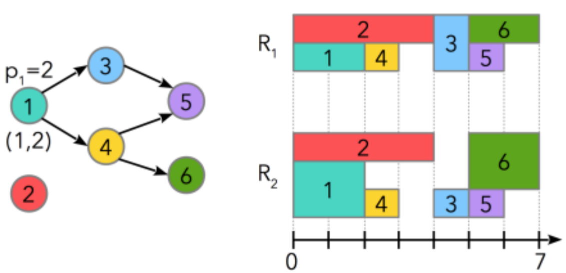
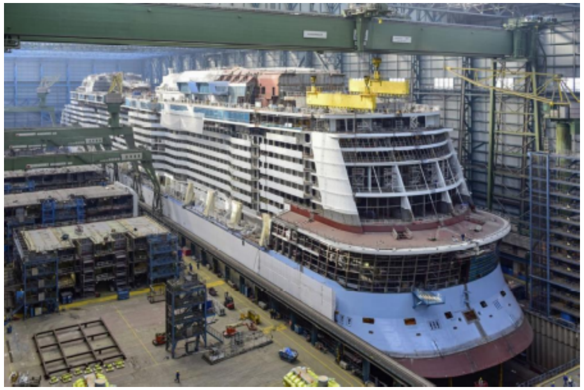
Recent advances on large scheduling problems in CP Optimizer

Philippe Laborie, IBM
laborie@fr.ibm.com





In this presentation

- 1. Recap about IBM CP Optimizer

  By the way: if you have CPLEX, you also have CP Optimizer!

- 2. Performance improvements on scheduling problems in coming release V12.9

CP Optimizer for scheduling in two sentences

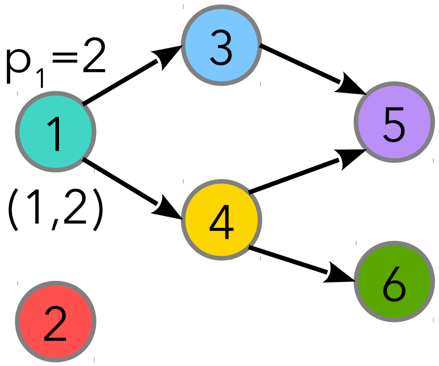
- A **mathematical modeling language** for combinatorial optimization problems that extends ILP (and classical CP) with some algebra on **intervals** and **functions** allowing compact and maintainable formulations for complex scheduling problems
- A continuously improving **automatic search algorithm** that is complete, anytime, efficient (e.g. competitive with problem-specific algorithms on classical problems) and scalable
- Recent review of CP Optimizer (modeling concepts, applications, examples, tools, performance,…) :
  - *IBM ILOG CP Optimizer for scheduling*. Constraints (2018) 23:210–250. http://ibm.biz/Constraints2018

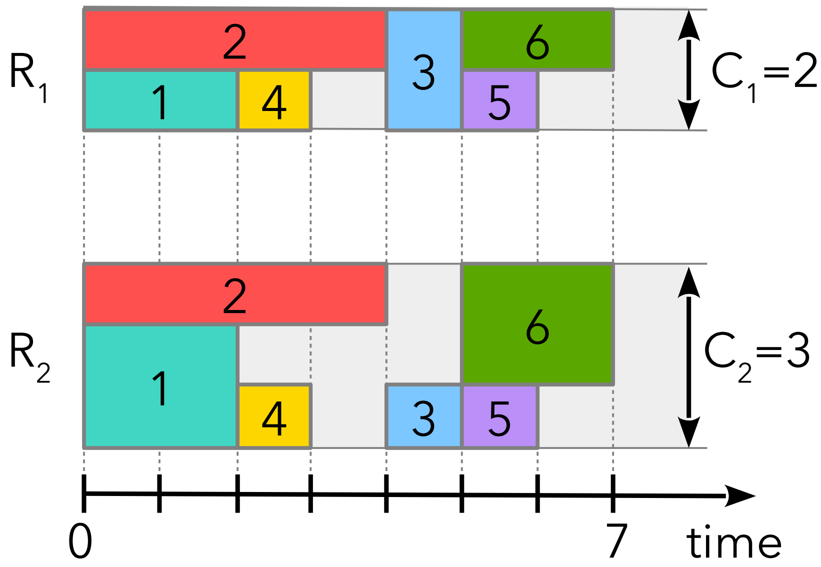
Typical scheduling problems for CP Optimizer

