Oven Scheduling 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

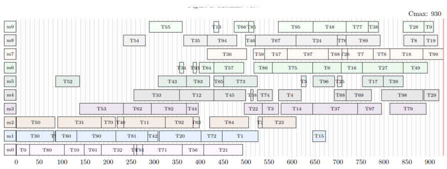
- Discusses two topics:
  - Solve a very specific industrial scheduling problem from the ASSISTANT EU project
  - Discuss the general issue of short-term scheduling vs. long-term objectives

Research Challenge

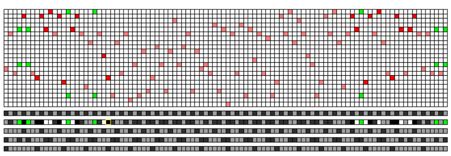
- Often the long-term business objectives are not visible in the operational decision problem
- We optimize a short-term objective without understanding the impact in the long term
- What choices should we make in short-term to improve overall result?
- Especially important when future data not yet visible
- Surprisingly, this problem is rarely discussed in literature

Examples

- Production Scheduling
- Nearly all scheduling benchmarks use c_{max} (makespan) as objective
- Why?
- Do we want to close factory as rapidly as possible?

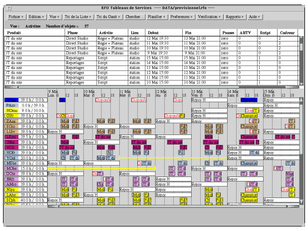


- Car Sequencing
- The best heuristics push difficult cars to the edge of schedule
- Because they are easier to schedule this way
- But: It makes it hard to schedule next day



Examples

- Personnel Rostering
- Satisfy working rules and demands for period
- But: rules apply on a rolling horizon
- Easy to over-constrain problem for next period



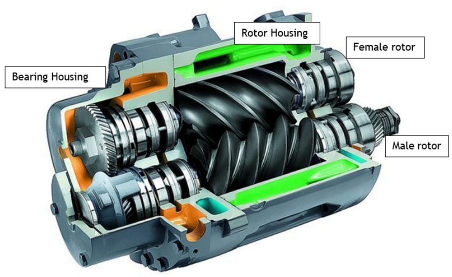
- Transportation Planning
- Build daily delivery tours, optimizing cost
- Where are your trucks at 10PM?
- Also, avoid cherry-picking at start of week

Problem Studied Here

- Example from the ASSISTANT EU project (ended last year)
- Oven schedule for one of the industrial partners
- Schedule tasks on a set of ovens
- Tasks can share oven only if they are compatible
- Conflicting objectives
  - Energy use of ovens very significant, reduce when ovens are used
  - Waiting for an oven affects quality of product
- Jobs only visible when previous process step starts
- Currently scheduled by hand, industry partner expressed strong need for change

What does this look like in the real world?

Industrial Oven



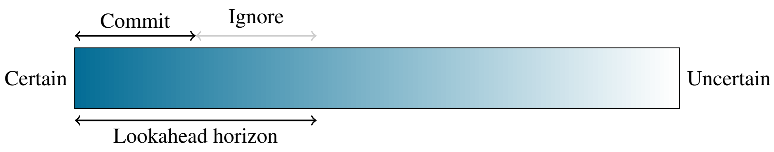
Rotors in Compressor

Solution Approach: Constraint Programming

- Declarative modelling approach for combinatorial problems
  - Problem expressed in terms of variables and constraints
- Global constraints
  - Combines expressive modelling abstractions and powerful reasoning
  - Examples: disjunctive, cumulative, global_cardinality
- Compositional: Add constraints as required
- Main application areas
  - Scheduling, rostering, transportation
  - Also: test generation, verification, configuration

Overall Decomposition (Standard)

- We can only see that far into future
- We do not want to take decisions now that we might regret later
- We have to make some decisions now otherwise we never do anything
- *Rolling horizon* decomposition
  - We schedule up to *lookahead horizon* units into the future
  - We commit to implement resulting schedule only to up *commitHorizon*
  - We reschedule when we receive new information, or we reach the end of commitment
  - We solve each short-term sub problem based on short-term objectives



Short-Term Schedule Modelling

- Challenge: There is no global constraint to express the oven resource constraint
- We are not able to invest a lot of time/resources to develop such a constraint
- Two choices:
  - Two traditional models with variables linking them (Lackner et al, Constraints 2023)
  - Direct model expressing conditions as disjunctions of basic constraints

