Encoding methods for the RCPSP

September 25, 2025

0 Some notation

Parameters

$V = \{0, \dots, n+1\}$	The set of all activities (jobs)
$A = \{1, \dots, n\}$	The set of non-dummy activities
p_i	Processing time (duration) of activity i
$G = (V, E)$	Precedence graph
$(i, j) \in E$	A precedence relation between activities i and j; specifically i must finish before j starts
$R = \{1, \dots, v\}$	Set of (renewable) resources
B_k	Capacity (maximum availability) of resource k
$b_{i,k}$	Resource requirement of activity i on resource k
UB	Upper bound on the possible makespan
LB	Lower bound on the possible makespan
$H = \{0, \dots, UB-1\}$	Scheduling horizon

Values derived from the parameters

$G^* = (V, E^*)$	Extended precedence graph
$(i, j, l_{i,j}) \in E^*$	Extended precedence relation with time lag $l_{i,j}$
$ES(i)$, $LS(i)$	Earliest and latest start times of activity i
$EC(i)$, $LC(i)$	Earliest and latest completion times of activity i
$RTW(i)$	The time range in which activity i can be executed
$STW(i)$	The time range in which activity i can start

Variables used in the encoding

S_i	Start time of activity i
$s_{i,t}$	Binary: activity i starts at time t
$x_{i,t}$	Binary: activity i is being processed at time t
m_t	Makespan equals t

1 Problem definition

The objective of the *Resource-Constrained Project Scheduling Problem (RCPSP)* is to determine a start time for each activity in the project so that the makespan is minimized.

Formally, the RCPSP is defined by the tuple (V, p, E, R, B, b) where:

- $V = \{0, 1, 2, \dots, n+1\}$ is the set of activities. Activities 0 and $n+1$ are dummy activities representing the start and the end of the project. The set of non-dummy activities is $A = \{1, 2, 3, \dots, n\}$.
- $p \in \mathbb{N}^{n+2}$ is a vector of natural numbers where p_i is the processing time of activity i. For dummy activities we have $p_0 = p_{n+1} = 0$, and $p_i > 0 \quad \forall i \in A$.
- E is a set of ordered pairs representing precedence relations between activities. Specifically, $(i, j) \in E$ iff activity i completes before activity j starts. We assume the precedence graph

$G = (V, E)$ is acyclic, and in G every activity is reachable from activity 0, and every activity can reach activity $n + 1$.

- $R = \{1, 2, .., v\}$ is the set of renewable resources.
- $B \in \mathbb{N}^v$ is a vector of natural numbers where B_k is the available capacity of resource k.
- $b \in \mathbb{N}^{(n+2)\times v}$ is a matrix of natural numbers corresponding to resource requirements of each activity, where $b_{i,k}$ is the amount of resource k required by activity i per unit time during its execution. $b_{0,k} = b_{n+1,k} = 0$ and $b_{i,k} > 0, \forall k \in 1..v, \forall i \in 1..n$.

A schedule is a vector of natural numbers $S = (S_0, S_1, .., S_{n+1})$ where S_i is the start time of activity i. Without loss of generality, we assume $S_0 = 0$. An optimal solution to the RCPSP is a schedule S that minimizes the makespan S_{n+1} while satisfying the following conditions:

- Precedence condition: each activity may only start after all its predecessors have finished.
- Resource condition: at any time point, the total demand for any resource by all activities being processed at that time must not exceed the available capacity of that resource.

2 Preprocessing

2.1 Extended precedence relations

Denote by $G^* = (V, E^*)$ the extended precedence graph, where E^* is a set of weighted edges. For each pair of activities (i, j) in V such that there exists a path from i to j in G, there is a corresponding edge $(i, j, l_{i,j})$ in E^*. The time lag $l_{i,j}$ is the minimal (lower bound) value of the difference between the start times of activities i and j. This value can be computed using the Floyd–Warshall algorithm.

2.2 Energetic reasoning on precedences

Note that for all activities a satisfying $(i, a, l_{i,a}) \in E^* \wedge (a, j, l_{a,j}) \in E^*$, they must be executed and finished in the interval $[S_i + p_i, S_j - 1]$. Therefore, this interval must be large enough to accommodate all such activities a without exceeding the capacity of any resource. Hence, for each resource $k \in R$, a lower bound on the distance between the completion of i and the start of j can be computed as:

$$RLB_{i,j,k} = \left\lceil \frac{1}{B_k} \times \sum_{\substack{a \in A \text{ s.t.} \\ (i,a,l_{i,a}) \in E^* \\ (a,j,l_{a,j}) \in E^*}} (p_a \cdot b_{a,k}) \right\rceil \tag{1}$$