	Family	Tests
	Total / Average	3406
1		7
2	ABCProblem	9
3	Acid	1
4	AircraftAssemblyDS	2
5	AircraftAssemblyShifts	1
6	AircraftLines	7
7	AircraftMaintenance	1
8	AircraftParts	1
9	AirLand	25
10	AirportGates	1
11	AirTrafficFlowManagement	1
12	AssemblyWithInventories	3
13	AuditScheduling	40
14	BlockingJobShop	5
15	BreakScheduling	1
16	Cargo	15
17	Carpet-cutting	10
18	Cities	1
19	Coils CoilBatching	1
20	Coils Global CoilBatching	1
21	CommonDueDate	20
22	CoupledTasks	4
23	CP2015Competition	14
24	CraneSchedulingAverage	14
25	CraneSchedulingDifficult	19
26	CropScheduling	1
27	CumulativeJobShop	38
28	CVRPTW	29
29	Cyclic-rcpsp	5
30	DetailedProjectScheduling	4
31	DevProject	1
32	Diffusion	10
33	Doors	1
34	DynamicResourceFeasibility	18
35	EarthObservationSatellite	1
36	Filters	15
37	FishBoats	2
38	Fjsp	5
39		1
40	FlexibleJobShop	56
41	FlowShop	12
42	FlowShopBuffers	30
43	FlowShopEarliTardi	12
44	FlowShopMinMax	6
45	GEOCAPEObservationScheduling	15
46	Gfd-schedule	10
47	Ghoulomb	5
48	HoistScheduling	10
49	Hooker	15
50	HookerCost	15
51	HugeFlexibleJobShop	4
52	HugeHookerCost	7
53	HugeJobShop	4
54	HugeOpenShop	4
55	HugeRCPSP	7
56	HugeSingleCumulative	4
57	HugeSingleNoOverlap	3
58	Inspectors	1
59	InstructionScheduling	1
60	JobShop	34
61	JobShopEarliTardi	48
62	JobShopEnergy	10
63	JobShopEnergyLimit	6
64	JobShopOperators	5
65	JobShopOperatorsFlowTime	5
66	JobShopTotalFlowTime	15
67	LargeRCPSP	6
68	LargeScheduling	8
69	Largescheduling	10
70	LargeShopScheduling	6
71	LargeTimeNet	3
72	MaintenanceScheduling	1
73	ManpowerScheduling	1
74	Manufacturing	1
75	MinPeak	1
76	MISTA2013Challenge	10
77	MixedCriticalityMatchUp	16
78	MMASP	34
79	MMRL	30
80	Mrcpsp	5
81	Mspsp	6
82	MultiModeRCPSP	402
83	MultiModeRCPSPMax	54
84	MultiprocessorMakespan	2
85	MultiProcessorTotalTardiness	20
86	MultiprocessorWithCommunicationDelays	60
87	MultiSkillsRCPSP	20
88	MultiStageHybridFlowShop	20
89	NewProductsTesting	8
90	OpenShop	28
91	Openshop	5
92	Oversubscribed	150
93	Pallets	1
94	ParallelMachines	1
95	ParallelMachinesDates	1
96	PathSelection	1
97	PatientScheduling	1
98	PermutationFlowShop	20
99	PermutationFlowshopMaintenanceEarliTardi	10
100	PermutationFlowshopMaintenanceMakespan	10
101	Photolithography	1
102	PickupDelivery	1
103	PPOSingleMachine	116
104	ProductionBatches	1
105	ProductionPlanning	2
106	ProductionWithAlternatives	1
107	ProductionWithCalendars	1
108	Profiles	18
109	QMRCPSP	400
110	QuarriesScheduling	1
111	QuayCraneScheduling	12
112	Racp	5
113	RCPSP	440
114	Rcpsp-Net	10
115	RCPSPDC	50
116	RCPSPEarliTardi	65
117	RCPSPImbalance	1
118	RCPSPMax	80
119	RCPSPMaxCal	6
120	RCPSPMaxInventories	12
121	Rcpsp	5
122	Rectangle-packing	5
123	ResourceAvailabilityCost	30
124	SchedSCS	4
125	SchedulingGeneralTimeLags	10
126	SelfPlanner	70
127	SemiconductorTesting	18
128	SequenceDates	1
129	SequenceSetupTimes	1
130	SetupMinimizer	1
131	SingleMachineEarliTardi	40
132	SingleMachineTardiTasks	100
133	SkilledOperators	14
134	Smelt	5
135	SMSDST	16
136	SPLC	10
137	StressedJobShop	15
138	TankTruckScheduling	1
139	Test-scheduling	5
140	TimeTabling	1
141	TrainingBatch	2
142	TrainingProject	3
143	TrainingSatellite	1
144	TrainingSelfPlanner	1
145	TrainingTransportation	2
146	Trolley	15
147	TruckingWood	1
148	TruckScheduling	1
149	TruckSchedulingFlexibleShifts	1
150	TSP	101
151	UnaryAlternativeTransitionTime	39
152	UPMST	10
153		1
154	VRPBalancedVehicles	7
155	Vrplc	5
156	YogurtScheduling	10

