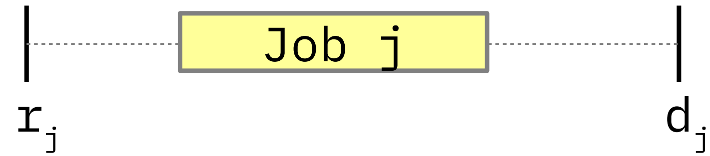
An Update on the Comparison of **MIP, CP** and **Hybrid Approaches** for **Mixed Resource Allocation** and **Scheduling**

Philippe Laborie, IBM
laborie@fr.ibm.com

The Problem

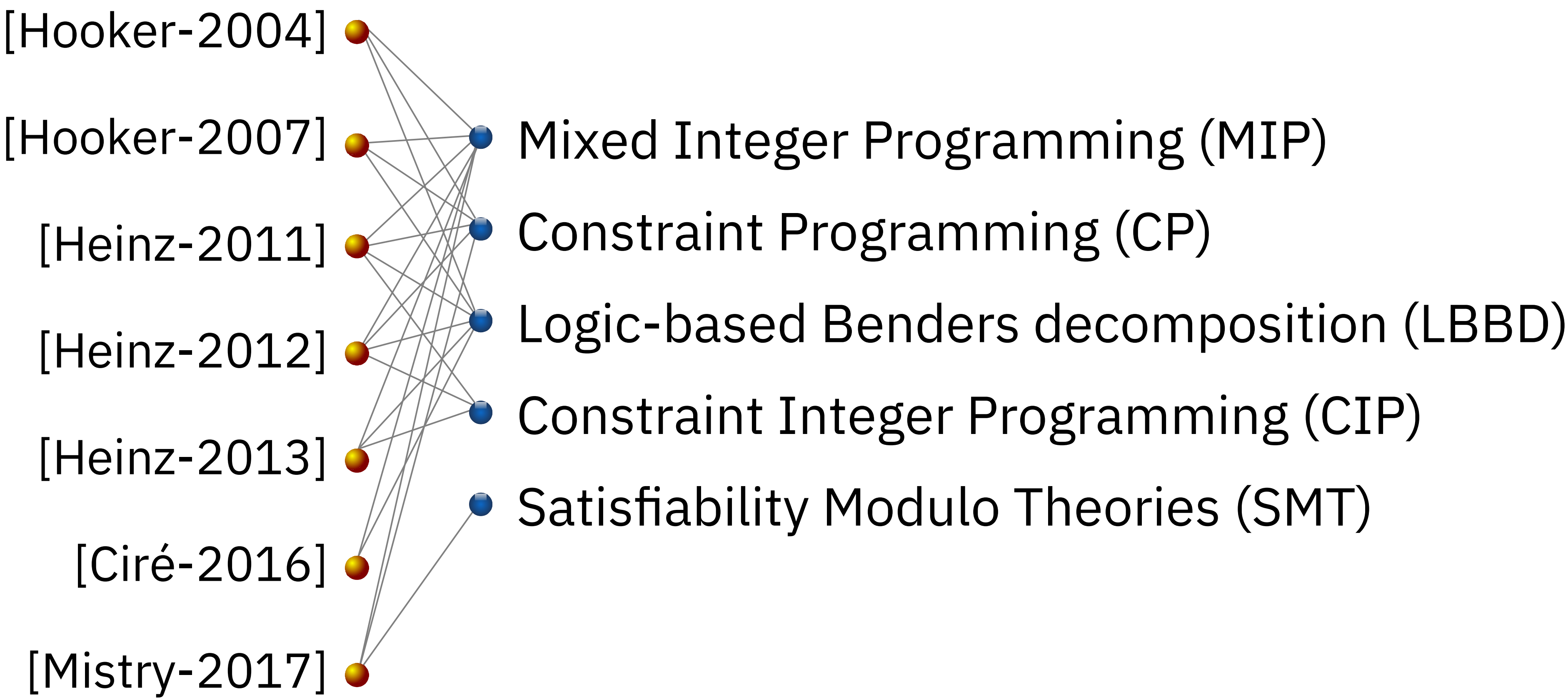
- Described in [1] where a benchmark is provided
- n jobs, for each job j:
  - Release date r_j, Due date d_j



- m facilities where the jobs can be executed
  For each facility i: maximal capacity C_i
- If a given job j is allocated to facility i:
  - Processing time of job j is p_{ij}
  - Job j requires c_{ij} units of facility i
  - Execution cost of job j is f_{ij}
- Decision variables: facility allocation & start time of jobs
- Minimize total job execution cost

[1] J. Hooker. *A Hybrid Method for Planning and Scheduling*. Proc. CP 2004.

