

Experiments

Helmut Simonis

Constraint Based Production Scheduling

Licence

This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

This license requires that reusers give credit to the creator. It allows reusers to distribute, remix, adapt, and build upon the material in any medium or format, for noncommercial purposes only. If others modify or adapt the material, they must license the modified material under identical terms.



Acknowledgments



This publication was developed as part of the ENTIRE EDIH project, which received funding from Enterprise Ireland and the European Commission.

Part of this work is based on research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics at UCC, which is co-funded under the European Regional Development Fund.

Part of this work is based on research conducted within the ASSISTANT European project, under the framework program Horizon 2020, ICT-38-2020, Artificial intelligence for manufacturing, grant agreement number 101000165.

Key Points



- This section describes the scheduling tool
 - This is a *preview* of the current state, not released yet!
- How to load/create data
 - From files
 - By instance generator
 - From benchmark problems
- How to run the solvers
 - Which solvers are supported
 - What to expect in terms of performance
- Experiments to try
 - Limited time
 - Possible "test before invest" continuation

The Scheduling Tool



- We create the tool as basis for experiments
- To test ideas and solvers
- As a teaching tool
- Slightly higher standard than usual academic prototypes
 - This is a *preview*, not released yet
- Not a commercial tool
 - But can use commercial solvers
 - Also open-source solvers
- Written in Java, JavaFX
- Can also be used as a back-end scheduling server
- Uses our Java application framework generator
- Will become available in early 2025

Back-end solvers



- Provide both open-source and commercial solver interfaces
- Allow experimentation without having to buy commercial tools straightaway
- Gives a level playing field to compare solvers and models
- Provides out-of-the-box, generic performance

Google OR-Tools CPSat Solver



- Open-Source tool provided by Google
- Available at https://developers.google.com/optimization/cp/cp_solver
- Probably best open-source CP solver for scheduling
- This solver is packaged with scheduler

Example Problem

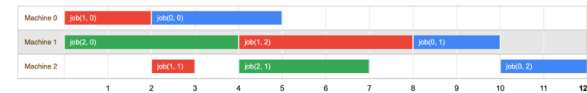
Below is a simple example of a job shop problem, in which each task is labeled by a pair of numbers (m, p) where m is the number of the machine the task must be processed on and p is the *processing time* of the task – the amount of time it requires. (The numbering of jobs and machines starts at 0.)

- Job 0 = $[(0, 3), (1, 2), (2, 2)]$
- Job 1 = $[(0, 2), (2, 1), (1, 4)]$
- Job 2 = $[(1, 4), (2, 3)]$

In the example, Job 0 has three tasks. The first, $(0, 3)$, must be processed on machine 0 in 3 units of time. The second, $(1, 2)$, must be processed on machine 1 in 2 units of time, and so on. Altogether, there are eight tasks.

A solution for the problem

A solution to the job shop problem is an assignment of a start time for each task, which meets the constraints given above. The diagram below shows one possible solution for the problem:



You can check that the tasks for each job are scheduled at non-overlapping time intervals, in the order given by the problem.

The length of this solution is 12, which is the first time when all three jobs are complete. However, as you will see [below](#), this is not the optimal solution to the problem.

(from OR-Tools website)

CP Optimizer from IBM



- Commercial tool of IBM
- <https://www.ibm.com/products/ilog-cplex-optimization-cplex-cp-optimizer>
- Part of optimization suite with Cplex, OPL
- We do **not** provide this solver, we allow to interface with it
- Academic licenses available
- Well-known for capabilities for scheduling

Resources



Applications of constraint programming

Explore applications of constraint programming including production problem and scheduling use cases.

