



Enterprise
Ireland



Funded by the
European Union
NextGenerationEU

EDIH | European
Digital Innovation
Hubs Network



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

Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Some Suggested Experiments

Summary

ENTIRE EDIH

Production Scheduling

Slide 5

The Scheduling Tool



- We created 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

Outline



The Scheduling Tool

Under the hood

Google CPSSat

IBM CPOptimizer

MiniZinc

Which solver is better?

Input Data

Result Output

Instance Generator

Predefined Problem Sets

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](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

A screenshot of the MiniZinc website. At the top, there's a navigation bar with links for "Get started", "Windows 10 or later", "Latest release 2.8.7 (GitHub)" (with a download icon), "Tasks", "Sources", "Code", "Licence information", and "About". Below the navigation, a section titled "MiniZinc" describes it as a high-level constraint modeling language for discrete optimization problems. It includes a "Get started" button and a "Windows 10 or later" link. Further down, there's a "Tasks" section with a grid visualization. The grid has columns labeled "Mon", "Tue", "Wed", "Thu", and "Fri" at the top, and rows labeled with names: Anne, Boyd, Clara, Darby, Derry, Eileen, Green, Jason, Katie, Lloyd, Matrix, Niall, and Paula. The grid cells are colored in shades of blue, yellow, and grey, representing different task assignments over time. A legend below the grid indicates colors for "Day", "Evening", and "Night".

	Mon	Tue	Wed	Thu	Fri
Anne	Day	Evening	Day	Day	Day
Boyd	Night	Evening	Day	Night	Day
Clara	Day	Day	Day	Night	Evening
Darby	Evening	Evening	Day	Day	Evening
Derry	Night	Day	Evening	Day	Evening
Eileen	Day	Day	Day	Day	Day
Green	Day	Day	Day	Day	Day
Jason	Evening	Night	Day	Evening	Day
Katie	Evening	Night	Day	Night	Day
Lloyd	Day	Evening	Day	Evening	Night
Matrix	Day	Day	Evening	Evening	Day
Niall	Day	Day	Day	Day	Evening
Paula	Evening	Evening	Evening	Evening	Day

(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

Taillard Job-Shop Benchmarks



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
15/15	10	90.00	0.00	0.00	10.00	105.19	141.18	100.00	100.00	97.17
20/15	10	20.00	0.00	0.00	80.00	267.27	263.20	100.99	100.05	98.50
20/20	10	0.00	0.00	0.00	100.00	n/a	n/a	100.74	100.06	97.96
30/15	10	10.00	0.00	10.00	80.00	174.32	100.00	100.18	100.49	99.87
30/20	10	0.00	0.00	0.00	100.00	n/a	n/a	100.30	101.30	99.40
50/15	10	100.00	0.00	0.00	0.00	100.00	685.09	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
100/20	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	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

Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Some Suggested Experiments

Summary

ENTIRE EDIH

Production Scheduling

Slide 16

Input Data

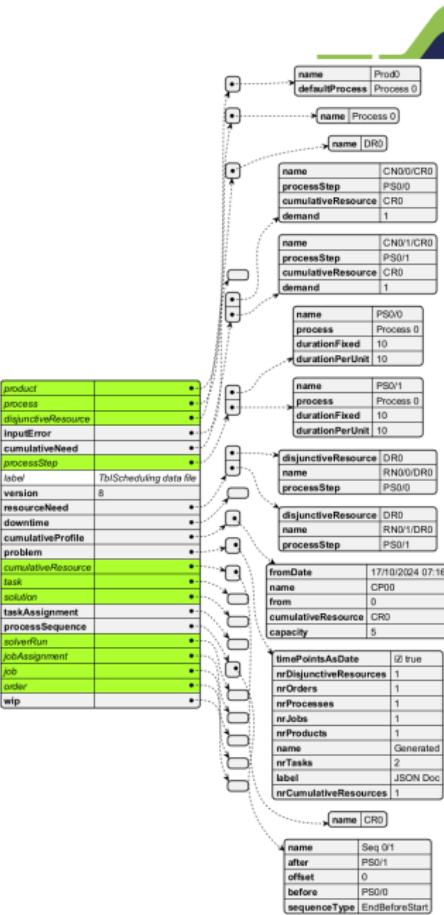


- We have defined a specific JSON data format to describe scheduling problems
- This is different from the native/XML data format of the application (do not use)
- Load with menu **File** – Load DataFile...
- Save with menu **File** – Save DataFile...
- The format is described in a document