Illustration on an academical scheduling problem

- 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





Formulation of the RCPSP in ILP

- Example of the Start/End Event-based formulation (SEE)

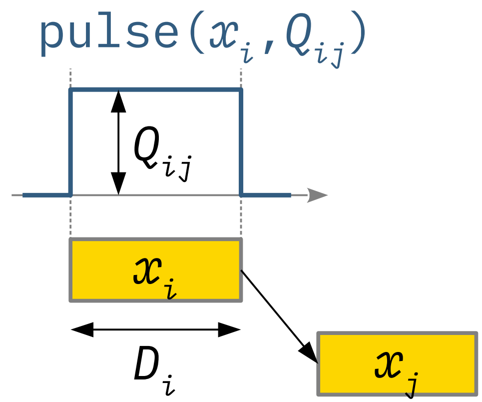
$$
\begin{aligned}
&\min \; s_{n+1} \\
&\sum_{k \in K} x_{ik} = 1 && \forall i \in I \\
&\sum_{k \in K'} y_{ik} = 1 && \forall i \in I \\
&s_k - s_l + p_i(x_{ik} + y_{il} - 1) \leq 0 && \forall i \in I, (k,l) \in A \\
&s_k - s_{k+1} \leq 0 && \forall k \in K \\
&\sum_{i \in I} d_{ir} \left(\sum_{k'=1}^{k} x_{ik'} - \sum_{k'=2}^{k} y_{ik'} \right) \leq D_r && \forall k \in K, r \in R \\
&\sum_{k'=2}^{k} y_{ik'} + \sum_{k'=k}^{n} x_{ik'} \leq 1 && \forall i \in I, k \in K \\
&\sum_{k'=1}^{k} x_{jk'} + \sum_{k'=k+1}^{n+1} y_{ik'} \leq 1 && \forall (i,j) \in E, k \in K \\
&x_{ik} \in \{0,1\} && \forall i \in I, k \in K \\
&y_{ik} \in \{0,1\} && \forall i \in I, k \in K' \\
&s_k \geq 0 && \forall k \in K \cup \{n+1\}
\end{aligned}
$$

$O(N^3)$

* A. Tesch. Compact MIP Models for the Resource-Constrained Project Scheduling Problem. Master's Thesis. Technische Universität Berlin. 2015.

Formulation of the RCPSP in CP Optimizer

- Use of **interval** variables for tasks and cumul **functions** for resource usage



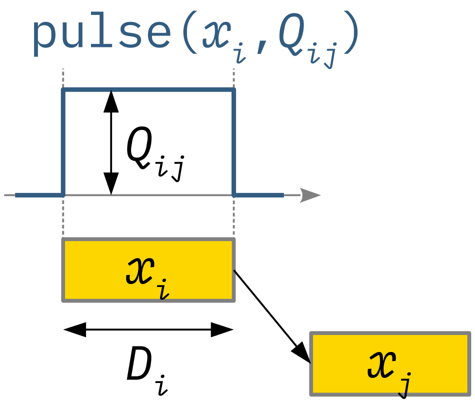
$$
\begin{array}{lll}
\min & \max\limits_{i \in [1,N]} \mathrm{endOf}(x_i) & \\
& \sum\limits_{i \in [1,N]} \mathrm{pulse}(x_i, Q_{ij}) \leq C_j & \forall j \in [1, M] \\
& \mathrm{endBeforeStart}(x_i, x_j) & \forall (i,j) \in P \\
& \mathrm{interval}\ \ x_i,\ \ \mathrm{size} = D_i & \forall i \in [1, N]
\end{array}
$$