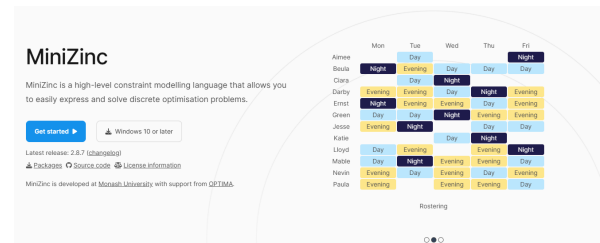
[Read the documentation](#) →

(from CPOptimizer website)

MiniZinc from Monash University



- Modelling language and backend tools from Monash University in Melbourne, Australia
- Available from <https://www.minizinc.org/>
- Widely used for teaching
- Allows different backend solver to run from same model
- Generic CP tool, not optimized for scheduling
- Requires separate installation, open-source



(from MiniZinc Website)

Which Solver is Better?



- We present results on a few benchmark types
- Fair comparison between solvers
 - Same hardware, Windows 11 laptop
 - CPU i7-10875H @ 2.3GHz, 64GB, four cores
 - Same timeout (600 s)
- Not a fair comparison to state-of-the-art
 - Uses out-of-the-box model
 - Significant improvements possible
 - More specific models
 - Parameter tuning
 - Unlimited runtime

Group	Nr	All Instances				Optimal Only		Non Optimal Only			
		Optimal (% of All Instances)				Time (% of VB)		Cost (% of VB)		Bound (% of VB)	
		Both	CPO	CPSat	None	CPO	CPSat	CPO	CPSat	CPO	CPSat
15/15	10	90.00	0.00	0.00	10.00	105.19	141.18	100.00	100.00	97.17	100.00
20/15	10	20.00	0.00	0.00	80.00	267.27	263.20	100.99	100.05	98.50	99.93
20/20	10	0.00	0.00	0.00	100.00	n/a	n/a	100.74	100.06	97.96	100.00
30/15	10	10.00	0.00	10.00	80.00	174.32	100.00	100.18	100.49	99.87	100.00
30/20	10	0.00	0.00	0.00	100.00	n/a	n/a	100.30	101.30	99.40	100.00
50/15	10	100.00	0.00	0.00	0.00	100.00	685.09	n/a	n/a	n/a	n/a
50/20	10	10.00	60.00	0.00	30.00	100.00	381.38	100.00	101.60	100.00	100.00
100/20	10	10.00	90.00	0.00	0.00	100.00	416.13	100.00	101.73	100.00	66.81

- Significant number of problems solved to optimality in 600s
- In terms of quality, solvers are quite similar
- CPO wins in terms of solution times for larger instances

Results for Hybrid Flexible Flow-Shop

Group	Nr	All Instances				Optimal Only		Non Optimal Only			
		Optimal (% of All Instances)				Time (% of VB)		Cost (% of VB)		Bound (% of VB)	
		Both	CPO	CPSat	None	CPO	CPSat	CPO	CPSat	CPO	CPSat
20	25	76.00	0.00	20.00	4.00	100.00	580.71	100.00	100.00	96.52	100.00
25	25	80.00	0.00	8.00	12.00	101.65	238.02	100.00	100.37	97.67	100.00
30	25	60.00	0.00	4.00	36.00	100.35	264.69	100.18	101.05	100.00	100.00
40	25	4.00	16.00	0.00	80.00	100.00	2554.03	100.00	104.68	100.00	100.00
50	25	0.00	4.00	0.00	96.00	n/a	n/a	100.00	107.87	100.00	100.00
100	25	0.00	0.00	0.00	100.00	n/a	n/a	100.00	120.43	100.00	100.00
200	25	0.00	0.00	0.00	100.00	n/a	n/a	100.00	188.60	100.00	100.00
300	24	0.00	0.00	0.00	100.00	n/a	n/a	100.00	263.22	100.00	100.00
400	25	0.00	0.00	0.00	100.00	n/a	n/a	100.00	246.34	100.00	100.00

- Only smaller/medium instances solved to optimality
- For those problems, both solvers perform well
- CPO significantly better on large instances

General Recommendations



- If you already have access to CPO, use it!
- For new problem types, do an evaluation with CPSat first
- Out of the box, CPO performs more consistently
- May be easier to extend CPSat with your own research
- Use multiple cores and memory to your advantage