The state-of-the-art



- The problem is very difficult. Even some problems with only 30 jobs and 2 facilities are still open !

CP Optimizer

- A component of IBM ILOG CPLEX Optimization Studio
  (so if you have CPLEX, you also have CP Optimizer)
- An optimization engine based on a **Model & Run** approach
  (like CPLEX)
- ... with a particular focus on **scheduling problems**

CP Optimizer: $\boxed{\textbf{Model}}$ & Run

- **Declarative** mathematical model of the problem
- Introduction of adequate mathematical concepts for scheduling problems (intervals, functions, permutations/sequences)
  - Modeling is easy
  - Modeling is fast
  - Models are compact and maintainable
  - Models generally scale well (size grows linearly with size of data)
- Uses classical ingredients of combinatorial optimization: variables, constraints, expressions, objective function

CP Optimizer: Model & $\boxed{\textbf{Run}}$

- Resolution is performed by an **automated search** algorithm that is:
  - Complete
  - Deterministic
  - Anytime
  - Efficient
  - Robust
  - Continuously improving

A complete CP Optimizer model for our problem

```
int n = ...; range J = 1..n; // Number of jobs
int m = ...; range I = 1..m; // Number of facilities
int r[J] = ...; // Release date of job j
int d[J] = ...; // Due date of job j
int C[I] = ...; // Capacity of facility i
int p[I][J] = ...; // Processing time of job j on facility i
int c[I][J] = ...; // Requirement of job j on facility i
int f[I][J] = ...; // Allocation cost of job j on facility i

dvar interval x[j in J] in r[j]..d[j];
dvar interval y[i in I][j in J] optional size p[i][j];

minimize sum(i in I, j in J) (f[i][j] * presenceOf(y[i][j]));
subject to {
  forall(j in J) { alternative(x[j], all(i in I) y[i][j]); }
  forall(i in I) { sum(j in J) pulse(y[i][j], c[i][j]) <= C[i]; }
}
```


Comparison with state-of-the-art approaches

- The tiny CP Optimizer model of previous slide closes the benchmark

Number of solved instances (out of 5)

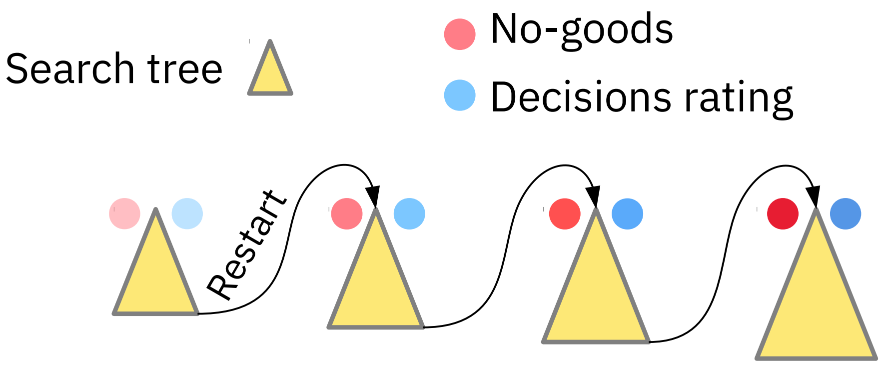
Average solve time (in s.)