O(N)

compact

Formulation of the RCPSP in CP Optimizer

- Use of **interval** variables for tasks and cumul **functions** for resource usage




```
from docplex.cp.model import *
model = CpoModel()

# Decision variables: tasks x
x = [interval_var(size=D[i]) for i in N]

# Objective: minimize project makespan
model.add(minimize(max(end_of(x[i]) for i in N)))

# Constraints: resource capacities
for j in M: model.add(sum(pulse(x[i],q) for [i,q] in R[j]) <= C[j])

# Constraints: precedence between tasks
for [i,j] in P: model.add(end_before_start(x[i],x[j]))

# Solve the model
sol = model.solve(TimeLimit=300)
```

compact

Classical RCPSP benchmarks (5023 instances)

- CP Optimizer 12.9 results with automatic search using a time limit of 5mn (code of previous slide)

Benchmark	Size	#Instances	Average distance to best known solution	Ratio of optimality proofs
Patterson	[5-49]	110	0.00%	100.00%
AT	[27-103]	144	-0.42%	74.31%
KSD30	30	480	0.00%	100.00%
KSD60	60	480	0.11%	88.33%
KSD90	90	480	0.33%	81.46%
KSD120	120	600	1.09%	46.00%
BL	[20-25]	39	0.00%	100.00%
PACK	[15-33]	55	-0.02%	67.27%
RG300	300	300	1.35%	6.04%
RG30	30	1800	-0.34%	93.44%
KSD15-D	15	480	0.00%	100.00%
PACK-D	[15-33]	55	0.00%	65.45%

complete

efficient

- Maximal time to first feasible solution: 0.08s

anytime

Classical RCPSP benchmarks

- CP Optimizer 12.9 results with automatic search

Benchmark	Average distance to best solution of [Schutt&all-2013] 300s	Ratio of optimality proofs CP Optimizer 12.9 300s / 600s-FDS	Ratio of optimality proofs LCG [Schutt&all-2013] 600s	Ratio of optimality proofs SMT [Ansótegui&all-2011] 500s
Patterson	**0.00%**	**100.00% / 100.00%**	100.00%	
AT	**-0.45%**	74.31% / 77.08%	**89.58%**	
KSD30	**0.00%**	**100.00% / 100.00%**	100.00%	100.00%
KSD60	**-0.18%**	88.33% / 89.17%	**90.00%**	
KSD90	**-0.48%**	81.46% / **83.75%**	83.33%	
KSD120	**-1.74%**	46.00% / **47.33%**	47.17%	
BL	**0.00%**	**100.00% / 100.00%**	100.00%	
PACK	**-0.02%**	67.27% / **74.55%**	70.91%	65.00%
KSD15-D	**0.00%**	**100.00% / 100.00%**	100.00%	100.00%
PACK-D	**0.00%**	65.45% / **70.91%**	67.26%	43.00%