Base Data



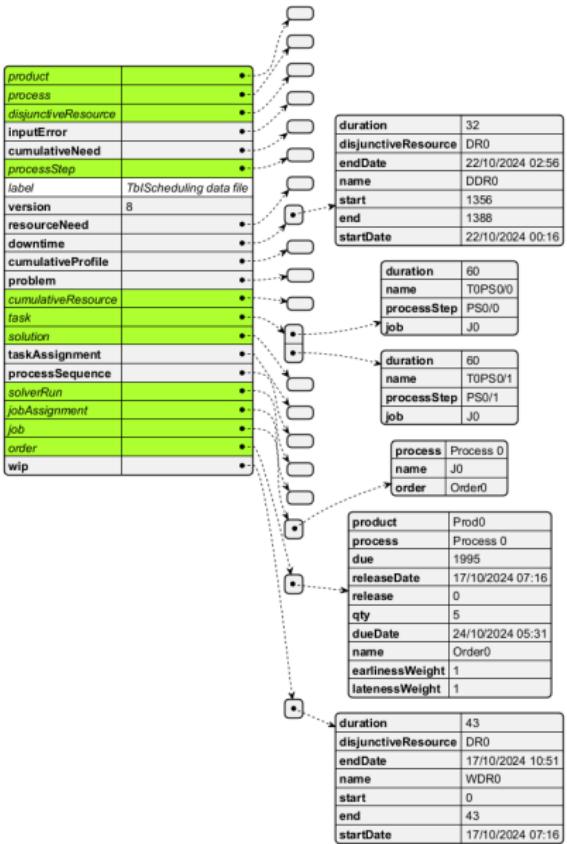
- Description of
 - Product
 - Process
 - DisjunctiveResource
 - CumulativeNeed
 - ProcessStep
 - ResourceNeed
 - CumulativeProfile
 - Problem
 - CumulativeResource
 - ProcessSequence



Schedule Input Data



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



Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Some Suggested Experiments

Summary

ENTIRE EDIH

Production Scheduling

Slide 20

Result Data



- 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

makeSpan	163
totalEarliness	1632
totalLate	0
totalLateBefore	0
early	1
percentEarly	100
earliest	163
earliestLateness	0
maxWaitAfter	0
idle	0
idleDuration	0
idleEarliness	163
idleLate	0
idleLateBefore	0
maxWaitBefore	0
best	163
solverStatus	Optimal
use	0
totalWaitAfter	0
percentLate	0
weightEarliness	0
maxEarliness	163
range	Soft
skipValue	163
timeout	163
yearDate	17102024 1051