		MIP		LBBD-CP		LBBD-SMT		CIP-CP		CPO (fds)	
#I	#J	opt	geom	opt	geom	opt	geom	opt	geom	opt	geom
2	16	5	8.0	5	**1.0**	5	2.82	5	4.7	5	**1.00**
	18	5	16.9	5	**1.3**	5	1.64	5	1.7	5	1.60
	20	5	29.0	5	3.7	5	1.47	5	**1.5**	5	1.62
	22	4	*812.4*	5	51.4	5	72.06	3	*382.5*	5	**4.06**
	24	3	*883.0*	4	*214.8*	5	196.72	2	*573.4*	5	**6.54**
	26	4	*1069.2*	5	209.0	3	*554.03*	4	*464.9*	5	**11.28**
	28	4	*378.9*	5	536.5	4	*38.58*	4	*42.0*	5	**17.00**
	30	3	*861.2*	3	*401.2*	1	*1147.84*	2	*587.6*	5	**92.30**
	32	3	*792.1*	0	-	3	*332.85*	2	*1140.5*	5	**120.14**
	34	3	*879.7*	2	*1745.1*	3	*509.45*	1	*1995.3*	3	***253.09***
	36	2	*1534.1*	1	*4770.2*	2	*450.68*	3	*548.4*	3	***491.11***
	38	2	*4980.2*	1	*5848.7*	4	*428.51*	2	*1334.0*	4	***127.07***
3	18	5	46.0	5	5.8	5	2.43	5	4.8	5	**1.56**
	20	4	*98.5*	5	1.5	5	**1.33**	5	6.9	5	1.75
	22	4	*554.6*	5	2.3	5	**2.17**	5	6.6	5	2.90
	24	5	304.5	5	6.7	5	9.41	5	78.6	5	**6.24**
	26	3	*1652.8*	5	19.8	5	44.50	5	40.2	5	**10.28**
	28	3	*987.6*	5	35.4	5	70.54	3	*194.9*	5	**15.13**
	30	3	*3100.2*	4	*178.3*	3	*540.18*	4	*520.9*	5	**54.17**
	32	2	*3601.3*	4	*1951.8*	2	*665.42*	3	*559.0*	5	**117.26**
4	20	5	25.3	5	1.8	5	1.15	5	4.3	5	**1.09**
	22	5	60.0	5	3.7	5	2.48	5	15.0	5	**2.29**
	24	4	*1399.0*	5	12.1	5	19.22	5	42.9	5	**4.95**
	26	3	*2787.8*	5	14.9	5	17.12	5	112.7	5	**12.44**
	28	3	*2124.2*	5	**9.6**	5	29.15	5	200.0	5	10.32
	30	2	*3253.6*	5	**31.7**	5	110.23	5	581.1	5	53.10
	32	1	*4691.0*	5	118.3	5	450.84	5	1519.1	5	**44.09**

The 5 remaining instances were closed by increasing the time-limit (up to 160h for the hardest one)

Under the Hood

- Search interleaves two complementary approaches:
  - Large Neighborhood Search (**LNS**) finds good solutions *C* [3]
  - Failure-Directed Search (**FDS**) tries proving infeasibility of *obj<C* [4]
    - Strong constraint propagation algorithms on **optional** intervals:
      - Alternative: constructive disjunction of interval domains [1]
      - Cumul functions: Edge-Finding variants [2]
    - Restarts, no-goods learning, dynamic rating of decisions



[1] P. Laborie, J. Rogerie. *Reasoning with Conditional Time-Intervals*. Proc. FLAIRS-2008, p555-560.
[2] P. Vilím: *Timetable Edge Finding Filtering Algorithm for Discrete Cumulative Resources*. CPAIOR-2011.
[3] P. Laborie, D. Godard. *Self-Adapting Large Neighborhood Search: Application to Single-Mode Scheduling Problems*. Proc. MISTA-2007.
[4] P. Vilím, P. Laborie, P. Shaw. *Failure-Directed Search for Constraint-Based Scheduling*. CPAIOR-2015.

Benchmark Extension

Given that the current benchmark is closed we generated more challenging instances (http://ibm.biz/AllocSched)

- Larger problems, finer time and capacity granularity

	Current benchmark	New benchmark
Size: # jobs # facilities	[10,50] [2,10]	[20,1000] [2,20]
Job duration	[5,30]	[100,4500]
Facility capacity Facility requirement	10 [1,10]	1000 [1,1000]
Facility requirements depends on task	No	Yes

More about CP Optimizer

- Recent overview of CP Optimizer for scheduling [1]

[1] P. Laborie, J. Rogerie, P. Shaw, P. Vilím. *IBM ILOG CP Optimizer for Scheduling.* Constraints Journal. April 2018, Volume 23, Issue 2, pp 210–250.

- Don’t miss tomorrow’s plenary talk by Paul Shaw:

	Friday June 29
9:00-10:00	**Invited talk**
	Paul Shaw: Ten Years of CP Optimizer