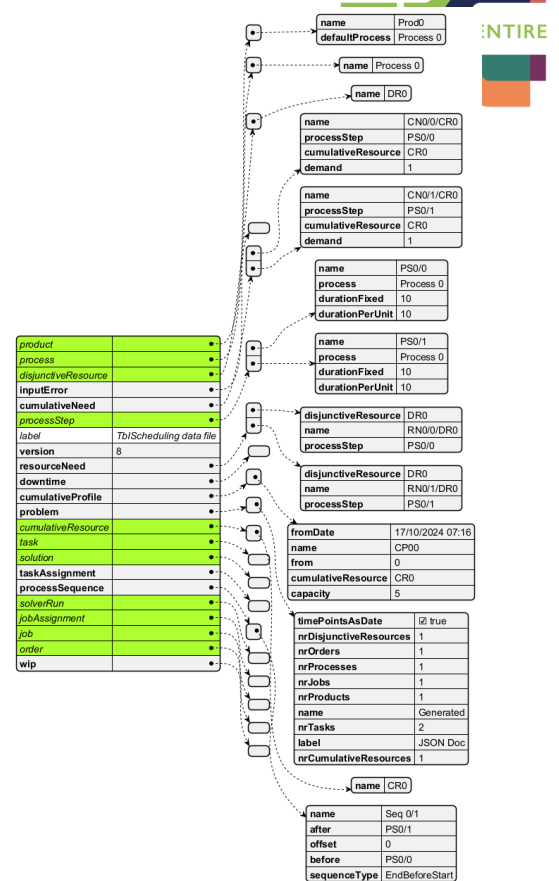
Input Data



- We have defined a specific JSON data format to describe scheduling problems
- This is different from the native/XML data format of the application (do not use)
- Load with menu `File - Load DataFile...`
- Save with menu `File - Save DataFile...`
- The format is described in a document

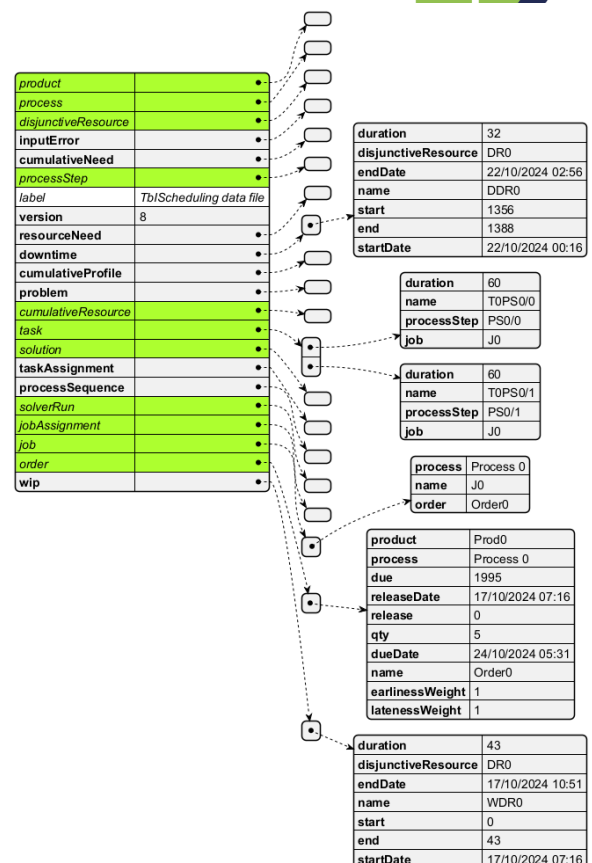
Base Data

- Description of
 - Product
 - Process
 - DisjunctiveResource
 - CumulativeNeed
 - ProcessStep
 - ResourceNeed
 - CumulativeProfile
 - Problem
 - CumulativeResource
 - ProcessSequence



