Sequencing Jobs with One Common and Multiple Individual Resources with Multivalued Decision Diagrams

Dissertantenseminar

Johannes Maschler, Günther R. Raidl, and Elina Rönnberg

January 8, 2017

ALGORITHMS AND COMPLEXITY GROUP

Problem Definition: PC-JSOCMSR

We consider the Prize-Collecting Job Sequencing with One Common and Multiple Secondary Resources (PC-JSOCMSR) problem.

We are given

- jobs $J = \{1, \ldots, n\}$, and
- resources $R_0 = \{0\} \cup R$ with $R = \{1, \ldots, m\}$

Problem Definition: PC-JSOCMSR

We consider the Prize-Collecting Job Sequencing with One Common and Multiple Secondary Resources (PC-JSOCMSR) problem.

We are given

- jobs $J = \{1, \ldots, n\}$, and
- resources $R_0 = \{0\} \cup R$ with $R = \{1, \ldots, m\}$

Each job $j \in J$ has

- a processing time $p_j > 0$
  - during which it fully requires resource $q_j \in R$ and
  - the common resource 0 for a part of its execution
    - for p_j^0 time beginning at p_j^{pre} after the jobs' start
- a set of time windows $W_j = \bigcup_{w=0,\ldots,\omega_j} W_{j,w}$
  - with $W_{j,w} = [W_{j,w}^{\mathrm{start}}, W_{j,w}^{\mathrm{end}}]$
- a prize $z_j > 0$.

Problem Definition: PC-JSOCMSR

We consider the Prize-Collecting Job Sequencing with One Common and Multiple Secondary Resources (PC-JSOCMSR) problem.

We are given

- jobs $J = \{1, \ldots, n\}$, and
- resources $R_0 = \{0\} \cup R$ with $R = \{1, \ldots, m\}$

Each job $j \in J$ has

- a processing time $p_j > 0$
  - during which it fully requires resource $q_j \in R$ and
  - the common resource 0 for a part of its execution
    - for p_j^0 time beginning at p_j^{pre} after the jobs' start
- a set of time windows $W_j = \bigcup_{w=0,\ldots,\omega_j} W_{j,w}$
  - with $W_{j,w} = [W_{j,w}^{\mathrm{start}}, W_{j,w}^{\mathrm{end}}]$
- a prize $z_j > 0$.

We are looking for a subset of jobs $S \subseteq J$

- that can be feasibly scheduled and
- maximizes the total prize, i.e., $\sum_{j \in S} z_j$.

Solution Representation

Observe that each job requires resource 0.

- Hence, a schedule of the jobs S implies a total ordering of the jobs.

We represent a solution by a permutation $\pi = (\pi_i)_{i=1,\ldots,|S|}$.

A normalized schedule is obtained by scheduling each job from S in the order given by π at the earliest feasible time.

Obviously, any optimal solution either is

- a normalized schedule, or
- there exists a corresponding normalized schedule.

Closely Related Literature

Andre A. Cire and Willem-Jan Van Hoeve.
Multivalued decision diagrams for sequencing problems.
Operations Research, 61(6):1411–1428, 2013.

David Bergman, Andre A. Cire, Willem-Jan van Hoeve, and J. N. Hooker.
Discrete optimization with decision diagrams.
INFORMS Journal on Computing, 28(1):47–66, 2016.

David Bergman, Andre A. Cire, Willem-Jan van Hoeve, and John N. Hooker.
Decision Diagrams for Optimization.
Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, 2016.

Recursive Model for PC-JSOCMSR

The control variables of the model are $\pi_1, \ldots, \pi_n \in J$.

A state (P, t) consists of

- the set $P \subseteq J$ of jobs that still can be scheduled, and
- the vector $t = (t_r)_{r \in R_0}$ of the times from which on each resource r is available for performing a next job.

The initial state is $\mathbf{r} = (J, (T^{\min}, \ldots, T^{\min}))$.

Recursive Model for PC-JSOCMSR (cont.)

The earliest feasible time for job $j \in J$ not smaller than t is given by

$$\mathrm{eft}(j,t) = \min\{\infty,\ t' \geq t \mid [t', t' + p_j] \subseteq W_j\}. \tag{1}$$

Let the starting time of a next job $j \in J$ w.r.t. a state (P, t) be

$$\mathrm{s}((P,t),j) = \begin{cases} \mathrm{eft}(j, \max(t_0 - p_j^{\mathrm{pre}}, t_{q_j})) & \text{if } j \in P \\ \infty & \text{else.} \end{cases} \tag{2}$$

Recursive Model for PC-JSOCMSR (cont.)