The Standard Pieces

- Jobs N consisting of multiple stages Q, tasks for each stage of each job, running on machines M
- Release dates r_i of jobs given by up-stream schedule
- WiP w_k on certain machines resulting from earlier schedule
- Machine m_{ij} and start variables s_{ij} for each task
- Precedence constraints between tasks of each jobs, with total waiting time c_i when waiting for resource
- Total number of ovens used in schedule *nrOvens* by *nvalue* constraint

$$\text{nvalue}(\text{nrOvens}, [m_{ij} | i \in N, j \in Q] ++ [k | k \in M \text{ s.t. } w_k > 0])$$

Resource Constraints

We start from the basic decomposition of thedisjunctive machine choice constraint

$$\begin{aligned} \forall_{i_1,i_2 \in N} \forall_{j_1,j_2 \in Q \text{ s.t. } <i_1,j_1> \neq <i_2,j_2>} : \quad m_{i_1 j_1} \neq m_{i_2 j_2} \vee \\ s_{i_1 j_1} \geq s_{i_2 j_2} + d_{i_2 j_2} \vee \\ s_{i_2 j_2} \geq s_{i_1 j_1} + d_{i_1 j_1} \end{aligned}$$

Express case where tasks share an oven (only when types and stages are the same)

$$\begin{aligned} \forall_{i_1,i_2 \in N \text{ s.t. } i_1 \neq i_2} \forall_{j \in Q} : \quad m_{i_1 j} \neq m_{i_2 j} \vee \\ s_{i_1 j} \geq s_{i_2 j} + d_{i_2 j} \vee \\ s_{i_2 j} \geq s_{i_1 j} + d_{i_1 j} \vee \\ (t_{i_1 j_1} = t_{i_2 j_2} \wedge m_{i_1 j} = m_{i_2 j} \wedge s_{i_1 j} = s_{i_2 j}) \end{aligned}$$

Limit stacking

Need binary variables $b_{i_1 i_2 j}$ to state that two jobsi_1 and i_2 share oven in stage j

$$\begin{aligned} \forall_{i_1,i_2 \in N \text{ s.t. } i_1 < i_2} \forall_{j \in Q} : \quad (b_{i_1 i_2 j} = 0 \wedge (m_{i_1 j} \neq m_{i_2 j} \vee \\ s_{i_1 j} \geq s_{i_2 j} + d_{i_2 j} \vee \\ s_{i_2 j} \geq s_{i_1 j} + d_{i_1 j}) \vee \\ (b_{i_1 i_2 j} = 1 \wedge t_{i_1 j_1} = t_{i_2 j_2} \wedge m_{i_1 j} = m_{i_2 j} \wedge s_{i_1 j} = s_{i_2 j}) \end{aligned}$$

Count how many jobs share stage j with job i

$$\forall_{i \in N} \forall_{j \in Q}: \quad z_{ij} = \sum_{i_1=1}^{i-1} b_{i_1 ij} + \sum_{i2=i+1}^{n} b_{ii_2 j}$$

Limit how many tasks can be stacked together

$$\forall_{i \in N} \forall_{j \in Q}: \quad z_{ij} < \text{maxStacked}$$

This should not work!

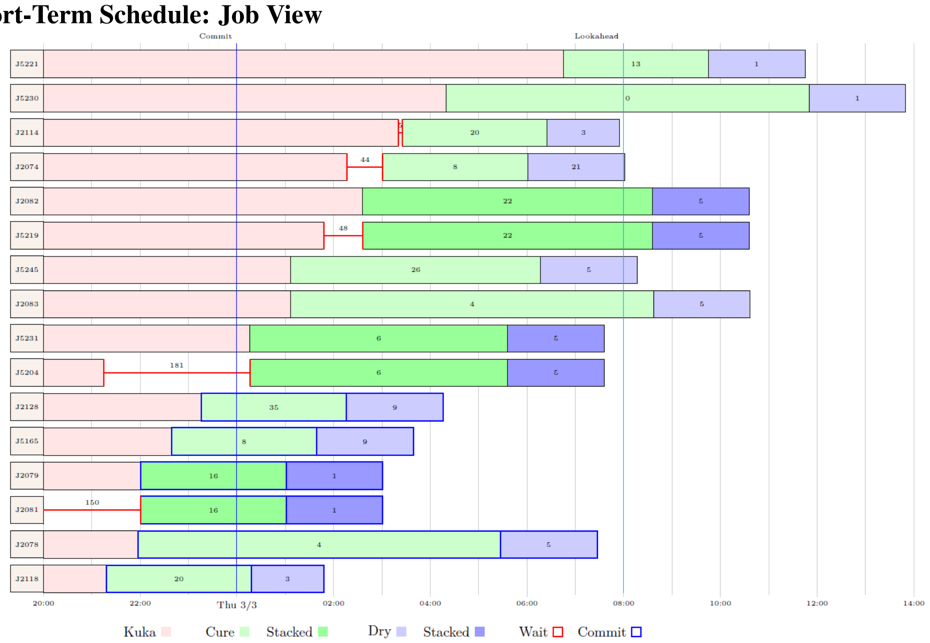
- Weakness of basic decomposition model was the reason to develop the scheduling constraints in the first place
- Does not scale well to thousands of tasks
- But model is well suited to some solvers
  - SAT based solvers, Chuffed, CP-SAT (OR-Tools)
  - MIP solvers
- This works (only) as long as problem size stays manageable

