Exact solution methods for the Resource Constrained Project Scheduling Problem with a Flexible Project Structure

van der Beek, T.; van Essen, J.T.; Pruijn, J.F.J.; Aardal, K.I.

Publication date
2022

Published in
Optimization Online

A. Notation

Variables

V_i	1 if activity $i \in N$ is selected for the NEES and zero otherwise
W_i	1 if activity $i \in N$ is selected for the MOES and zero otherwise
X_{it}	1 if activity $i \in N$ is executed at time $t \in T$, zero otherwise
Y_g	1 if group $g \in H$ is selected and zero otherwise
Z_i	1 if activity $i \in N$ is selected as successor activity and zero otherwise

Parameters

a_g	Activating activity of selection group $g \in G$
d_i	Duration of activity $i \in N$
f_i	Latest finish time of activity $i \in N$
k_{ri}	Usage of resource $r \in R$ for activity $i \in N$.
l	Lower bound on objective.
M	Very large number
n	Number of non-dummy activities
s_i	Earliest start time of activity $i \in N$
u	Upper bound on objective.
$\ell(P)$	Number of vertices in path P
λ_r	Capacity of resource $r \in R$

Sets

$E^{(a)}$	Active edges representing a CRP
$E^{(f)}$	Final edges representing a CRP
$E^{(n)}$	New edges representing a CRP
G	Selection groups
H	Selection groups with full precedence
N	Activities
R	Resources
S_g	Successor activities of selection group $g \in G$
T	Time periods
$\mathcal{C}_i$	Cutting planes for activity $i \in N$
$\mathcal{F}_i$	Forced activities for activity $i \in N$
$\mathcal{P}$	Precedence relationships (tuples of 2 activities)
$\mathcal{P}_j$	Predecessors of activity $j \in N$ in the precedence graph
$\mathcal{R}_i(A)$	All paths starting in i and ending in an activity in set A, with only the last activity in set A
$\mathcal{S}_i$	Successors of activity $i \in N$ in the precedence graph
$\mathcal{V}(P)$	Vertex set of path P
Γ	All activity pairs (i, j) if $i \in N$ and $j \in N$ are successors in the selection graph of the same selection group
Θ	Pairs (i, j) of activities where i is reachable from j of vice versa
Θ_i	All activities that are reachable from activity $i \in N$
Ω_i	Successors of activity $i \in N$ in the selection graph

B. Dummy variables

Lemma 2

Consider a selection group $g \in G$ for which S_g does not contain the final activity $n+1$. This selection group will be modified such that at least one successor $i \in S_g$ has to be executed if the activator a_g is executed, instead of exactly one. This can be achieved by applying the following algorithm:

Step 1 *Let $S'_g = \{i : i \in S_g, \{k : k \in G, i \in S_k, k \neq g\} \neq \emptyset\}$ be a subset of S_g containing only activities that are also successors of another group. Create a dummy activity D_i for each successor activity $i \in S'_g$.*

Step 2 *Create a selection group h with $a_h = a_g$ and $S_h = \{D_i : i \in S'_g\} \cup \{i \in S_g \setminus S'_g\}$*

Step 3 *Create a selection group h_i for each $i \in S'_g$ with $a_{h_i} = D_i$ and $S_{h_i} = \{i\}$.*

Step 4 *Remove selection group g.*

Proof. The algorithm adds dummy activities for all activities $i \in S'_g$. We first show that if we do this for all activities $i \in S_g$ instead, we impose an 'at least one' constraint instead of an 'exactly one' constraint. After that, we show that if we remove all dummy activities (and corresponding groups) for $i \in S_g \setminus S'_g$, the solution stays feasible and the optimal solution value does not change.

First, apply the algorithm with $S'_g = S_g$. Now, Constraints (1d) and (1e) impose for selection group h that if a_h is executed, *exactly* one dummy variable has to be executed. Consequently, Constraints (1d) impose that *at least* one activity $i \in S_g$ has to be executed, since activity $i \in S_g$ can also be executed when the activating dummy activity D_i is not executed.

Let $\mathcal{A}$ be the problem instance obtained by performing the algorithm for $S'_g = S_g$. Furthermore, let $\mathcal{B}$ be the problem instance we obtain by using S'_g instead of S_g. Converting $\mathcal{A}$ to $\mathcal{B}$ can be done as follows: For each activity $i \in S_g \setminus S'_g$, remove dummy activity D_i, remove selection group h_i, and replace successor activity D_i in selection group h by the original successor activity i.

To show that the 'at least one'-criterium also holds for $\mathcal{B}$, we will show that each solution X to instance $\mathcal{A}$ can be converted to a solution Y to instance $\mathcal{B}$ and vice versa, while keeping the same objective value.

Let X be a solution to problem $\mathcal{A}$. We now show that solution X can be converted to a feasible solution Y for problem $\mathcal{B}$ with the same objective value. Converting is done by projecting all values of X on Y and modifying the values for $i \in S_g \setminus S'_g$ if needed.

Firstly, we consider the case where activity $i \in S_g \setminus S'_g$ is not executed in solution X. By Constraints (1d), it follows that dummy activity D_i is also not executed. Therefore, removing D_i and selection group h_i does not cause any infeasibilities and Y remains a feasible solution for $\mathcal{B}$.

Secondly, consider the case where activity $i \in S_g \setminus S'_g$ is executed. If dummy activity D_i was executed, the number of executed successor activities for selection group h stays the same in solution Y, which therefore remains feasible for problem $\mathcal{B}$. If D_i was not executed, which is possible considering Constraints (1d), there is an infeasibility in problem $\mathcal{B}$ for group h in Constraints (1e). However, since activity i is not the successor of any other group, it can be set to not executed. This does not cause any infeasibilities because Constraints (1d) impose in the direction of activator to successor and not in the reverse direction.

Therefore, any solution X for problem $\mathcal{A}$ can be converted to a solution Y for problem $\mathcal{B}$. Since the value of the objective activity $n+1$ is not changed, the objective value remains equal.

Next, we show that a solution Y for problem $\mathcal{B}$ can be converted to a solution X for problem $\mathcal{A}$ with equal objective value. This is done by projecting Y on X and setting the values for the dummy activities as required.

Again, consider activity $i \in S'_g \setminus S_g$. If this activity is not executed, set the corresponding dummy activity D_i to not executed in X as well. Then, the problem remains feasible.

Now, consider the case where activity $i \in S'_g \setminus S_g$ is executed. In this case, none of the dummy activities are executed and dummy activity D_i can be set to executed in solution X to obtain a feasible solution. Similar as for the reverse case, the objective activity $n+1$ is not changed, and therefore, the objective value remains equal.

Thus, there exists a solution X for problem $\mathcal{A}$ if and only if there exists a solution Y for problem $\mathcal{B}$ with the same objective value. Therefore, the 'at least one'-criterium from problem $\mathcal{A}$ is also imposed on problem $\mathcal{B}$. $\square$