MODELLING AND SOLVING SCHEDULING PROBLEMS USING CONSTRAINT PROGRAMMING

Two worlds

- planning vs. scheduling
  - planning is about finding activities to achieve given goal
  - scheduling is about allocating known activities to limited resources and time

- generic (AI) vs. specific (OR) approaches
  - flexible techniques but bad worst-case runtime (due to search)
  - guaranteed runtime and schedule quality, but inflexible techniques

- theory vs. practice
Talk outline

- Motivation
  - scheduling in practice and in academia
- Constraint programming
  - principles and application in scheduling
- Scheduling model
  - temporal network with alternatives
- System demo
  - FlowOpt project
- Concluding remarks

What you will hear in factory

- “We are different...”
  - means, what you know is useless here
- “ Outsiders cannot understand it, it takes a lot of time...”
  - means, you have to listen to us or to spend part of your life here
- “Methods that suite others cannot implemented here...”
  - means, your experience and knowledge are impressive, but you have to start from scratch
Theory vs. practice

- **Academy**
  - the researcher’s world consists of resources and their usage
    - “how can I use the resources to get max X and min Y...”
    - “how can I get, using objective metrics, a plan that for the long term, will improve the plant efficiency...”

- **Factory planners**
  - the planner’s world consists of products and their flow
    - “how can I produce this product now, and this one and that one...”
    - “how can I satisfy Mr. X from sales and Mr. Y from the plant and the customer at the same time, without getting into new troubles...”

Our approach

- Be close to the customer
  - use notions that factory planners are familiar with

- Translate the problem to solving formalism
  - use flexible modelling and solving approach

- Solve the problem to acceptable quality
  - combine heuristics and inference

- Allow customers to modify the solution
  - support interactive changes of solutions
What is CP?

**Constraint Programming** is a technology for solving combinatorial optimization problems modeled as constraint satisfaction problems:
- a finite set of decision **variables**
- each variable has a finite set of possible values (**domain**)
- combinations of allowed values are restricted by **constraints** (relations between variables)

**Solution** to a CSP is a complete consistent instantiation of variables.

How does CP work?

How to find a solution to a CSP?

Mainstream **solving approach** combines

- **inference**
  - removing values violating constraints
  - consistency techniques

- **search**
  - trying combinations of values
  - depth-first search
Constraint Inference

Example:
- $D_a = \{1, 2\}$, $D_b = \{x, 2, 3\}$
- $a < b$

▷ Value 1 can be safely removed from $D_b$.

- Constraints are used **actively to remove inconsistencies** from the problem.
  - inconsistency = a value that cannot be in any solution
- The most widely-used technique removes values that violate any constraint until a fixed point is reached (no value violates a single constraints).

Search / Labeling

Consistency techniques are (usually) incomplete.
▷ We need a search algorithm to resolve the rest!

Labeling
- depth-first search
  - assign a value to the variable
  - propagate = make the problem locally consistent
  - backtrack upon failure
- $X$ in \[1...5\] \approx X=1 \lor X=2 \lor X=3 \lor X=4 \lor X=5 \quad \text{(enumeration)}$

In general, search algorithm resolves remaining disjunctions!
- $X=1 \lor X \neq 1$ \quad \text{(step labeling)}
- $X<3 \lor X \geq 3$ \quad \text{(domain splitting)}
- $X<Y \lor X \geq Y$ \quad \text{(problem splitting)}
How to use CP?

- Using Constraint Programming is less about solving algorithms and more about modeling (similarly to SAT or MIP)
  - constraint modeling = formulation of problem as a CSP
- Moreover, CP directly supports integration of ad-hoc solving techniques via global constraints and natural expression of search heuristics (differently from SAT and MIP).

ABC of CBS

Constraint-based scheduling = Scheduling + Constraint Satisfaction

Variables
- a position of activity in time and space
  - time allocation: start(A), p(A), end(A)
  - resource allocation: resource(A)

Constraints
- Temporal relations:
  - start(A) + p(A) = end(A)
  - precedences A«B: end(A) ≤ start(B)
- Resource relations:
  - unary resource A«B ∨ B«A: end(A) ≤ start(B) ∨ end(B) ≤ start(A)
Edge finding
resource inference

- Can we restrict time windows more than using disjunctive constraints?

\[
p(\Omega \cup \{A\}) > lct(\Omega \cup \{A\}) - \text{est}(\Omega) \Rightarrow A \ll \Omega \\
A \ll \Omega \Rightarrow \text{end}(A) \leq \min\{lct(\Omega') - p(\Omega') \mid \Omega' \subseteq \Omega\}
\]

In practice:
- there are \(O(n \cdot 2^n)\) pairs \((A, \Omega)\) to consider (too many!)
- instead of \(\Omega\) use so called task intervals \([X, Y]\)
  \(\{C \mid \text{est}(X) \leq \text{est}(C) \land lct(C) \leq lct(Y)\}\)
  \(\bowtie\) time complexity \(O(n^3)\), frequently used incremental algorithm
- there are also \(O(n^2)\) and \(O(n \cdot \log n)\) algorithms

---

Our problem

- Real-life production scheduling with alternative process routes and earliness/tardiness cost.
- Involves planning (selection among alternative processes) and scheduling (time and resource allocation).
Conceptual Model

- We model the workflow as a directed acyclic graph called **Temporal network with alternatives (TNA)**: nodes = operations, arcs = precedence (temporal) relations logical dependencies between nodes – **branching relations**.
  - The process can split into **parallel branches**, i.e., the nodes on parallel branches are processed in parallel (all must be included).
  - The process can select among **alternative branches**, i.e., nodes of exactly one branch are only processed (only one branch is included).
  - The **problem** is to select a sub-graph satisfying logical, temporal, and resource constraints.

