizer can analyze the model and print some warnings
  - When there is something suspicious in the model
  - Regardless how the model was created (C++, OPL, ...)
  - Including guilty part of the model in the cpo file format
  - Including source code line numbers (if known)
  - 3 levels of warnings, more than 50 types of warnings

```
satellite.cpo:2995:1: Warning: Constraint 'alternative': there is only one alternative
interval variable.
                    alternative("task(1)", ["opp({1,1,62})"], 1)

satellite.cpo:2996:1: Warning: Constraint 'alternative': there is only one alternative
interval variable.
                    alternative("task(1A)", ["opp({1A,1,32})"], 1)
```

Tools: model presolve

- Objective: **automatically** reformulate the model in order to speed-up its resolution
- Works on an internal representation of the model
- Different types of presolve:
  - Aggregation of basic constraints into global constraints
  - Constraint strengthening
  - Simplifications and factorizations

Tools: model presolve

- Examples of presolve rules
  - Common sub-expression elimination
  - Aggregation of `x!=y` cliques as **`allDifferent`**`([x,y,...])`
  - Precedence strengthening
    - If a and b cannot overlap and **`startsBeforeStart`**`(a,b)`
    - Then **`endsBeforeStart`**`(a,b)`
  - Precedence recognition
    - If **`endOf`**`(a,-∞)` ≤ **`startOf`**`(b,+∞)`
    - Then **`endsBeforeStart`**`(a,b)`
    - Precedences are aggregated into a "time net" (STN) for faster and stronger propagation
  - 2-SAT clauses recognition
    - **`presenceOf`**`(a)` ≤ **`presenceOf`**`(b)`
    - Such clauses are aggregated into a "logical net" for stronger propagation

Tools: search log

- Objective: understand what happens during the automatic search

```
! ----------------------------------------------------------------------
! Maximization problem - 2980 variables, 853 constraints
! Workers              = 2
! TimeLimit            = 30
! Initial process time : 0.01s (0.00s extraction + 0.01s propagation)
!  . Log search space  : 4627.3 (before), 4627.3 (after)
!  . Memory usage      : 16.9 MB (before), 19.7 MB (after)
! Using parallel search with 2 workers.
! ----------------------------------------------------------------------
!          Best Branches  Non-fixed    W       Branch decision
*           746     3945 0.79s         1              -
            746     4000       2924    1       on task("8")
            746     4000       2908    2       on opp({"186",2,66})
 …
! Time = 1.37s, Explored branches = 35832, Memory usage = 55.5 MB
!          Best Branches  Non-fixed    W       Branch decision
            818    12000       2920    1       on task("184")
 …
! ----------------------------------------------------------------------
! Search terminated by limit, 6 solutions found.
! Best objective          : 826
! Number of branches      : 709092
! Number of fails         : 179648
! Total memory usage      : 54.5 MB (52.9 MB CP Optimizer + 1.6 MB Concert)
! Time spent in solve     : 30.03s (30.01s engine + 0.01s extraction)
! Search speed (br. / s)  : 23625.4
! ----------------------------------------------------------------------
```

Problem characteristics

Tools: search log

- Objective: understand what happens during the automatic search

```
! ----------------------------------------------------------------------------
! Maximization problem - 2980 variables, 853 constraints
! Workers              = 2
! TimeLimit            = 30
! Initial process time : 0.01s (0.00s extraction + 0.01s propagation)
!  . Log search space  : 4627.3 (before), 4627.3 (after)
!  . Memory usage      : 16.9 MB (before), 19.7 MB (after)
! Using parallel search with 2 workers.
! ----------------------------------------------------------------------------
!          Best Branches  Non-fixed    W       Branch decision
*           746     3945 0.79s         1          -
            746     4000        2924   1       on task("8")
            746     4000        2908   2       on opp({"186",2,66})
 …
! Time = 1.37s, Explored branches = 35832, Memory usage = 55.5 MB
!          Best Branches  Non-fixed    W       Branch decision
            818    12000        2920   1       on task("184")
 …
! ----------------------------------------------------------------------------
! Search terminated by limit, 6 solutions found.
! Best objective         : 826
! Number of branches     : 709092
! Number of fails        : 179648
! Total memory usage     : 54.5 MB (52.9 MB CP Optimizer + 1.6 MB Concert)
! Time spent in solve    : 30.03s (30.01s engine + 0.01s extraction)
! Search speed (br. / s) : 23625.4
! ----------------------------------------------------------------------------
```