problem	*
process	*
processAssignment	*
representer	*
cumulativeNeed	*
duration	*
earliestDeadline	*
earliestEarliness	*
earliestLate	*
earliestLateBefore	*
earliestWaitAfter	*
earliestWaitBefore	*
earliestWaitDuration	*
earliestWaitEarliness	*
earliestWaitLate	*
earliestWaitLateBefore	*
earliestWaitWaitAfter	*
earliestWaitWaitBefore	*
earliestWaitWaitDuration	*
earliestWaitWaitEarliness	*
earliestWaitWaitLate	*
earliestWaitWaitLateBefore	*
earliestWaitWaitWaitAfter	*
earliestWaitWaitWaitBefore	*
earliestWaitWaitWaitDuration	*
earliestWaitWaitWaitEarliness	*
earliestWaitWaitWaitLate	*
earliestWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitAfter	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitBefore	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitDuration	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitEarliness	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLate	*
earliestWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitWaitLateBefore	*
earliestWaitAfter	*
earliestWaitBefore	*
earliestWaitDuration	*
earliestWaitEarliness	*
earliestWaitLate	*
earliestWaitLateBefore	*
earliestWaitAfter	*
earliestWaitBefore	*
earliestWaitDuration	*
earliestWaitEarliness	*
earliestWaitLate	*
earliestWaitLateBefore	*
earliestWaitAfter	*
earliestWaitBefore	*
earliestWaitDuration	*
earliestWaitEarliness	*
earliestWaitLate	*
earliestWaitLateBefore	*
earliestWaitAfter	*
earliestWaitBefore	*
earliestWaitDuration	*
earliestWaitEarliness	*
earliestWait	*
duration	80
waiter	0
waitDuration	0
diagnoseResources	DPO
endDate	17102024 15:31
name	JAD
start	103
jobAssignment	JAD
waiterhere	0
workDate	17102024 15:31
duration	80
waiter	0
waitDuration	0
diagnoseResources	DPO
endDate	17102024 23:31
name	JAD
start	103
jobAssignment	JAD
waiterhere	0
workDate	17102024 23:31
nrThreads	2
weightEarliness	1
solverBackend	None
weightLate	1
solverStatus	Optimal
description	
label	
rootType	CPO
referencing	0
referencedBy	0
objectiveType	MakeSpan
deadline	30
nrOfJobs	1
nrOfCumulative	0
name	Runt
weightEarliness	1
start	0
jobAssignment	None
enforceDeadline	0
enforceDuration	0
enforceEarliness	0
enforceLate	0
enforceLateBefore	0
duration	103
addition	0
rate	0
endDate	17102024 30:51
name	JAD
start	103
end	103
job	JAD
early	1032
starDate	17102024 10:51

Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Some Suggested Experiments

Summary

ENTIRE EDIH

Production Scheduling

Slide 23

Instance Generator

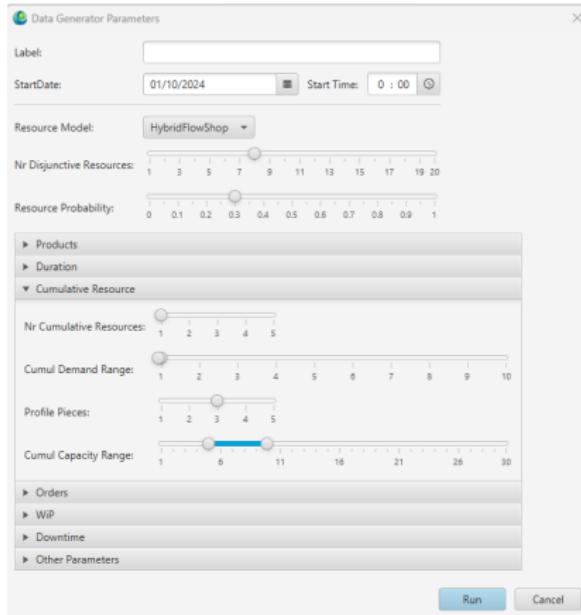


- Application allows to generate different types of test problems
- Different types of resource models
- Different numbers of orders, resources, WiP, downtime
- Useful to generate more life-like examples combining different constraint types

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



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

Data Generator Parameters

Label:

Start Date: 01/10/2024 Start Time: 0 : 00

Resource Model: HybridFlowShop

Nr Disjunctive Resources:

Resource Probability:

Products

Nr Products:

Stages Range:

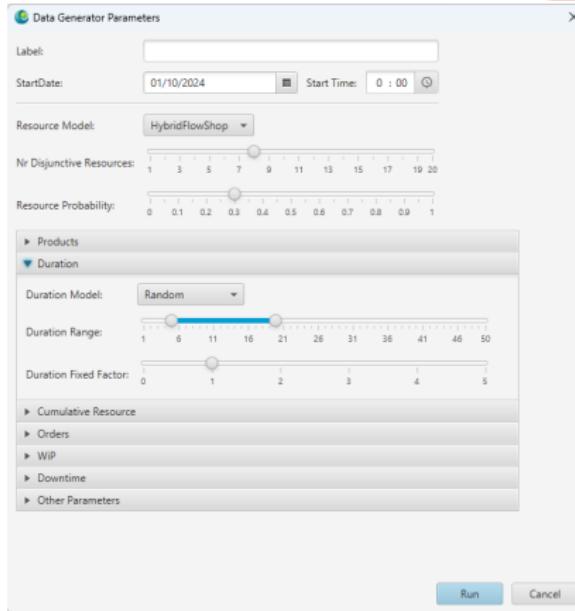
Duration
 Cumulative Resource
 Orders
 WIP
 Downtime
 Other Parameters