Schedule Input Data


- Description of
 - Downtime
 - Task (x2)
 - Job
 - Order
 - WiP



- We use the same JSON format to describe the results of the schedule
- Added field types for SolverRun, Solution, assigned Jobs and Tasks

Sample Results

- Description of
 - Solution
 - SolverRun
 - Job Assignment
 - Task Assignment



product	103
process	1032
disjunctiveResource	17/10/2024 20:51
inputError	0
totalWaitBefore	0
totalWaitAfter	0
duration	100
maxLateness	0
maxWaitAfter	0
inLate	0
gfp	0
solverRun	Run1
end	103
weightedEarliness	1032
totalLateness	0
maxWaitBefore	0
bound	103
solverStatus	Optimal
start	43
totalWaitAfter	0
percentLate	0
weightedLateness	0
maxEarliness	1032
name	So11
objectiveValue	103
flowtime	103
startDate	17/10/2024 10:51

duration	60
waitAfter	0
task	TSPSO0
disjunctiveResource	DR0
endDate	17/10/2024 15:51
name	TA0
start	43
end	103
jobAssignment	JAO
waitBefore	0
startDate	17/10/2024 10:51

duration	60
waitAfter	0
task	TSPSO1
disjunctiveResource	DR0
endDate	17/10/2024 20:51
name	TA1
start	103
end	103
jobAssignment	JAO
waitBefore	0
startDate	17/10/2024 15:51

nThreads	2
weightEarliness	1
seed	42
solverBackend	None
weightMakespan	1
producePDF	<input type="checkbox"/> false
solverStatus	Optimal
description	
label	
modelType	CPO
enforceWtp	<input checked="" type="checkbox"/> true
objectiveType	Makespan
timeout	30
weightFlowtime	1
enforceCumulative	<input checked="" type="checkbox"/> true
name	Run1
weightLateness	1
time	0.142
produceReport	<input type="checkbox"/> false
enforceOverdime	<input checked="" type="checkbox"/> true
removeSolution	<input type="checkbox"/> false
enforceReleaseDate	<input checked="" type="checkbox"/> true
enforceDueDate	<input type="checkbox"/> false

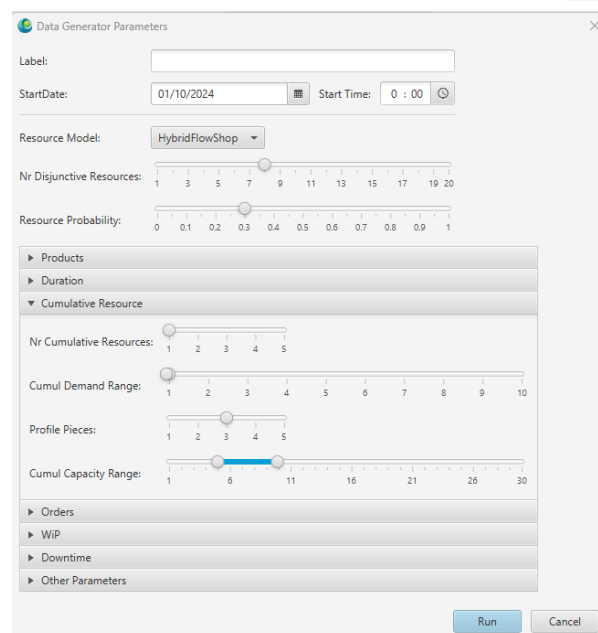
duration	100
solution	So11
late	0
endDate	17/10/2024 20:51
name	JAO
start	43
end	103
job	J0
early	1032
startDate	17/10/2024 10:51

- Application allows to generate different types of test problems
- Different types of resource models
- Different numbers of orders, resources, WiP, downtime
- Useful to generate more life-like examples combining different constraint types

Instance Generator Dialog



- Resource Model
 - Select a resource model defining the overall structure of problem
- Nr Disjunctive Resources
 - Describe how many disjunctive resources are generated
- Resource Probability
 - The probability that a resource is compatible with a task
 - Only for some resource models