Compound Objective

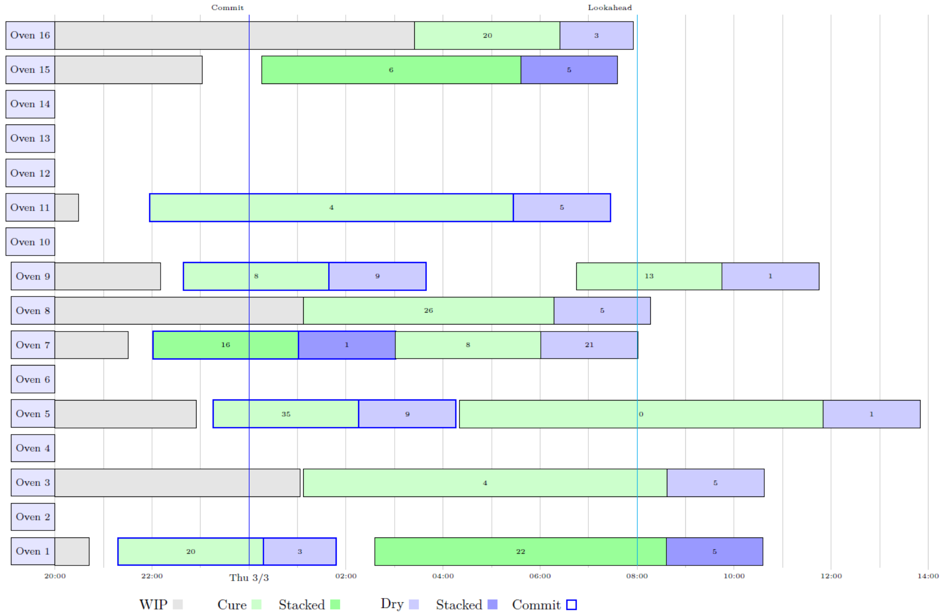
$$\min \alpha_1 \sum_{i \in N} c_i + \alpha_2 \text{nrOvens} + \alpha_3 \sum_{i \in N, j \in Q} z_{ij}$$

- Three conflicting elements
  - Total waiting time for jobs
  - Number of ovens used
  - Number of tasks stacked (negative coefficient)
- Reducing waiting time requires using more ovens
- Improved stacking will require for one job to wait until second is ready