Modified parameter values

Tools: search log

- Objective: understand what happens during the automatic search

```
! ----------------------------------------------------------------------------
! Maximization problem - 2980 variables, 853 constraints
! Workers              = 2
! TimeLimit            = 30
! Initial process time : 0.01s (0.00s extraction + 0.01s propagation)
!  . Log search space  : 4627.3 (before), 4627.3 (after)
!  . Memory usage      : 16.9 MB (before), 19.7 MB (after)
! Using parallel search with 2 workers.
! ----------------------------------------------------------------------------
!         Best Branches  Non-fixed    W       Branch decision
*          746      3945 0.79s        1              -
           746      4000       2924   1       on task("8")
           746      4000       2908   2       on opp({"186",2,66})
 …
! Time = 1.37s, Explored branches = 35832, Memory usage = 55.5 MB
!         Best Branches  Non-fixed    W       Branch decision
           818     12000       2920   1       on task("184")
 …
! ----------------------------------------------------------------------------
! Search terminated by limit, 6 solutions found.
! Best objective         : 826
! Number of branches     : 709092
! Number of fails        : 179648
! Total memory usage     : 54.5 MB (52.9 MB CP Optimizer + 1.6 MB Concert)
! Time spent in solve    : 30.03s (30.01s engine + 0.01s extraction)
! Search speed (br. / s) : 23625.4
! ----------------------------------------------------------------------------
```

Root node information

Tools: search log

- Objective: understand what happens during the automatic search

```
! ----------------------------------------------------------------------------
! Maximization problem - 2980 variables, 853 constraints
! Workers              = 2
! TimeLimit            = 30
! Initial process time : 0.01s (0.00s extraction + 0.01s propagation)
!  . Log search space  : 4627.3 (before), 4627.3 (after)
!  . Memory usage      : 16.9 MB (before), 19.7 MB (after)
! Using parallel search with 2 workers.
! ----------------------------------------------------------------------------
!          Best Branches  Non-fixed    W       Branch decision
*           746     3945 0.79s         1              -
            746     4000       2924    1       on task("8")
            746     4000       2908    2       on opp({"186",2,66})
 …
! Time = 1.37s, Explored branches = 35832, Memory usage = 55.5 MB
!          Best Branches  Non-fixed    W       Branch decision
            818    12000       2920    1       on task("184")
 …
! ----------------------------------------------------------------------------
! Search terminated by limit, 6 solutions found.
! Best objective         : 826
! Number of branches     : 709092
! Number of fails        : 179648
! Total memory usage     : 54.5 MB (52.9 MB CP Optimizer + 1.6 MB Concert)
! Time spent in solve    : 30.03s (30.01s engine + 0.01s extraction)
! Search speed (br. / s) : 23625.4
! ----------------------------------------------------------------------------
```

New incumbent solutions (time, worker)

Tools: search log

- Objective: understand what happens during the automatic search

```
! ----------------------------------------------------------------------------
! Maximization problem - 2980 variables, 853 constraints
! Workers              = 2
! TimeLimit            = 30
! Initial process time : 0.01s (0.00s extraction + 0.01s propagation)
!  . Log search space  : 4627.3 (before), 4627.3 (after)
!  . Memory usage      : 16.9 MB (before), 19.7 MB (after)
! Using parallel search with 2 workers.
! ----------------------------------------------------------------------------
!          Best Branches  Non-fixed    W       Branch decision
*           746     3945 0.79s         1             -
            746     4000       2924    1       on task("8")
            746     4000       2908    2       on opp({"186",2,66})
 …
! Time = 1.37s, Explored branches = 35832, Memory usage = 55.5 MB
!          Best Branches  Non-fixed    W       Branch decision
            818    12000       2920    1       on task("184")
 …
! ----------------------------------------------------------------------------
! Search terminated by limit, 6 solutions found.
! Best objective         : 826
! Number of branches     : 709092
! Number of fails        : 179648
! Total memory usage     : 54.5 MB (52.9 MB CP Optimizer + 1.6 MB Concert)
! Time spent in solve    : 30.03s (30.01s engine + 0.01s extraction)
! Search speed (br. / s) : 23625.4
! ----------------------------------------------------------------------------
```

Periodical log
with fail information,
number of unfixed
variables, current decision