efficient

complete

Classical RCPSP benchmarks

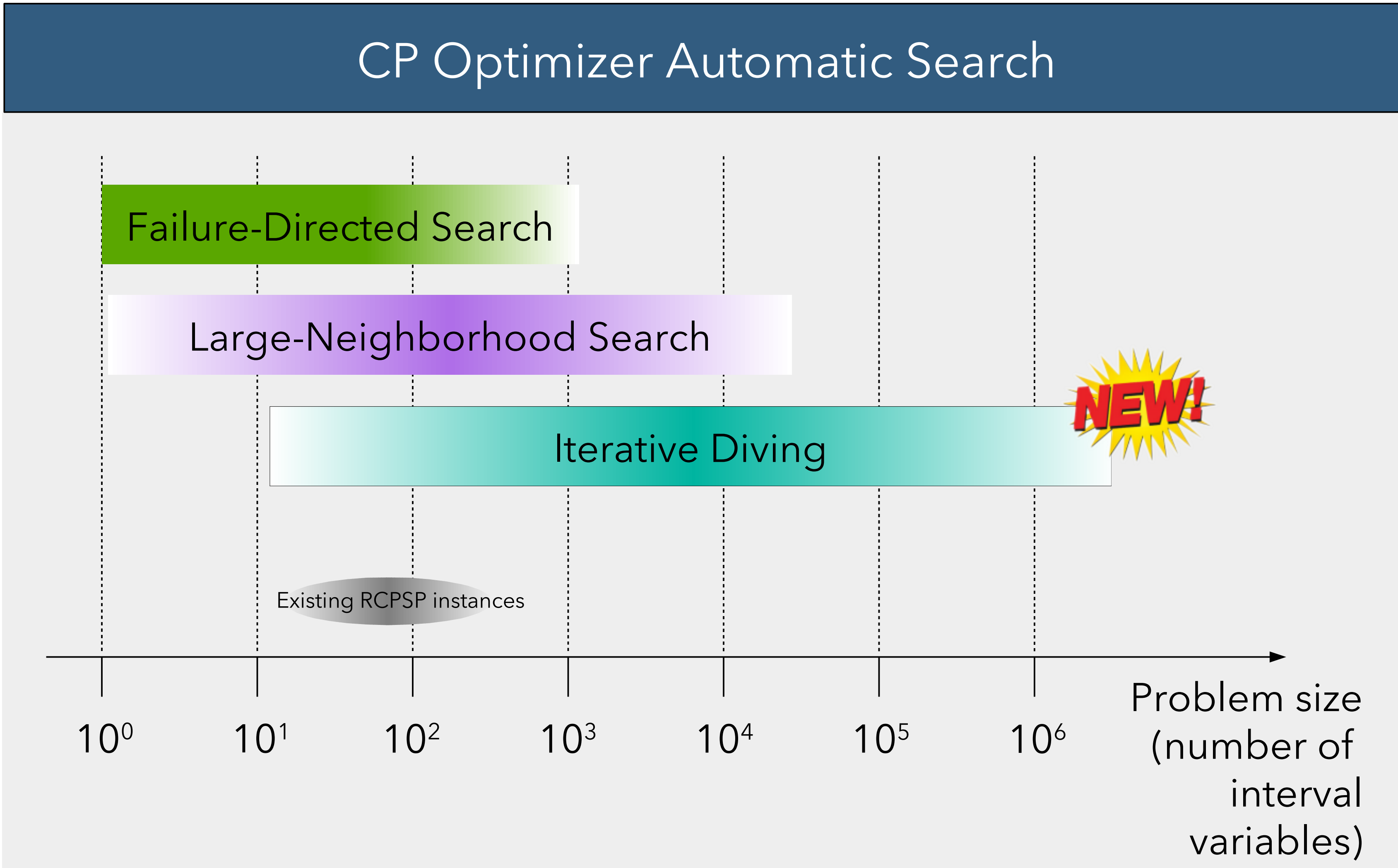
- CP Optimizer 12.9 results with automatic search using a time limit of 5mn

Benchmark	Size	#Instances	Average distance to best known solution	Ratio of optimality proofs
Patterson	[5-49]	110	0.00%	100.00%
AT	[27-103]	144	-0.42%	74.31%
KSD30	30	480	0.00%	100.00%
KSD60	60	480	0.11%	88.33%
KSD90	90	480	0.33%	81.46%
KSD120	120	600	1.09%	46.00%
BL	[20-25]	39	0.00%	100.00%
PACK	[15-33]	55	-0.02%	67.27%
RG300	300	300	1.35%	6.04%
RG30	30	1800	-0.34%	93.44%
KSD15-D	15	480	0.00%	100.00%
PACK-D	[15-33]	55	0.00%	65.45%

- These problems are tiny compared to industrial problems !