Based on this formula, we can update the time lags computed in section 2.1 by:

$$l^*_{i,j} = \max(l_{i,j}, p_i + \max_{k \in R}(RLB_{i,j,k})) \qquad \forall (i, j, l_{i,j}) \in E^* \tag{2}$$

2.3 Time windows

Given the extended precedence graph, for each activity in the project we can compute its earliest start, latest start, earliest completion and latest completion times. Specifically:

$$ES(i) = l_{0,i}$$
$$EC(i) = l_{0,i} + p_i$$
$$LS(i) = UB - l_{i,n+1}$$
$$LC(i) = UB - l_{i,n+1} + p_i$$

We also define:

$$RTW(i) = [ES(i), LC(i) - 1]$$
$$STW(i) = [ES(i), LS(i)]$$

3 Constraint formulations

Activity 0 starts at time 0

$$s_{0,0} \tag{3}$$

Each activity starts at exactly one time in $STW(i)$

$$\sum_{t\in STW(i)} s_{i,t} = 1 \quad \forall i \in V \setminus \{0\} \tag{4}$$

Precedence relations: there are three cases for each edge $(i, j, l_{i,j}) \in E^*$:

1. $S_i + l_{i,j} - 1 < ES(j)$ (Activity i starts at a time such that it will finish before activity j starts): no encoding needed for this case.
2. $ES(i) \leq S_i + l_{i,j} - 1 \leq LS(j)$ (Activity i starts at a time such that it will finish within $STW(j)$):

$$\begin{aligned} s_{i,t'} + \sum_{t\in[ES(j),\,k]} s_{j,t} \leq 1 \quad & \forall (i,j) \in E, \\ & \forall k \in STW(j) \text{ s.t. } k - l_{i,j} + 1 \leq LS(i), \\ & t' = \max(ES(i),\, k - l_{i,j} + 1) \end{aligned} \tag{5}$$

3. $S_i + l_{i,j} - 1 > LS(j)$ (Activity i starts at a time such that it will finish after activity j has started):

$$s_{i,t'} + \sum_{t\in STW(j)} s_{j,t} \leq 1 \quad \forall t' \text{ s.t. } \max(LS(j) - l_{i,j} + 1, ES(i) - 1) < t' \leq LS(i) \tag{6}$$

If activity i starts at time t then it must be processing during the interval $[t, t + p_i - 1]$

$$\neg s_{i,t} \vee x_{i,t'} \quad \forall i \in V, \forall t \in STW(i), \forall t' \in [t, t + p_i - 1] \tag{7}$$

Back-propagation constraints to accelerate solving

$$\neg s_{i,t} \vee x_{i,t+1} \vee s_{i,t-p_i+1} \qquad \forall i \in V, \forall t \in RTW(i) \setminus \{LC(i) - 1\} \tag{8}$$

Resource limits

$$\sum_{\substack{i\in A \text{ s.t.} \\ t\in RTW(i)}} b_{i,k}\, x_{i,t} \leq B_k \quad \forall k \in R, \forall t \in H \tag{9}$$

4 Encoding start conditions and precedence relations

Denote $R_{i,a,b}$ as the staircase representation equivalent to $\sum_{a\leq t<b}, s_{i,t} \leq 1$. We have:

$$s_{i,a} \vee \neg R_{i,a,a} \tag{10}$$

$$\neg s_{i,t} \vee R_{i,a,t} \quad t \in [a, b] \tag{11}$$

$$\neg R_{i,a,t-1} \vee R_{i,a,t} \quad \forall t \in (a, b] \tag{12}$$

$$R_{i,a,t-1} \vee s_{i,t} \vee \neg R_{i,a,t} \quad \forall t \in (a, b] \tag{13}$$

$$\neg R_{i,a,t-1} \vee \neg s_{i,t} \quad \forall t \in (a, b] \tag{14}$$

Then clauses 4, 5, 6 become respectively:

$$R_{i,ES(i),LS(i)} \tag{15}$$

$$\neg s_{i,t'} \vee \neg R_{j,ES(j),k} \tag{16}$$

$$\neg s_{i,t'} \vee \neg R_{j,ES(j),LS(j)+1} \tag{17}$$

5 Optimizing the makespan

Use an incremental SAT approach. After finding a solution with makespan $S_{n+1} = T$, add the following assumption clause

$$\neg s_{n+1,T} \tag{18}$$

and continue searching. If UNSAT is returned then conclude $S_{n+1} = T$. If a solution is found, repeat the step until UNSAT or until reaching the LB.