Run Cancel

Instance Generator - Duration



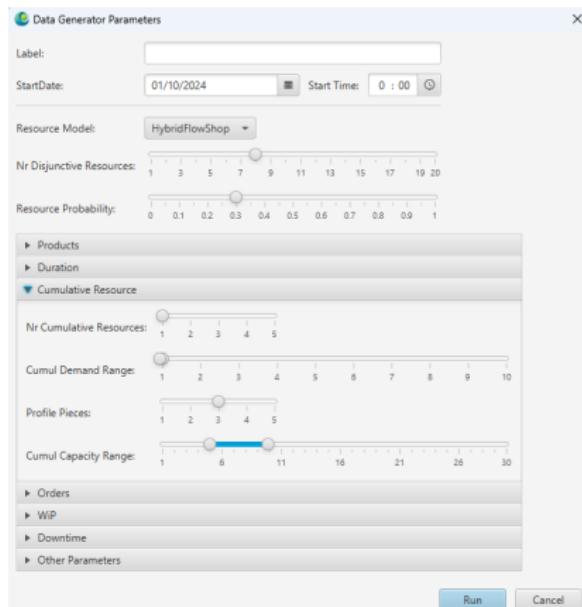
- Duration Model
 - Different ways to link duration of processSteps
- Duration Range
 - Range slider to set lower and upper bounds on perUnit duration
- Duration Fixed Factor
 - How fixed and perUnit duration values are linked



Instance Generator - Cumulative



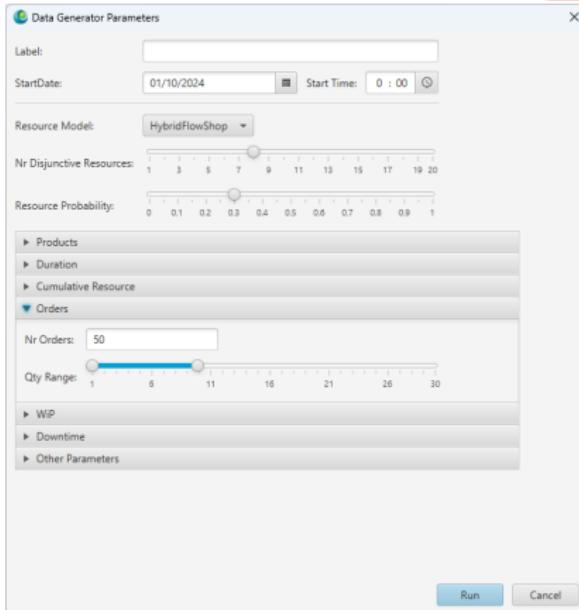
- Nr Cumulative Resources
 - Number of cumulative resources generated
- Cumul Demand Range
 - Range slider to select lower and upper bound on cumulativeResource-Need demands
- Profile Pieces
 - Number of segments of CumulativeProfile generated for each resource
- Cumul Capacity Range
 - Range slider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



