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

Key Points

- This section describes the scheduling tool
- 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

- We create the tool as basis for experiments
- To test ideas and solvers
- As a teaching tool
- Slightly higher standard than usual academic prototypes
- Not a commercial tool
  - But can use commercial solvers
  - Also open-source solvers
- Written in Java, JavaFX
- Uses our Java application framework generator
- Will become available in early 2025

Outline

The Scheduling Tool

Under the hood
    Google CPSat
    IBM CPOptimizer
    MiniZinc
    Which solver to use?

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 3 units of time. The second, (1, 2), must be processed on machine 1 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)

Outline

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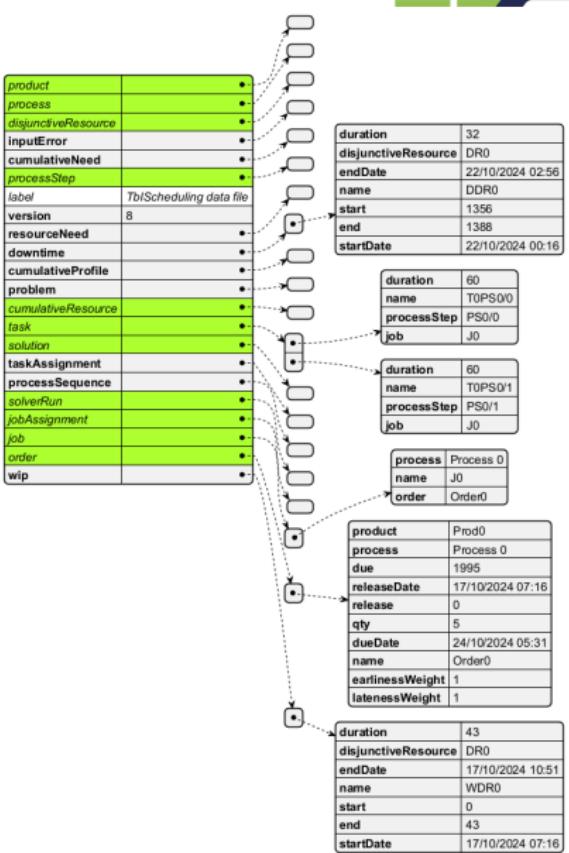
- Description of
  - Product
  - Process
  - DisjunctiveResource
  - CumulativeNeed
  - ProcessStep
  - ResourceNeed
  - CumulativeProfile
  - Problem
  - CumulativeResource
  - ProcessSequence

Schedule Input Data

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

product	•
process	•
disjunctiveResource	•
inputError	•
cumulativeNeed	•
processStep	•
label	TblScheduling data file
version	8
resourceNeed	•
downtime	•
cumulativeProfile	•
problem	•
cumulativeResource	•
task	•
solution	•
taskAssignment	•
processSequence	•
solverRun	•
jobAssignment	•
job	•
order	•
wip	•

duration	32
disjunctiveResource	DR0
endDate	22/10/2024 02:56
name	DDR0
start	1356
end	1388
startDate	22/10/2024 00:16

duration	60
name	T0PS0/0
processStep	PS0/0
job	J0

duration	60
name	T0PS0/1
processStep	PS0/1
job	J0

process	Process 0
name	J0
order	Order0

product	Prod0
process	Process 0
due	1995
releaseDate	17/10/2024 07:16
release	0
qty	5
dueDate	24/10/2024 05:31
name	Order0
earlinessWeight	1
latenessWeight	1

duration	43
disjunctiveResource	DR0
endDate	17/10/2024 10:51
name	WDR0
start	0
end	43
startDate	17/10/2024 07:16

Outline

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

Outline

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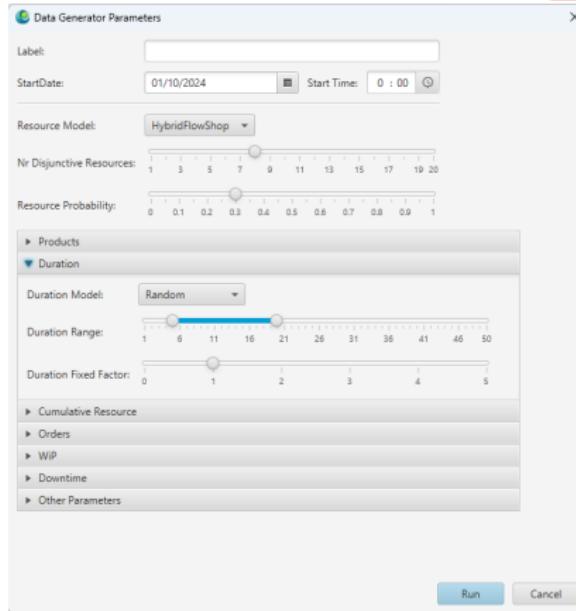
- Nr Products
  - Number of products to be generated
  - Products may be reused by multiple orders
- Stages Range
  - Range slider, sets lower and upper bound on number of stages