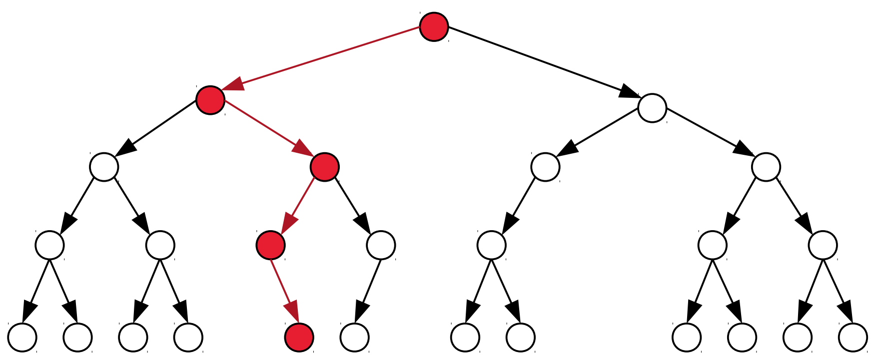
Automatic Search

- Main principle: cooperation between several approaches



Iterative diving

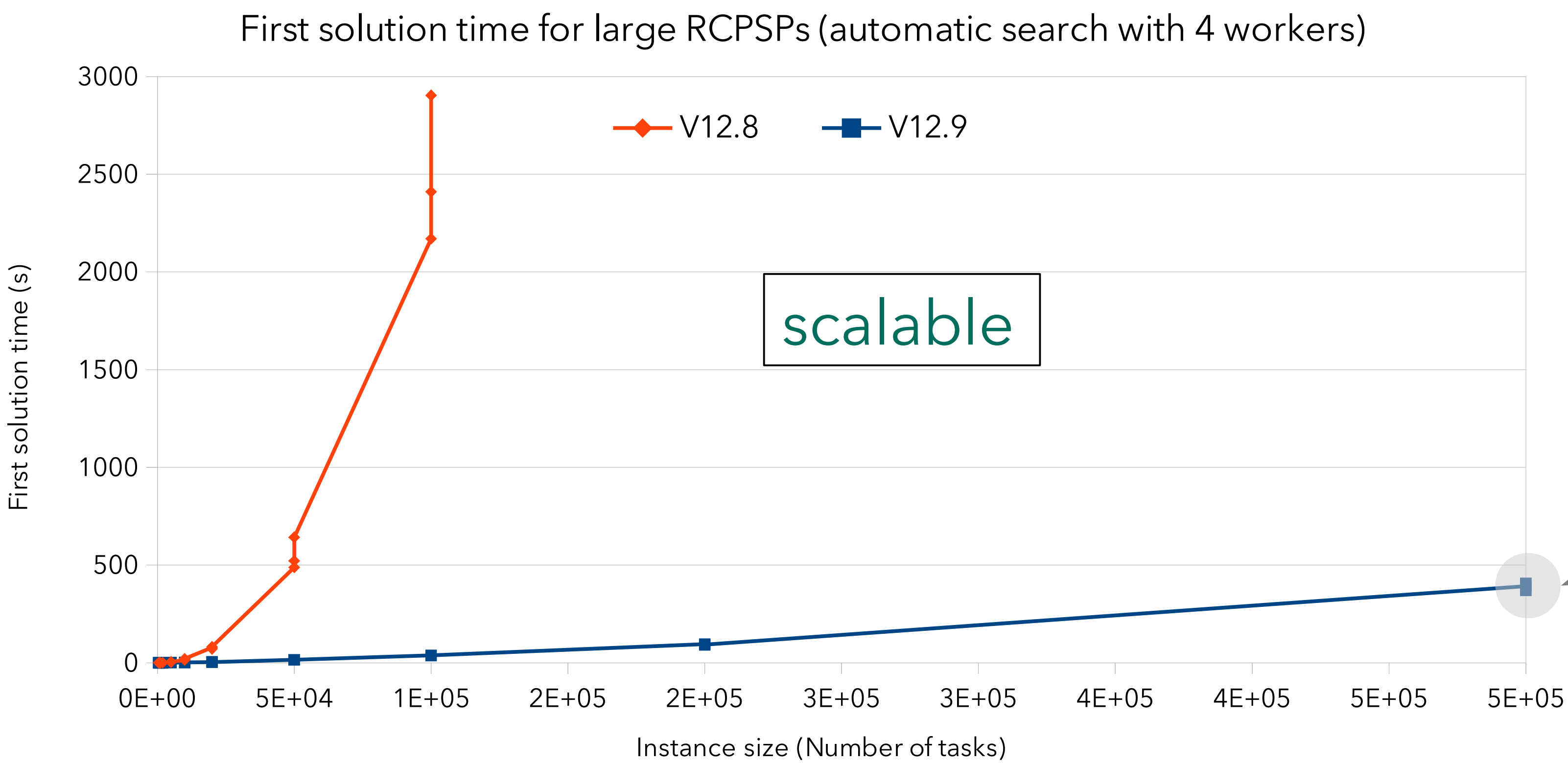
- By the way: "**CP**" in "**CP** Optimizer" means Constraint Programming ☺
- Idea of iterative diving: perform aggressive dives (no backtrack) in the search tree explored by CP



- For instance on RCPSP it boils down to some very classical ideas (list scheduling, decoding schemes, …)
- The challenge was to generalize these problem-specific ideas to the general modeling concepts of CP Optimizer

