Assembly Line Balancing Case Study

Helmut Simonis

email: helmut.simonis@insight-centre.org
homepage: http://http://insight-centre.org/

ENTIRE EDIH
Insight SFI Centre for Data Analytics
School of Computer Science and Information Technology
University College Cork
Ireland

Constraint Based Production Scheduling

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

- There can be more than one formulation of a problem
- Typically there is a direct model where all constraints are taken from problem description
- There may be other representations of problem using other variables, domains, constraints
- Needs mapping to original problem
- Performance may vary a lot

Problem Description

A car is made on an assembly line, where the n pieces of the car are added at different stations. The car moves along the assembly line, running through the stations in sequence, where it stays for a fixed amount of time (the *cycle time* or *Takt* t) at each station. At each station different pieces can be added, each item i requiring a certain amount of time d_i. The total amount of time spent in each station cannot be more than the cycle time. A precedence graph states which items must be installed before another item can be added. All items must be placed on the car in some station, the precedence constraints state that all preceding items must be placed on the car at an earlier or the same station. The problem is to design the assembly line, and minimize k, the number of stations needed. The stations should be numbered consecutively from 1 to k.

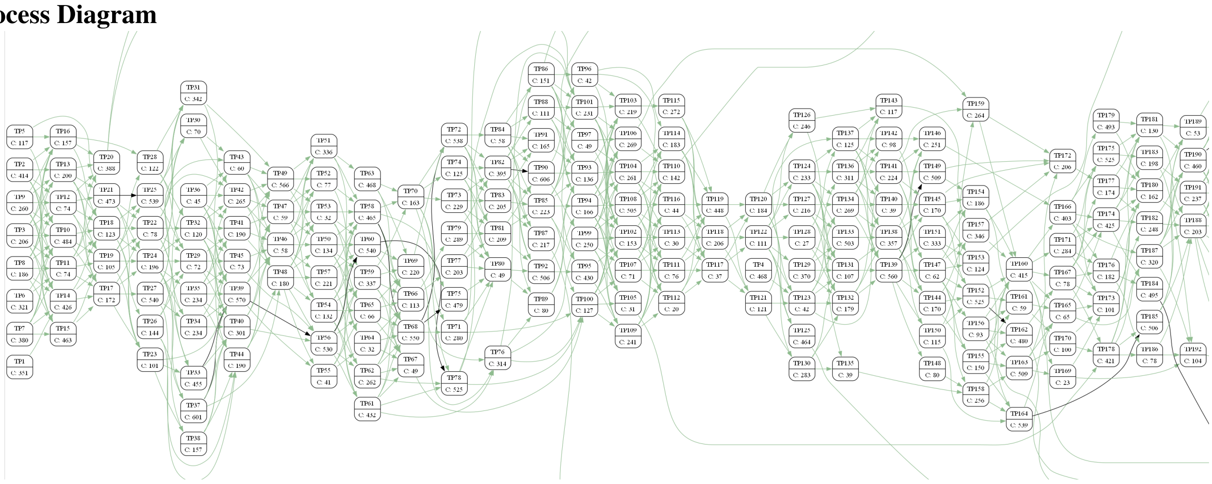
Feature Overview

- Direct model
  - One task for each item to be placed, with a variable indicating the station it is added
  - Domain: possible stations, all tasks have duration one
  - Tasks with StartBeforeStart time constraints
  - Can be strengthened to EndBeforeStart for certain task pairs
  - One cumulative constraint with overall Takt resource capacity t
  - CumulativeNeed of each Task is its duration d_i in seconds
- Other models possible

Large Scale Example

- Problem `instance_n=1000\_511.alb`
- 1000 tasks, complex precedence graph
- Cumulative lower bound 230
  - Sum of CumulativeNeed 229,447
  - Takt 1,000