Tools: search log

- Objective: understand what happens during the automatic search

```
! ----------------------------------------------------------------------------
! Maximization problem - 2980 variables, 853 constraints
! Workers              = 2
! TimeLimit            = 30
! Initial process time : 0.01s (0.00s extraction + 0.01s propagation)
!  . Log search space  : 4627.3 (before), 4627.3 (after)
!  . Memory usage      : 16.9 MB (before), 19.7 MB (after)
! Using parallel search with 2 workers.
! ----------------------------------------------------------------------------
!          Best Branches  Non-fixed    W       Branch decision
*           746     3945 0.79s         1              -
            746     4000        2924    1       on task("8")
            746     4000        2908    2       on opp({"186",2,66})
 …
! Time = 1.37s, Explored branches = 35832, Memory usage = 55.5 MB
!          Best Branches  Non-fixed    W       Branch decision
            818    12000        2920    1       on task("184")
 …
! ----------------------------------------------------------------------------
! Search terminated by limit, 6 solutions found.
! Best objective         : 826
! Number of branches     : 709092
! Number of fails        : 179648
! Total memory usage     : 54.5 MB (52.9 MB CP Optimizer + 1.6 MB Concert)
! Time spent in solve    : 30.03s (30.01s engine + 0.01s extraction)
! Search speed (br. / s) : 23625.4
! ----------------------------------------------------------------------------
```

Final information with solution status and search statistics

Tools: warm start

- Objective: Start search from a known (possibly incomplete) solution given by the user (warm start) in order to further improve it or to help to guide the engine towards a first feasible solution
- API: `IloCP::setStartingPoint(IloSolution warmstart)`
- Use cases:
  - Restart an **interrupted search** with the current incumbent
  - Start from an initial solution found by an available **heuristic**
  - Goal programming for **multi-objective** problems
  - When finding an initial solution is hard, solve an initial problem that **maximizes constraint satisfaction** and start from its solution
  - Successively solving **similar** problems (e.g. dynamic scheduling)
  - **Hierarchical** problem solving (e.g. planning → scheduling)

Tools: conflict refiner

- Objective: identify a reason for an inconsistency by providing a **minimal infeasible subset** of constraints for an infeasible model
- Use cases:
  - **Model debugging** (errors in model)
  - **Data debugging** (inconsistent data)
  - The model and data are correct, but the associated data represents a **real-world conflict** in the system being modeled
  - You create an infeasible model to test properties of (or extract information about) a similar model

Tools: conflict refiner

```
using CP;

tuple Station {
  string name; // Ground station name
  int id;      // Ground station identifier
  int cap;     // Number of available antennas
}

tuple Opportunity {
  string task; // Task
  int station; // Ground station
  int smin;    // Start of visibility window of opportunity
  int dur;     // Task duration in this opportunity
  int emax;    // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks];
dvar interval opp[o in Opportunities] optional in o.smin..o.emax size o.dur;


subject to {
  forall(t in Tasks)
    opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);
  forall(s in Stations)
    numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o],1) <= s.cap;
}
```

Tools: conflict refiner

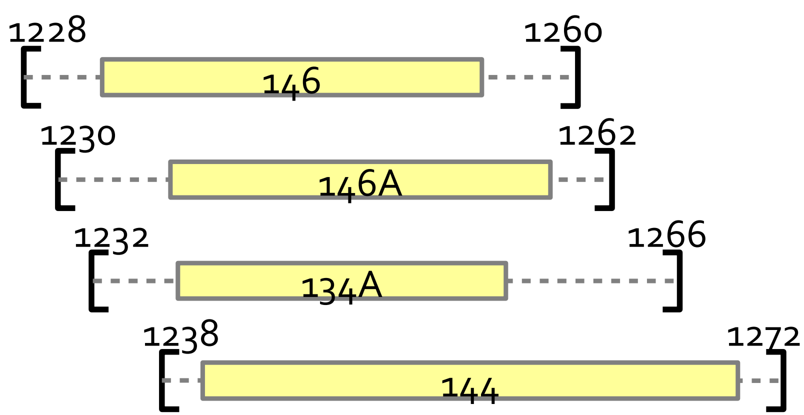
```
!-----------------------------------------------------------------------------
! Satisfiability problem - 2,980 variables, 851 constraints
! Problem found infeasible at the root node
! ----------------------------------------------------------------------------
  ...
! ----------------------------------------------------------------------------
! Conflict refining - 851 constraints
! ----------------------------------------------------------------------------
!   Iteration      Number of constraints
*          1                         851
*          2                         426
 ...
*         58                           5
*         59                           5
! Conflict refining terminated
! ----------------------------------------------------------------------------
! Conflict status           : Terminated normally, conflict found
! Conflict size             : 5 constraints
! Number of iterations      : 59
! Total memory usage        : 13.3 MB
! Conflict computation time : 0.51s
! ----------------------------------------------------------------------------
```