The screenshot shows the 'Data Generator Parameters' dialog box. It contains several input fields and sliders for configuring test problem parameters. The 'Label' field is empty. The 'StartDate' is set to '01/10/2024' and 'Start Time' is '0 : 00'. The 'Resource Model' is set to 'HybridFlowShop'. The 'Nr Disjunctive Resources' slider is set to 7. The 'Resource Probability' slider is set to 0.3. The 'Products' section is expanded, showing 'Nr Cumulative Resources' set to 1, 'Cumul Demand Range' set to 1, 'Profile Pieces' set to 3, and 'Cumul Capacity Range' set to 6. Other sections like 'Orders', 'WiP', 'Downtime', and 'Other Parameters' are collapsed. 'Run' and 'Cancel' buttons are at the bottom right.

Resource Models



- Flow-Shop
 - Multiple stages, all jobs use machines in same order
- Job-Shop
 - Multiple stages, jobs use machines in different order
- Open-Shop
 - Multiple stages, no predefined order of machines
- Hybrid Flow-Shop (default)
- Hybrid Job-Shop
- Hybrid Open-Shop
 - Like x-shop, but with multiple machines per stage
- Random
 - Multiple stages, each stage using a random subset of machines
- All
 - Multiple stages, each stage allowing all machines

Instance Generator - Products



- Nr Products
 - Number of products to be generated
 - Products may be reused by multiple orders
- Stages Range
 - Range slider, sets lower and upper bound on number of stages

A screenshot of the "Data Generator Parameters" dialog box. It contains several input fields and sliders. The "Label" field is empty. The "StartDate" is set to "01/10/2024" and the "Start Time" is "0 : 00". The "Resource Model" is set to "HybridFlowShop". The "Nr Disjunctive Resources" slider is set to 7. The "Resource Probability" slider is set to 0.3. The "Products" section is expanded, showing "Nr Products" set to 10 and "Stages Range" set to 4. Other sections like "Duration", "Cumulative Resource", "Orders", "WIP", "Downtime", and "Other Parameters" are collapsed. "Run" and "Cancel" buttons are at the bottom right.

Instance Generator - Duration



- Duration Model
 - Different ways to link duration of processSteps
- Duration Range
 - Range slider to set lower and upper bounds on perUnit duration
- Duration Fixed Factor
 - How fixed and perUnit duration values are linked

The screenshot shows the 'Data Generator Parameters' dialog box with the 'Duration' section expanded. The 'Duration Model' is set to 'Random'. The 'Duration Range' slider is positioned between 6 and 21 on a scale from 1 to 50. The 'Duration Fixed Factor' slider is positioned at 1 on a scale from 0 to 5. The 'Run' button is highlighted in blue.

Instance Generator - Cumulative



- Nr Cumulative Resources
 - Number of cumulative resources generated
- Cumul Demand Range
 - Range slider to select lower and upper bound on cumulativeResource-Need demands
- Profile Pieces
 - Number of segments of CumulativeProfile generated for each resource
- Cumul Capacity Range
 - Range slider to select lower and upper bounds on cumulative profile capacity values

The screenshot shows the 'Data Generator Parameters' dialog box with the 'Cumulative Resource' section expanded. The 'Nr Cumulative Resources' slider is positioned at 3 on a scale from 1 to 5. The 'Cumul Demand Range' slider is positioned between 1 and 4 on a scale from 1 to 10. The 'Profile Pieces' slider is positioned at 3 on a scale from 1 to 5. The 'Cumul Capacity Range' slider is positioned between 6 and 11 on a scale from 1 to 30. The 'Run' button is highlighted in blue.