Process Diagram

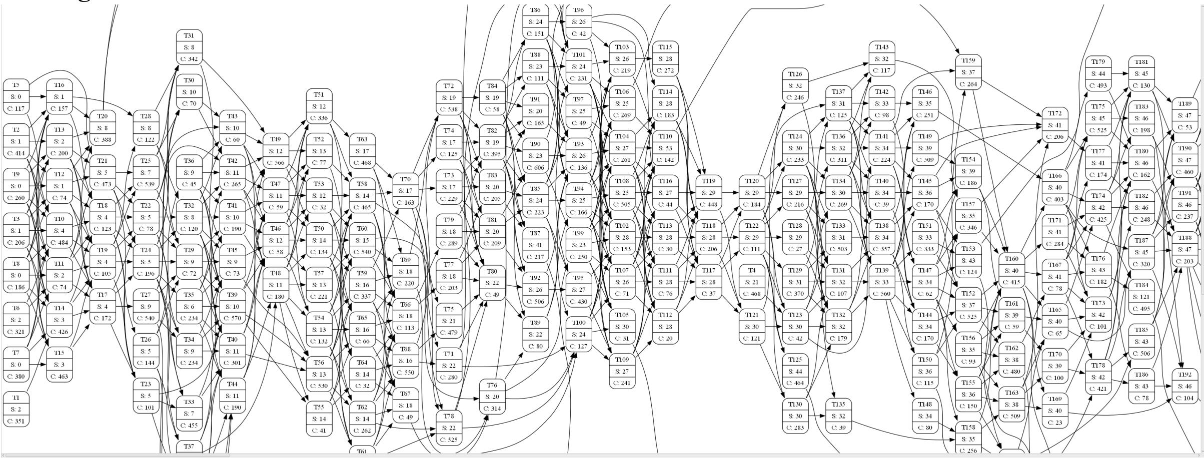


- Only part of overall process shown
- StartBeforeStart links shown in green
- Derived EndBeforeStart links in black

Solutions

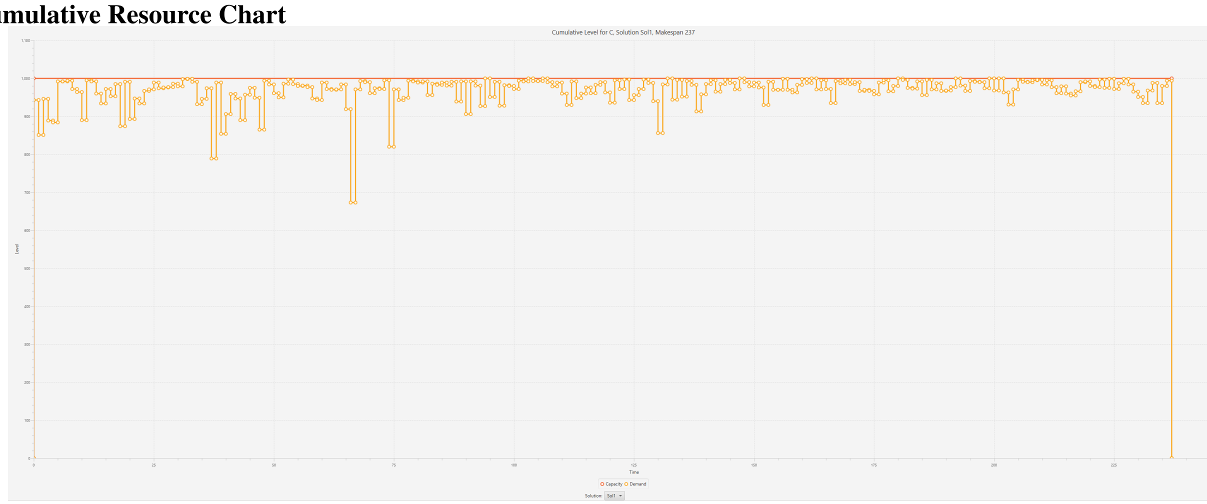
- CPO
  - 300s timeout, 4 threads
  - Solver lower bound 230
  - Cost 237 (Gap 2.95%) after 30s
- CPSat
  - 300s timeout, 8 threads
  - Solver lower bound 224
  - Cost 237 (Gap 2.95%) after 22s