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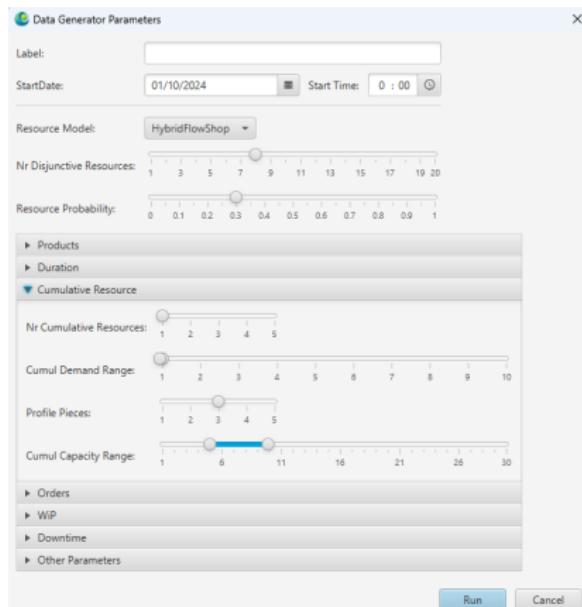
- Duration Model
  - Different ways to link duration of processSteps
- Duration Range
  - Range slider to set lower and upper bounds on perUnit duration
- Duration Fixed Factor
  - How fixed and perUnit duration values are linked

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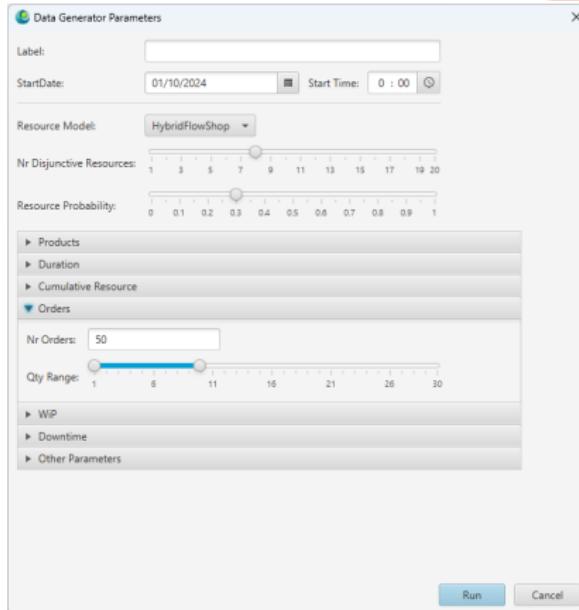
- Nr Cumulative Resources
  - Number of cumulative resources generated
- Cumul Demand Range
  - Range slider to select lower and upper bound on cumulativeResource-Need demands
- Profile Pieces
  - Number of segments of CumulativeProfile generated for each resource
- Cumul Capacity Range
  - Range slider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

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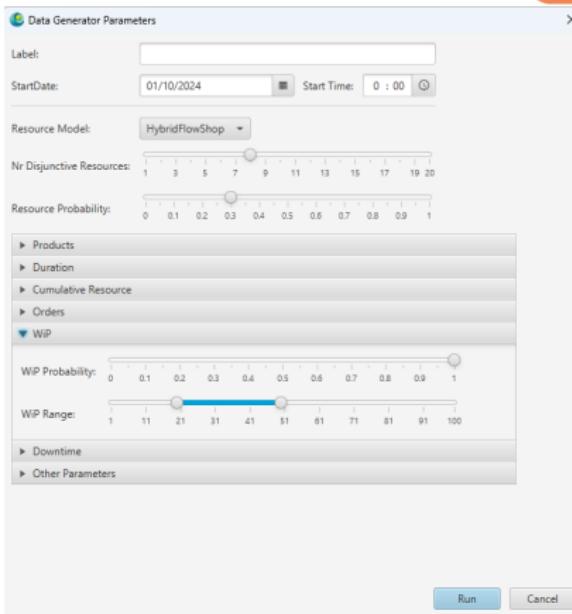
- Nr Orders
  - Number of orders generated, each order is assigned a random product/process
- Qty Range
  - Range slider to select lower and upper bounds on quantity for each order

Instance Generator - WiP (Work in Progress)
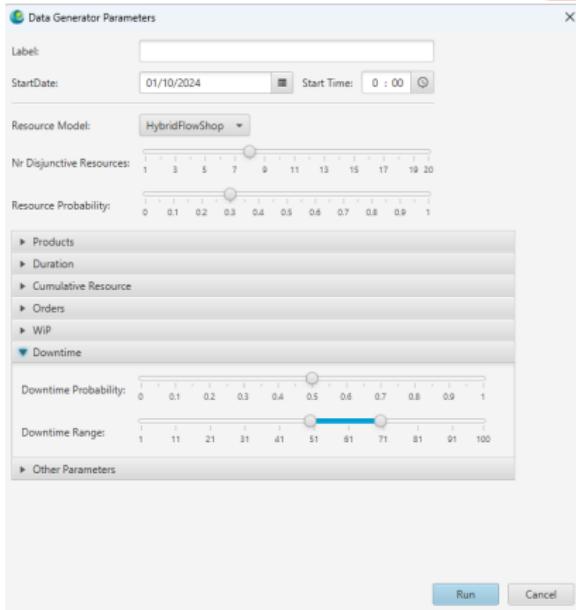
- WiP Probability
  - Probability of generating a WiP for a disjunctive resource
- WiP Range
  - Range slider to set lower and upper bound on WiP duration

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

Some Suggested Experiments

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

Production Scheduling

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`

Outline

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