Instance Generator - Orders



- Nr Orders
 - Number of orders generated, each order is assigned a random product/process
- Qty Range
 - Range slider to select lower and upper bounds on quantity for each order

The screenshot shows the 'Data Generator Parameters' dialog box with the 'Orders' section expanded. The 'Nr Orders' is set to 50. The 'Qty Range' slider is positioned between 1 and 30, with a blue bar indicating the range. Other sections like 'Products', 'Duration', 'Cumulative Resource', 'WIP', 'Downtime', and 'Other Parameters' are collapsed. The 'Run' and 'Cancel' buttons are at the bottom right.

Instance Generator - WiP (Work in Progress)



- WiP Probability
 - Probability of generating a WiP for a disjunctive resource
- WiP Range
 - Range slider to set lower and upper bound on WiP duration

The screenshot shows the 'Data Generator Parameters' dialog box with the 'WiP' section expanded. The 'WiP Probability' slider is set to 0.5. The 'WiP Range' slider is positioned between 1 and 100, with a blue bar indicating the range. Other sections like 'Products', 'Duration', 'Cumulative Resource', 'Orders', 'Downtime', and 'Other Parameters' are collapsed. The 'Run' and 'Cancel' buttons are at the bottom right.

Instance Generator - Downtime



- Downtime Probability
 - Probability of generating a downtime for a disjunctive resource
- Downtime Range
 - Range slider to select lower and upper bounds on downtime duration

The screenshot shows the 'Data Generator Parameters' dialog box with the 'Downtime' section expanded. The 'Downtime Probability' slider is set to 0.5. The 'Downtime Range' slider is set from 51 to 71. The 'Run' button is highlighted.

Instance Generator - Other Parameters



- Earliest Due
 - Smallest allowed value for a due date
- Horizon Days
 - What planning horizon to consider (in days)
- Time Resolution
 - In minutes, links internal and external time presentation
- Random Seed
 - Random seed to make reproducible random choices

The screenshot shows the 'Data Generator Parameters' dialog box with the 'Other Parameters' section expanded. The 'Earliest Due' field is set to 20, 'Horizon Days' is set to 20, 'Time Resolution' is set to 5, and 'Random Seed' is set to 42. The 'Run' button is highlighted.



- Three datasets of different sizes
 - Job-shop
 - Flow-shop
 - Open-shop
- Load with menu `File - Load DataFile...` -
`Taillard -`
- Larger instances need more solver time to reach good solutions (600 s)

Simple Assembly Line Balancing Problem (SALBP)



- Will be discussed on more details as case study
- Design an assembly line setup by solving a scheduling problem
- Balance a set of operations across a number of stations of an assembly line
- Precedence graph is not a chain, can be very complex
- Specialized problem normally solved with specialized tools
- Load with menu `File - Load SALBP Problem...`

Test Scheduling Benchmark set from ABB



- Will be discussed in more details as case study
- Schedule a set of tests on a number of machines, minimizing total duration
- Single stage tests, possibly large number of resources
- Closely related to bin-packing
- Load with menu `File - Load Test Scheduling Problem...`

Experiment 1



- Start the application
 - Our running example will be automatically generated
- Look at the process diagram `Window-Product-Process Diagram`
- Run the solver `Scenario - Run ScheduleJobs Solver`
- Observe the results in Gantt Chart
- Customize display
- Look at Cumulative Resource Chart
`Window-Solution-Cumulative Resource Chart`

Experiment 2



- Re-run solver disabling cumulative constraint
- Observe result in Gantt chart
- See impact on Cumulative Resource chart
- Switch between solutions in charts

Experiment 3



- Check Gantt chart display for delayed tasks, enabling lateness display
- Re-run solver, enforcing due-date constraints
- What impact does this have on objective

Experiment 4



- Change objective to on-time delivery
- Results are very different, why?
- More explanations on this tomorrow

Experiment 5



- Load one of the other example types
- For example, Taillard Job-shop 15x15
- Understand process diagram
- Run solver
- Look at intermediate solutions found



- We presented an overview of our generic scheduling tool
- Discussed available solvers, both commercial and open-source
- Described the JSON data format for input and output
- Gave an overview of the instance generator provided
- Shows example problems included with tool
- Suggested some experiments to run