Tools: conflict refiner

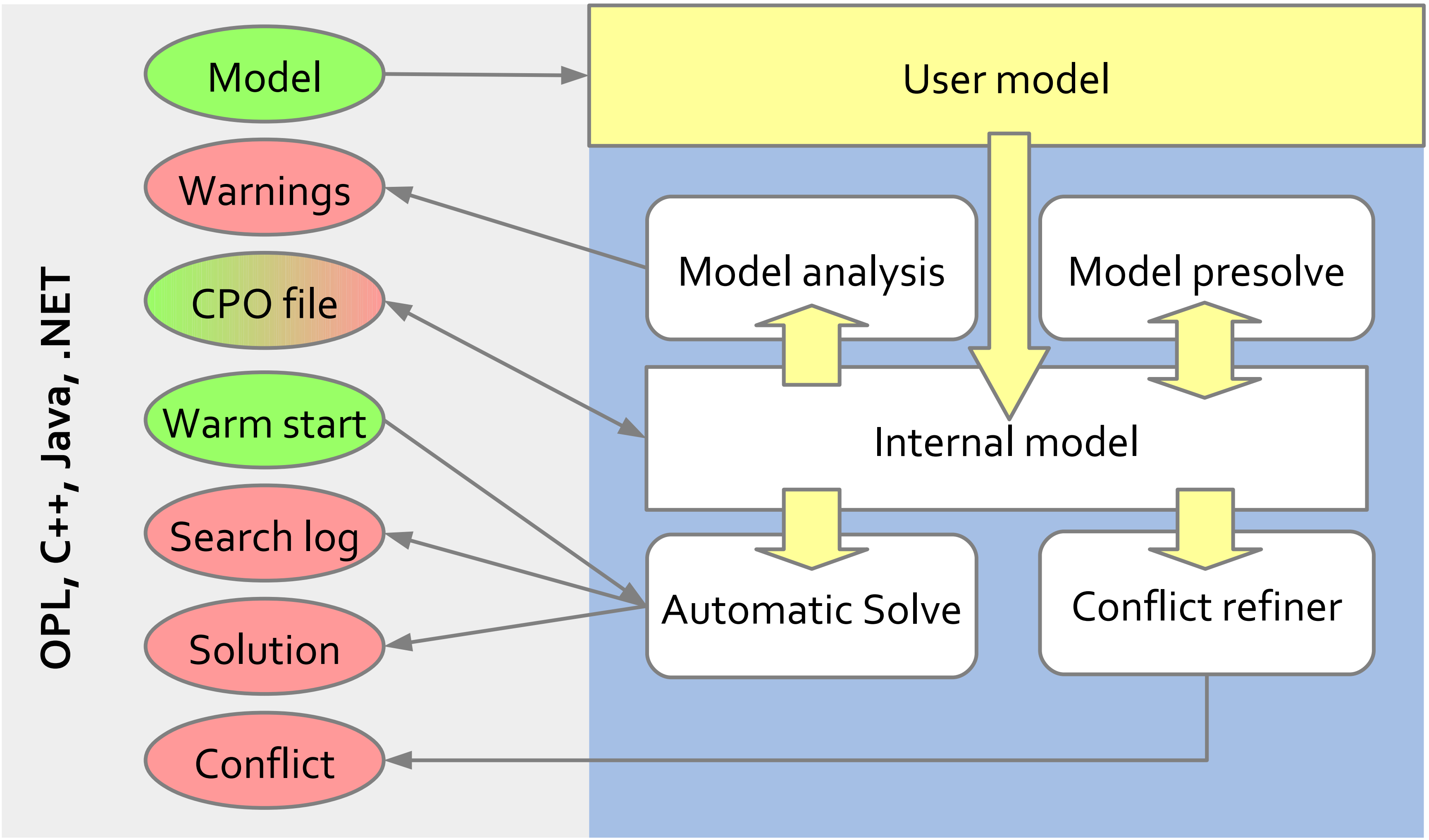
- Conflict:

Line	In conflict	Element (5)
26	Yes	opportunitySelection["134A"]
26	Yes	opportunitySelection["144"]
26	Yes	opportunitySelection["146"]
26	Yes	opportunitySelection["146A"]
28	Yes	numberOfAntennas[<"LION",6,3>]

- There is not enough antennas to accommodate all 4 tasks on their time-window on ground station “LION” (3 antennas):
  - <134A,6,1232,19,1266>
  - <144, 6,1238,31,1272>
  - <146, 6,1228,22,1260>
  - <146A,6,1230,22,1262>



Simplifying model design & problem resolution



From mathematical tools to real applications