Short-Term Schedule: Job View



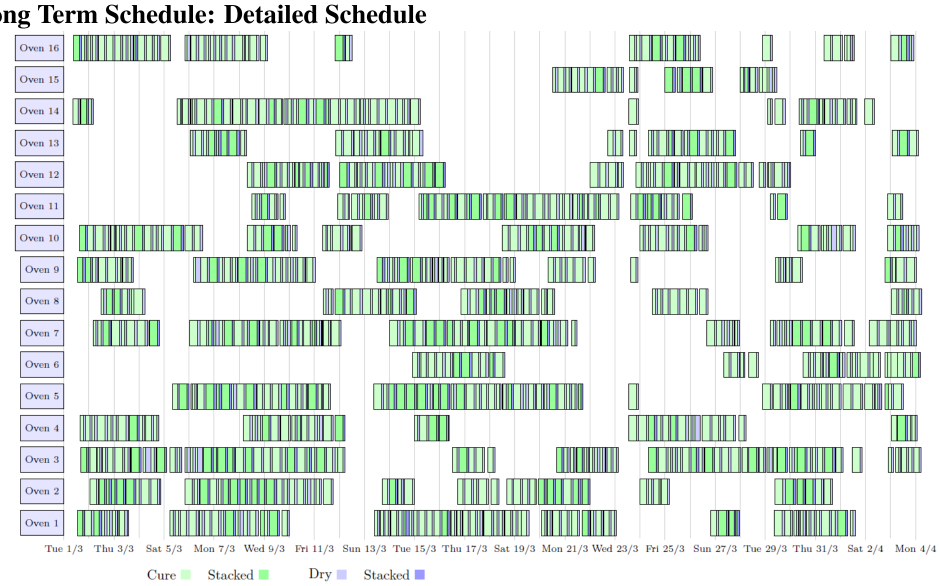
Short Term Schedule: Resource View



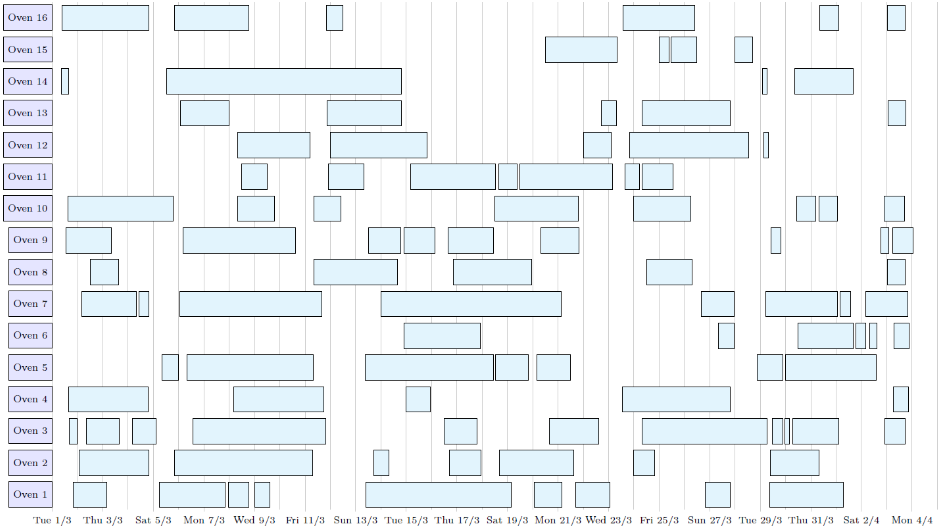
Are the short-term solutions good?

- We solve many problems to optimality, depending on solver
- Optimality gap is small, increasing search time helps a bit
- But are we optimizing the best possible objective?

Long Term Schedule: Detailed Schedule



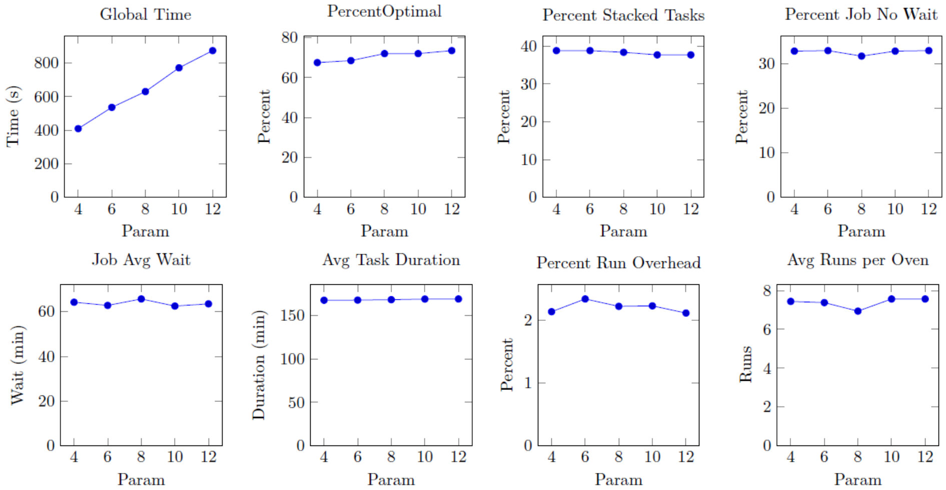
Long Term Schedule: Abstracted Oven Runs