Problem hardness

- If all nodes are made invalid (removed from the graph) then we have a **trivial solution** satisfying all the constraints.
  - Assume that **some node must be valid**, i.e., it is specified to be included in TNA.
    - For example, a demand must be fulfilled
  - Is it hard to find if it is possible to select a sub-graph satisfying the branching constraints?
    - Is it possible to select a process satisfying the demand?
    - The problem is **NP-complete!!!** [FLAIRS 2007].
Real processes

- Real manufacturing process networks frequently have a **specific structure**.
  - The process network is built by decomposing a "meta-processes" into more specific processes:
    - serial decomposition
    - parallel/alternative decomposition

Nested graphs

- graphs constructed from a single arc by the following **decomposition operation**:

  ![Diagram](image)

- **Features:**
  - it is a temporal network with alternatives
  - we can algorithmically recognize nested graphs
  - the assignment problem is tractable
Logical constraints

- The path selection problem can be modeled as a constraint satisfaction problem.

  - Each node $A$ is annotated by $\{0,1\}$ variable $V_A$
  - Each arc $(A,B)$ from a parallel branching defines the constraint $V_A = V_B$
  - Let arc $(A,B_1),..., (A,B_k)$ be all arcs from some alternative branching, then $V_A = \sum_{i=1}^{k} V_{Bi}$

Temporal constraints

- So far we assumed that an arc in the graph describes a precedence.
- We can annotate each arc $(X,Y)$ by a simple temporal constraint $[a,b]$ with the meaning $a \leq Y-X \leq b$.
- (Nested) Temporal Network with Alternatives
- Base constraint model:
  - Each node $A$ is annotated by a temporal variable $T_A$ with a domain $\langle 0,\text{MaxTime} \rangle$, where MaxTime is a constant given by the user.
  - Temporal relation $[a,b]$ between nodes $X$ and $Y$ must hold if both nodes are valid!

  $$V_X \land V_Y \land (T_X + a) \leq T_Y \land V_X \land V_Y \land (T_Y - b) \leq T_X.$$ 

Notes:
- $V_X = 0 \lor V_Y = 0 \rightarrow 0 \leq T_Y \land 0 \leq T_X$
- $V_X = V_Y = 1 \rightarrow (T_X + a) \leq T_Y \land (T_Y - b) \leq T_X$
- The above temporal constraint does not assume the type of branching!
Temporal hardness

- Is it possible to achieve global consistency of temporal relations in nested graphs?
- Unfortunately, the problem is NP-complete 😞
  - Subset sum problem can be converted to temporal feasibility of nested graphs.
  - Let $Z_i$, $i = 1, \ldots, n$ be integers, is there a subset $S$ of $\{1, \ldots, n\}$ such that $\sum_{i \in S} Z_i = K$?

Resource constraints

- standard scheduling model
  - start time variable: $T_A$
  - duration variable: $\text{Dur}_A$

- unary (disjunctive) resource constraints
  - two operations allocated to the same resource do not overlap in time

$$V_x \cdot V_y \cdot (T_x + \text{Dur}_x) \leq T_y \lor V_x \cdot V_y \cdot (T_y + \text{Dur}_y) \leq T_x$$

- or, we can use existing global constraints modeling unary resource (edge-finding, not-first/not-last, etc. inference techniques) extended to optional operations
  - (in)valid operations: $\text{Val}_A = 1 \iff \text{Dur}_A > 0$
Branching Strategy

1. ordering of activities in resources (with activity selection)
   - select some activity (earliest start combined with other criteria)
   - make the activity valid
   - decide its position in the resource (from start)
2. decision of times

Demo

- **FlowOpt** tools build on top of enterprise optimisation system MAK€ for SMEs
  - build-to-order (engineer-to-order) production
  - on-time-in-full objective (earliness/tardiness)

- What will you see?
  - interactive graphical design of workflows
  - creating and scheduling custom orders
  - visualisation and modification of schedules
  - schedule analysis
Workflow editor

- top-down and bottom up approach to design nested workflows
- supports extra logical (mutual exclusion,...) and temporal (synchronization,...) constraints

Optimiser

a fully automated scheduler that takes description of workflows for ordered products and generates a schedule

- implemented in ILOG CP Optimiser (OPL Studio)
- branch-and-bound optimisation (earliness and lateness costs and cost for alternatives are assumed)
Gantt Viewer

- visualization and modification of schedules

Analyser

- analysis of problems in schedules (late deliveries) and suggestions for enterprise improvements (buying a new resource)
Some results

- Number of resources: 34
- Number of activity types: 991
- Number of items: 294
- Number of orders: 45
- Total ordered quantity: 88.5 tons (88 485 kg)
- Schedule period: 1 week (10 080 minutes)

- Number of activities: 5946
- Number of precedences: 9325
- Runtime: 53 mins (Pentium 4/1700 MHz)

Summary

- Scheduling is not only mathematics but first of all a knowledge handling process.
  - how to capture real knowledge?
  - how to represent it formally so the user can verify it and update it?
  - how to exploit mathematical methods when real-life constraints are present?

- The art of real-life scheduling is to deliver a plan which is good enough and fast enough.
  - good enough – the user cannot improve it in reasonable time
  - fast enough – depends on the plant dynamics. One hour can be too late for one plant and very fast to another.