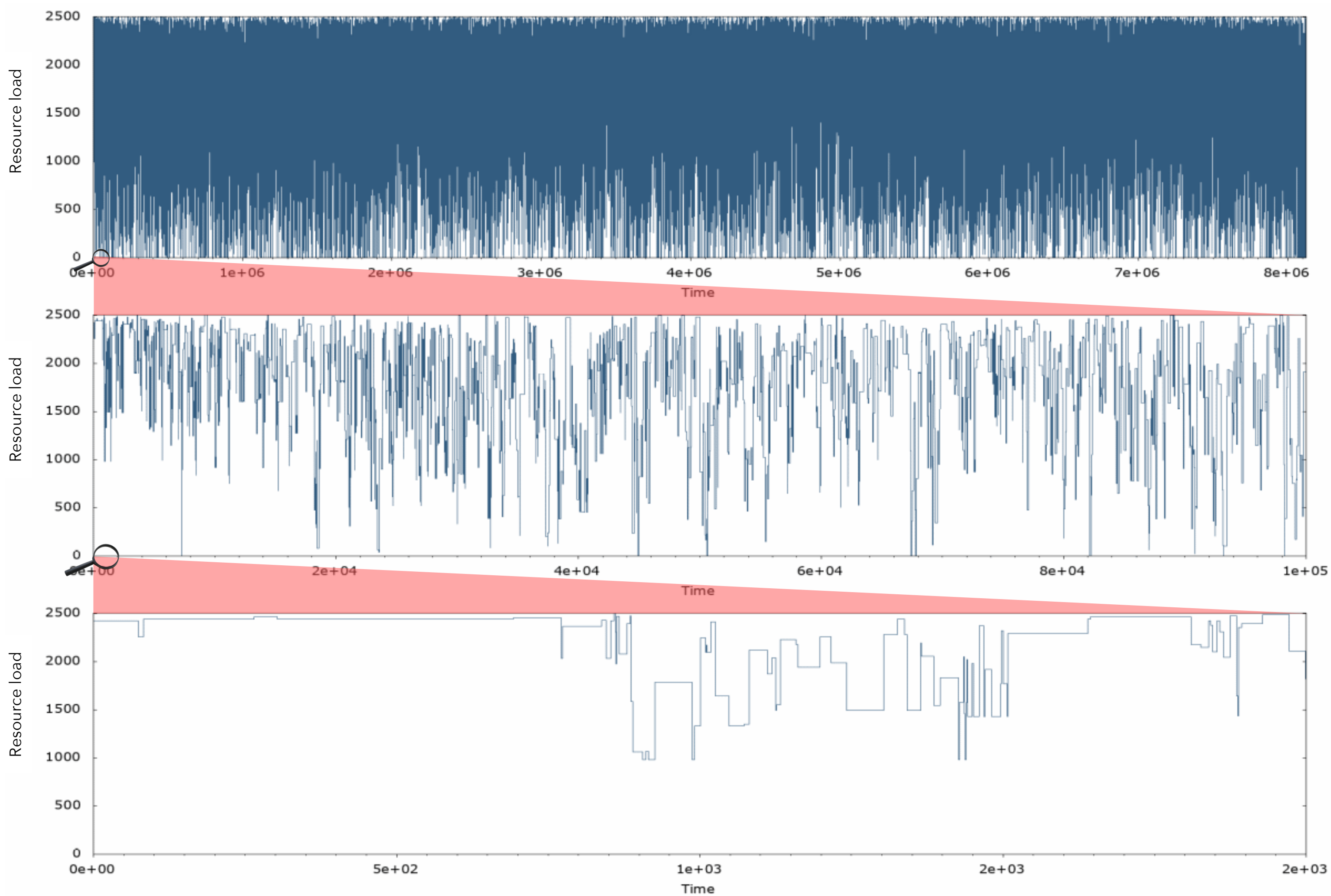
CP Optimizer V12.9

- New benchmark with RCPSPs from 500 to 500.000 tasks
  - Largest problem: 500.000 tasks, 79 resources, 4.740.783 precedences, 4.433.550 resource requirements
- Time to first feasible solution V12.8 v.s. V12.9 (with still the very same formulation as on slide 8)



