AdM III: Scheduling and Project Planning

Steffen Suerbier
Benjamin Labonte

October 26, 2012

Abstract

Lecture notes taken during the 2012/2013 lecture "Scheduling and Project Planning" by Prof. Dr. Rolf H. Möhring.

Contents

1 Projects and Partial Orders

What makes a project?

- activities or jobs $j \in V = \{1, 2, \cdots, n\}$
- job data
  - processing time/duration (deterministic/random)
  - resource consumption
  - processing cost
- project data
  - available resources
  - limited budget
- project rules for carrying out the project
  - temporal conditions
  - resource conditions
  - setups

Precedence constraints, i.e. job j has to wait for i define a partial ordering on V. We write $i < j$ if j must wait for i.

Definition 1.1 (Partial Ordering)**.** *A (strict) partial order is a binary relation $<$ over a set V which is*

(i) assymmetric: $i < j \Rightarrow j \nless i$

(ii) transitive: $i < j,\ j < k \Rightarrow i < k$

Definition 1.2 (Transitive Reduction of $<$, Transitive closure)**.** *The transitive reduction of the finite partial order $(V, <)$ is the digraph $G = (V, E)$ with the edges*

$$E = \{(i, j) \mid i, j \in V,\ i < j,\ \nexists k : i < k < j\}.$$

In the context of scheduling this is also called activity-on-node diagram. For any graph $G = (V, E)$ the edge set defined by

$$E^{trans} = \{(i, j) \mid \exists \text{finite sequence } i = i_0, \cdots, i_k = j : (i_j, i_{j+1}) \in E\}$$

denotes the transitive closure of E.

Example 1.3 (Bridge Construction Project)**.**

Definition 1.4.

$$\begin{aligned} Pred_G(i) &= \{j \in V \mid j < i\} && \textit{set of predecessors} \\ Suc_G(i) &= \{j \in V \mid j > i\} && \textit{set of successors} \\ ImPred_G(i) &= \{j \in V \mid (j,i) \in E\} && \textit{set of predecessors} \\ ImSuc_G(i) &= \{j \in V \mid (i,j) \in E\} && \textit{set of predecessors} \end{aligned}$$

We call an element maximal if it has no successors, minimal if it has no predecessors. If there is only one maximal (minimal) element it is called greatest (smallest). i, j are comparable ($i \sim_G j$) if $i < j$ or $j > i$ otherwise they are incomparable ($i \parallel_G j$).

Lemma 1.5. *Let $(V, <)$ be a finite partial order and (V, E) its transitive reduction. Then E is the smallest (under $\subset$) binary relation, s.t. $E^{trans} = <$.*

Proof. Show that $E^{trans} = <$. Let $i < j$ and $(i, j) \notin E$. Then there is a $k \in V$ s.t. $i < k < j$. As $<$ is acyclic and finite, by iterating, we obtain a finite sequence $i = i_0 \to \cdots \to j$. □

Remark 1.6. Lemma 1.5 does not hold for relations with cycles or infinite ground sets.

Exercise 0. Show that E^{trans} exists.

Exercise 1. Formulate an algorithm for constructing the transitive closure of a digraph.

Exercise 2. Formulate an algorithm for constructing the transitive reduction of a partial order or a directed acyclic graph.

Exercise 3. Prove an $\mathcal{O}(n^3)$ upper bound on the running times of the algorithms.

2 Scheduling subject to scarce resources: Introduction and complexity

Example 2.1. Consider the following standard model in parallel processing environments. Every long job is cut into pieces of length 1. m-machine problems for $m \geq 2$, G precedence constraints, $x_j = 1, \kappa = C_{max}$.

Results $m = 2$ polynomially solvable
$m = 3$ open (for a long time)
$m \geq 4$ $\mathcal{NP} - complete$

Idea for $m = 2$ Match jobs that are scheduled in same time slot. Use incomparability graph $Incomp(G)$ with vertices $V =$ set of jobs and edges $(i, j) \in E \Leftrightarrow i \parallel_G j$.

Theorem 2.2. $C_{max}^{OPT,G}(\mathbb{1})$ = *maximum size of a matching in $Incomp(G)$ + #unmatched jobs [Fujii et al. 67,71].*

Proof. Construct a schedule from the scheduling from the matching requires "uncrossing" of crossings.

crossing: Assume (a, c) and (b, d) are precedence constraints, then a matching of a with d and b with c would lead to an infeasible schedule. The uncrossing is simply the matching a with b and c with d.

Claim. $\forall$ matchings M there is a non-crossing matching of the same size that can be obtained by uncrossing crossings in any order

Proof. Every oncrossing reduces the number of crossings. Suppose not, i.e. consider $(u_1, u_2), (u_3, u_4)$, uncross the matching $u_1 \leftrightarrow u_4, u_2 \leftrightarrow u_3$ and this gives a new crossing $u_1 \leftrightarrow u_2, v_1 \leftrightarrow v_2 \Rightarrow \exists u_1 < v_1, v_2 < u_2 : (v_1, v_2) \in M$. But that means that before uncrossing (v_1, v_2) crossed with $(u_1 m u_4)$ which no longer exists.

This shows that for every new crossing an old crossing involving the same edges is uncrossed. □

Now assume that M is non-crossing. Turn that into a schedule.

1. order the matched edges [picture], if one of $\{a, b\}$ is before on of $\{c, d\}$ in G^{trans} (possible because of non-crossing)
2. order unmatched jobs and matched edges [picture] if a is before one of $\{c, d\}$ in G^{trans}. (similar: [picture matching before job]). This is well defined since $c \parallel_G d$.
3. take any linear extension of the transitive closure of $\prec$ defined in (1), (2) and $a <_G$ between unmatched jobs.
4. Replace matching edges by their two end points. This defines G^* on V.
5. Consider $ES_{G^*}(\mathbb{1})$

□

Example 2.3 (continued)**.** TODO

Problem 1 (n-Machine Problem)**.**
Input: $G, m, t, x = \mathbb{1}$
Output: *Is there a feasible schedule of length $\leq t$?*

Theorem 2.4. *The m-machine problem is $\mathcal{NP}$ – complete for $m \geq 4$.*

Proof. Reduction from Clique. Vertex jobs $u_1, \cdots, u_n$, edge jobs u_{ij}, precedence constraints $u_{ij} > u_i, u_{ij} > u_j$ and dummy jobs which provide a frame. $k = 3$ and $t = 3$ (always).

Idea Can schedule k vertex jobs in slot A, $\binom{k}{2}$ edge jobs and remaining vertex jobs in slot B and remaining edge jobs in slot C.

If G constains a clique of size k then we can schedule the vertex jobs from the clique in the first time slot, the edge jobs of the clique and the remaining vertex jobs in the second slot and the remaining edge jobs in the third time slot. So we get a schedule in 3 time slots.

If G does not contain a cliqe of size k then one machine is idle in time slot 2 and we would need 4 time slots. □

Exercise 4. Show that Jacksons's rule constructs an optimal schedule for L_{max} on one machine.

Exercise 5. Show that the 3-machine problem can be solved in polynomial time for precedence constrains of fixed with (i.e. the size of the largest antichain is bounded by a constant).