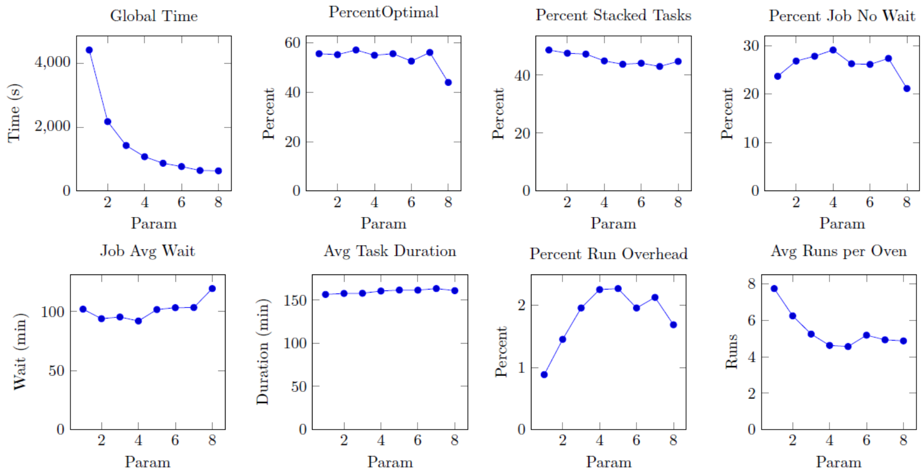
Is that a good global schedule? KPIs

Name	Unit	Explanation
Global Time	Seconds	Total time for solving all sub problems
Nr Jobs	-	Total number of jobs scheduled
Nr Tasks	-	Total number of tasks scheduled
Percent Optimal	Percentage (0-100)	How many sub problems were solved to optimality
Percent Stacked Tasks	Percentage (0-100)	Percentage of all tasks scheduled that were stacked
Percent Jobs No Wait	Percentage (0-100)	Percentage of jobs that were scheduled without any waiting time
Job Average Wait	Minutes	Average wait time over all jobs
Job Maximal Wait	Minutes	Largest waiting time for any job scheduled
Ovens Used	-	Total number of ovens used during period
Avg Task Duration	Minutes	Average tasks duration (influenced by stacking)
Oven Runs	-	Number of oven runs over total horizon
Run Overhead Percent	Percentage (0-100)	Overhead during oven runs when machine is idle
Avg Runs per Oven Used	-	Average number of oven runs per oven used

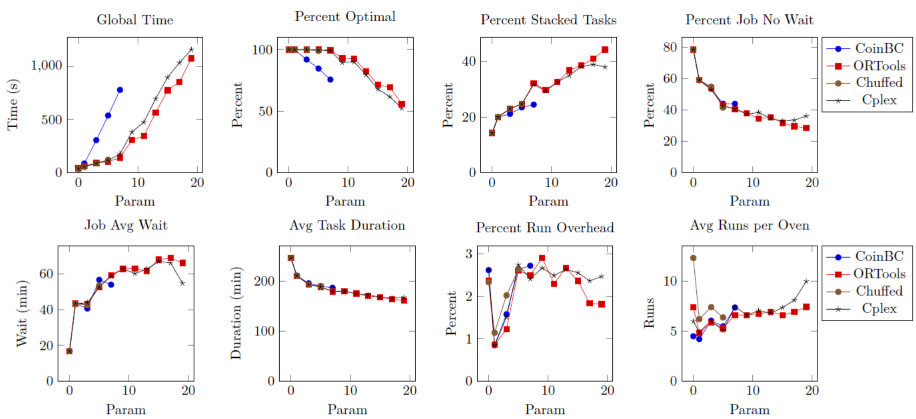
Impact of Lookahead Parameter



Impact of CommitHorizon Parameter



Comparing Different Solvers



Is the global solution really good?

- We schedule with limited information
- Hindsight is 20/20, we cannot expect best possible solution from partial information
- Process Challenge: Can we improve data visibility?
- Demand is variable over time, no steady-state solution
- Modelling Challenge: Can we define a short-term objective that produces better long-term solutions?
- Algorithm Challenge: Can we solve the global problem to optimality?
  - Assumes "a priori" visibility of data
  - This would provide a lower bound
  - But we need optimality to use as bound

Summary

- Discussed a non-standard oven scheduling problem from industry
- Models with decomposition of resource constraints
- Good/very good short-term solutions

- But is the overall schedule close to the global optimum?
- In any case, industry partner was happy with solution and analysis