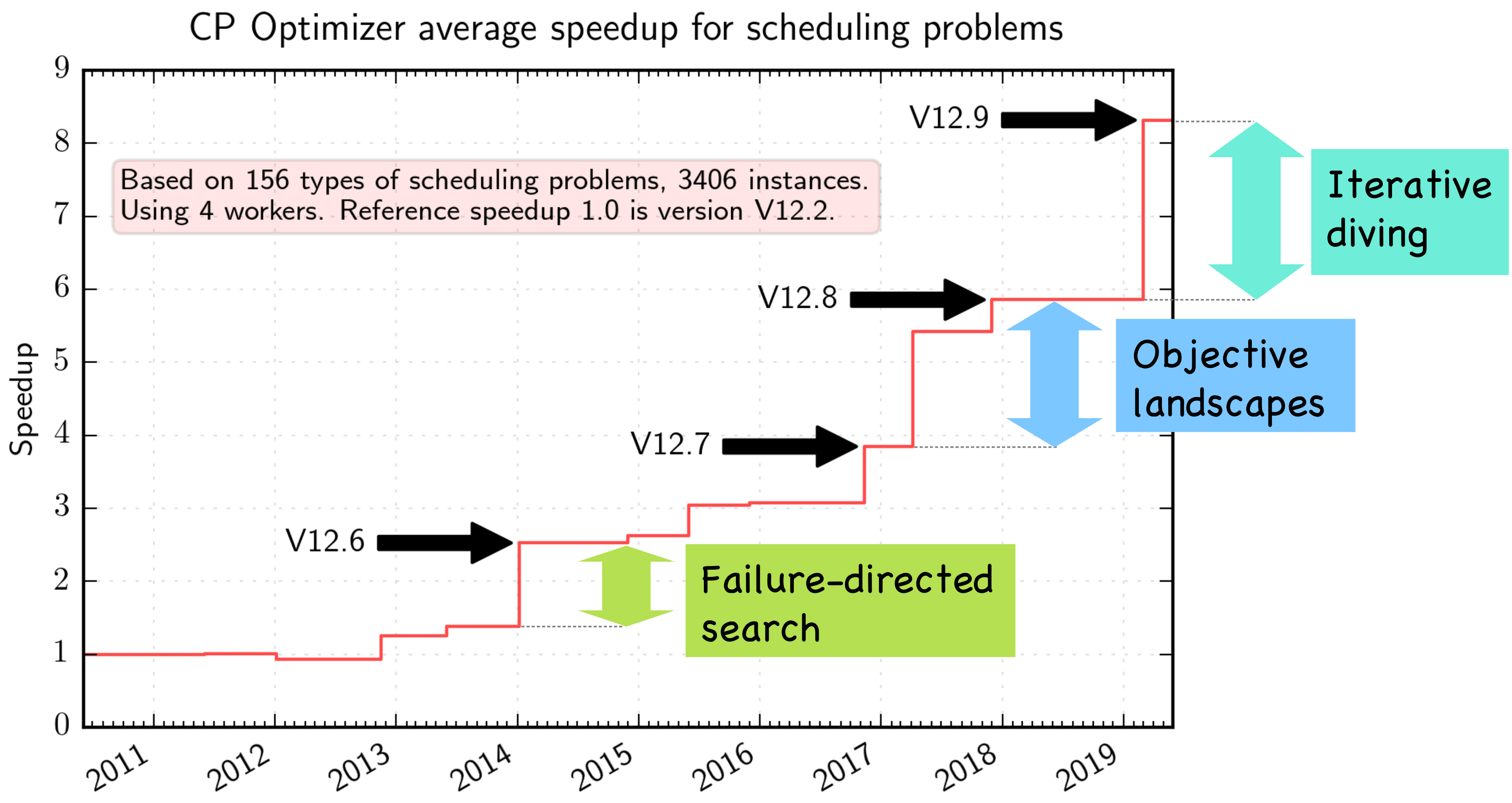
Example of cumul function value

- Resource load for one of the resources of a large instance (500.000 tasks)



CP Optimizer automatic search

- With 4 workers, average speed-up of 42% between 12.8 and coming version 12.9



Conclusion

- A **mathematical modeling language** for combinatorial optimization problems that extends ILP (and classical CP) with some algebra on **intervals** and **functions** allowing compact and maintainable formulations for complex scheduling problems

- A continuously improving **automatic search algorithm** that is complete, anytime, efficient (e.g. competitive with problem-specific algorithms on classical problems) and scalable

- Recent review of CP Optimizer (modeling concepts, applications, examples, tools, performance,…) :
  - ***IBM ILOG CP Optimizer for scheduling***. Constraints (2018) 23:210–250. **http://ibm.biz/Constraints2018**