The transition function to obtain the successor (P', t') of state (P, t) when scheduling job $j \in J$ next is

$$\tau((P,t),j) = \begin{cases} (P \setminus \{j\}, t') & \text{if } \mathrm{s}((P,t),j) \neq \infty \\ \hat{0} & \text{else,} \end{cases} \quad (3)$$

with

$$t'_0 = \mathrm{s}((P,t),j) + p_j^{\mathrm{pre}} + p_j^0 \quad (4)$$

$$t'_r = \mathrm{s}((P,t),j) + p_j \qquad \text{for } r = q_j \quad (5)$$

$$t'_r = t_r \qquad \text{for } r \in R \setminus \{q_j\} \quad (6)$$

and $\hat{0}$ representing the infeasible state.

Recursive Model for PC-JSOCMSR (cont.)

All states except the infeasible state $\hat{0}$ are terminal states.

Any sequence of state transitions $\tau(\ldots\tau(\mathbf{r},\pi_1)\ldots,\pi_i)$ yielding a terminal state represents a solution $(\pi_1,\ldots,\pi_i)$.

The cost associated with a state transition are $h((P,t),j) = z_j$.

PC-JSOCMSR can be solved by calling the following function with $Z^*(\mathbf{r})$:

$$Z^*(P,t) = \max\{0, z_j + Z^*(\tau((P,t),j)) \mid j \in P \wedge \tau((P,t),j) \neq \hat{0}\}. \quad (7)$$

Multivalued Decision Diagrams for PC-JSOCMSR

An MDD is a directed acyclic multi-graph $G = (V, A)$.

MDD $G = (V, A)$ is obtained from the recursive model by creating

- nodes for the terminal (feasible) states,
- arcs for all state transitions between terminal states
  - of length $h((P, t), j) = z_j$,

Paths from **r** to some node $v \in V$ correspond to solutions.

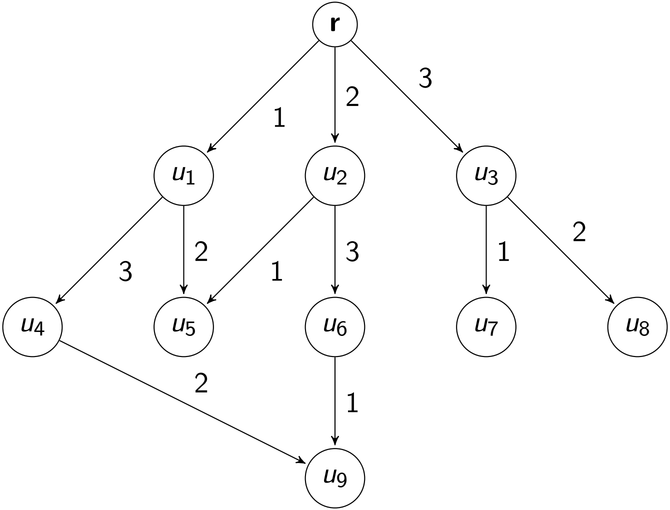
An optimal solution corresponds to a longest path in the MDD.

Such an MDD is called exact because we have

$$\mathrm{Sol}(\mathcal{P}) = \mathrm{Sol}(G), \tag{8}$$

$$Z(\pi) = Z^{\mathrm{lp}}. \tag{9}$$

Multivalued Decision Diagrams for PC-JSOCMSR



Relaxed MDDs for PC-JSOCMSR

A simple relaxation scheme merges a subset M of feasible states to obtain the state

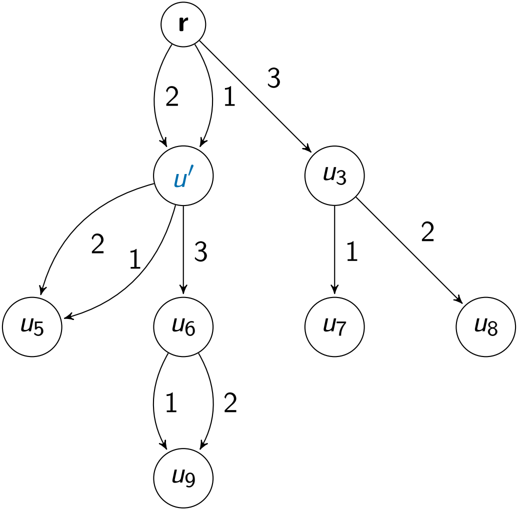
$$\oplus(M) = \left(\bigcup_{(P,t)\in M} P, \left(\min_{(P,t)\in M} t_r \right)_{r\in R_0} \right). \tag{10}$$

For a relaxed MDD we have

$$\mathrm{Sol}(\mathcal{P}) \subseteq \mathrm{Sol}(G), \tag{11}$$

$$Z(\pi) \leq Z^{\mathrm{lp}}. \tag{12}$$

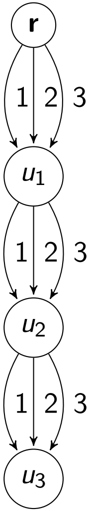
Relaxed MDDs for PC-JSOCMSR



Relaxed MDDs of Width One

A relaxed MDD of width one can be obtained by

- adding $\{0, \ldots, n+1\}$ nodes
  - where node 0 is associated with the initial state
- connect node $i = 0, \ldots, n$ with node $i+1$
  - by n arcs representing transitions for each job $j \in J$