PERT Diagram of Solution



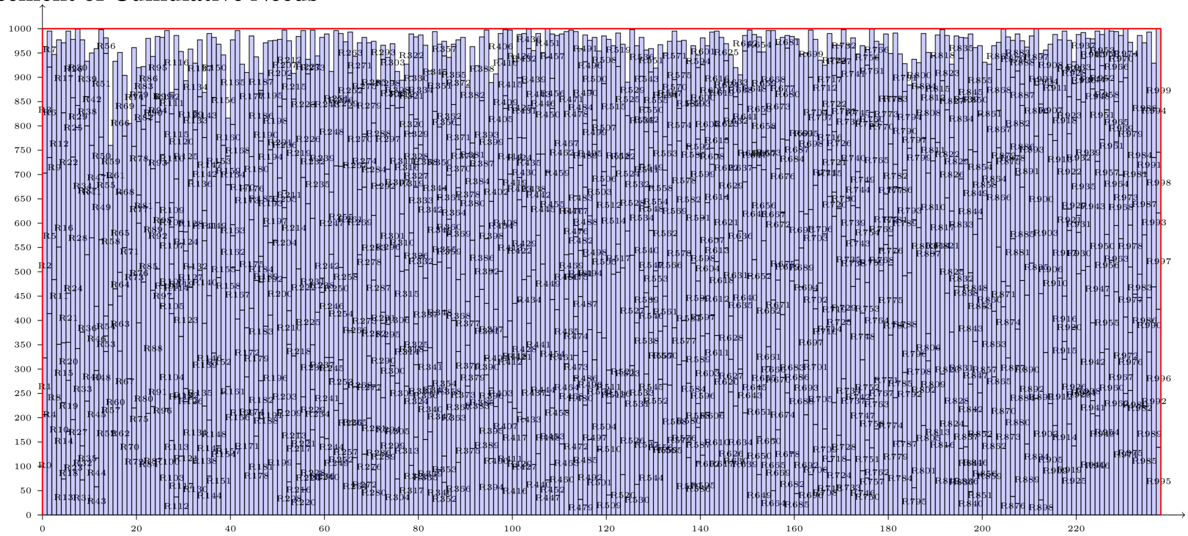
- Only part of diagram shown

Cumulative Resource Chart



- Overall resource use quite balanced
- Some stations have spare capacity

Placement of Cumulative Needs



- Not the same solution
- Task height is high compared to station capacity

Overview of Benchmark Results

		All Instances				Optimal Only		Non Optimal Only			
		Optimal (% of All Instances)				Time (% of VB)		Cost (% of VB)		Bound (% of VB)	
Group	Nr	Both	CPO	CPSat	None	CPO	CPSat	CPO	CPSat	CPO	CPSat
20	525	99.62	0.00	0.38	0.00	1363.95	118.02	100.00	100.00	82.14	100.00
50	525	72.38	0.38	21.71	5.52	330.66	2609.56	100.22	100.05	92.69	100.00
100	525	57.14	0.57	11.43	30.86	177.94	499.89	101.17	100.04	94.77	100.00
1000	525	0.00	0.00	0.00	100.00	n/a	n/a	100.05	101.07	100.00	73.86

- Benchmark set from Otto et al, EJOR, 2013
- 30s timeout, 4 threads for CPO, 8 threads for CPSat
- Small instances solved to optimality
- Larger instances have good, but not optimal solutions for both CPO and CPSat
- Weaker lower bound for CPSat on large instances

Alternative Model

- One task per item to be placed, with a variable indicating the start time within the station
- Domain: k*(t+1), tasks have duration d_i
- Place *Exclusion markers* in timeline to avoid tasks spanning multiple stations
  - Place fixed Downtime at end of each station period
  - Start $t + i * (t + 1)$
  - Duration one
  - Example for t=1000 and start=0: 1000, 2001, 3002, 4003 ...
- EndBeforeStart precedence constraints between tasks
  - Preceding tasks are either in the same station (ordered in time), or in an earlier station
- One disjunctive resource: maximum time available in each station is t

Alternative Model (II)

- Objective
  - Minimize makespan (latest end) of tasks
  - Ignore downtime objects in objective
- Needs a mapping to find station of each item
  - $station = \lceil start/(t+1) \rceil$
- Needs an upper bound on the number of stations to create the correct number of downtime objects
  - Trivial: n

Summary

- Scheduling can be used to design factories as well
- There can be more than one model for the same problem
- Choosing the best model is quite difficult
  - Experiments with realistic data are key
- Good results compared to specialized methods