Instance Generator - Orders



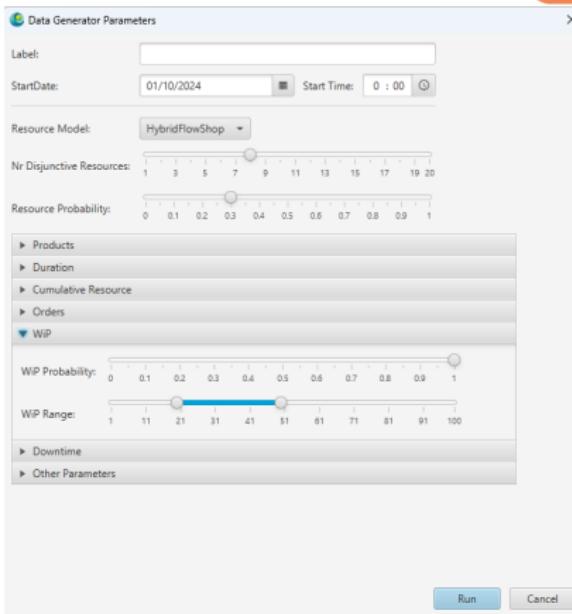
- Nr Orders
 - Number of orders generated, each order is assigned a random product/process
- Qty Range
 - Range slider to select lower and upper bounds on quantity for each order



Instance Generator - WiP (Work in Progress)



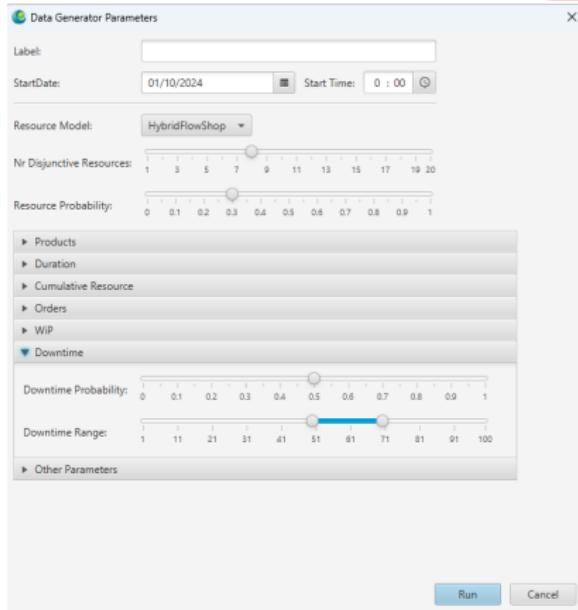
- WiP Probability
 - Probability of generating a WiP for a disjunctive resource
- WiP Range
 - Range slider to set lower and upper bound on WiP duration



Instance Generator - Downtime



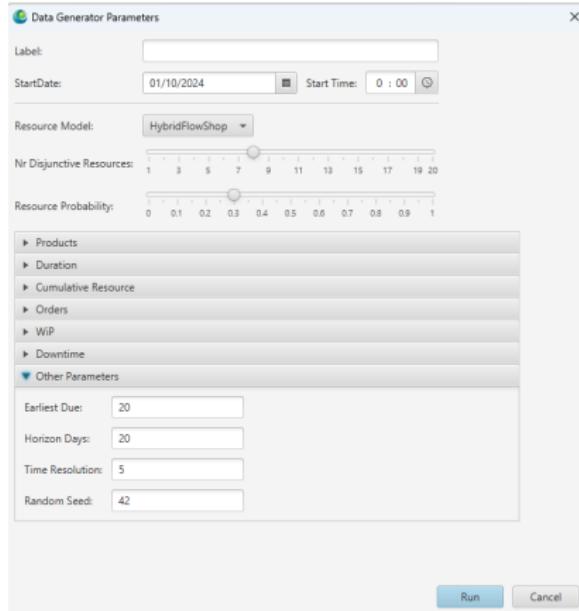
- Downtime Probability
 - Probability of generating a downtime for a disjunctive resource
- Downtime Range
 - Range slider to select lower and upper bounds on downtime duration



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



Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Taillard

SALBP

Test Scheduling

Taillard Scheduling Benchmarks



- 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...

Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Some Suggested Experiments

Summary

ENTIRE EDIH

Production Scheduling

Slide 38

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

Outline



The Scheduling Tool

Under the hood

Input Data

Result Output

Instance Generator

Predefined Problem Sets

Some Suggested Experiments

Summary

ENTIRE EDIH

Production Scheduling

Slide 44

Summary



- 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