An Incremental Refinement Algorithm for MDDs

Input: relaxed MDD $G = (V, A)$ with source node $\mathbf{r}$
Let $p = (a^{(1)}, \ldots, a^{(n)})$ be the longest path in G;
while *p is infeasible* **do**
    if *p contains a repetition of job j* **then**
        refine reptition of job j;
    else
        refine time window violation;
    end
    update longest path p;
end

An Incremental Refinement Algorithm for MDDs

Let $\mathrm{All}_u^{\downarrow} \subseteq J$ be the jobs on all paths from $\mathbf{r}$ to $u \in V$, i.e.,

$$\mathrm{All}_u^{\downarrow} = \bigcap_{a=(v,u)\in A^+(u)} \left(\mathrm{All}_v^{\downarrow} \cup \{\mathrm{job}(a)\}\right) \tag{13}$$

Let $\mathrm{Some}_u^{\downarrow} \subseteq J$ be the jobs on some path from $\mathbf{r}$ to $u \in V$, i.e.,

$$\mathrm{Some}_u^{\downarrow} = \bigcup_{a=(v,u)\in A^+(u)} \left(\mathrm{Some}_v^{\downarrow} \cup \{\mathrm{job}(a)\}\right) \tag{14}$$

Let $\mathrm{Some}_u^{\uparrow} \subseteq J$ be the jobs on some path from $u \in V$ to any reachable node, i.e.,

$$\mathrm{Some}_u^{\uparrow} = \bigcup_{a=(v,u)\in A^-(u)} \left(\mathrm{Some}_v^{\uparrow} \cup \{\mathrm{job}(a)\}\right) \tag{15}$$

Filtering a Relaxed MDD

We remove arcs for which all paths that cross them violate a constraint.

We can remove any arc $a = (u, v)$ if

- $\mathrm{s}(u, \mathrm{job}(a)) = \infty$
- $\mathrm{job}(a) \in \mathrm{All}_u^{\downarrow}$
- $|\mathrm{Some}_u^{\downarrow}| = \mathrm{Hops}_u^{\min}$ and $\mathrm{job}(a) \in \mathrm{Some}_u^{\downarrow}$
- $Z^{\mathrm{lp}}(v) + Z^{\mathrm{ub}}(v) < Z^{\mathrm{lb}}$

Nodes without an ingoing arc except **r** can be removed together with all its outgoing arcs.

Refinement of Job Repetitions

Lemma

A job j is assigned on each path starting from $\mathbf{r}$ *at most once if and only if* $j \notin \mathrm{Some}_u^{\downarrow} \cap \mathrm{Some}_u^{\uparrow} \setminus \mathrm{All}_u^{\downarrow}$ *for all nodes* $u \in V$*.*

Given job j. For all nodes $u \in V$ with $j \in \mathrm{Some}_u^{\downarrow} \cap \mathrm{Some}_u^{\uparrow} \setminus \mathrm{All}_u^{\downarrow}$:

- Replace u by two nodes u_1 and u_2
  - redirect all incoming arcs $a = (v, u)$ to u_1 if $j \in \mathrm{All}_v^{\downarrow} \cup \{\mathrm{job}(a)\}$ and
  - to u_2 otherwise,
  - replicate all outgoing arcs for both nodes.

Refinement of Time Window Violations

Let $(a^{(1)}, \dots, a^{(k)})$ be a path in our MDD starting at root $\mathbf{r}$, where

- $(a^{(1)}, \dots, a^{(k-1)})$ is a feasible solution and
- the last job violates its time windows.

Let $(a^{(i)}, \dots, a^{(k)})$ be the smallest subpath s.t.
$\tau(\dots \tau((P^{(i)}, t^{(i)}), j^{(i)}) \dots, j^{(k)})$ violates the last job's time window.

For node $(P^{(i)}, t^{(i)})$ to $(P^{(k)}, t^{(k)})$ do:

- replace current node $u = (P^{(l)}, t^{(l)})$ by nodes u_1 and u_2.
- Let $(P', t') = \tau((P^{(l-1)}, t^{(l-1)}), j^{(l-1)})$.
- Redirect all incoming arcs $a = (v, u)$ to u_1 if $(P'', t'') = \tau(v, \mathrm{job}(a))$ and $t''_r \geq t'_r$ for all $r \in R$.
- All other incoming arcs are redirected to u_2.
- The outgoing arcs are replicated for u_1 and u_2.

Next Steps

- Implementation of the algorithm
- Alternative initial (relaxed) MDDs
- Preprocessing of initial (relaxed) MDDs
- Identifying supplementary filtering rules
- Combination with A* approach