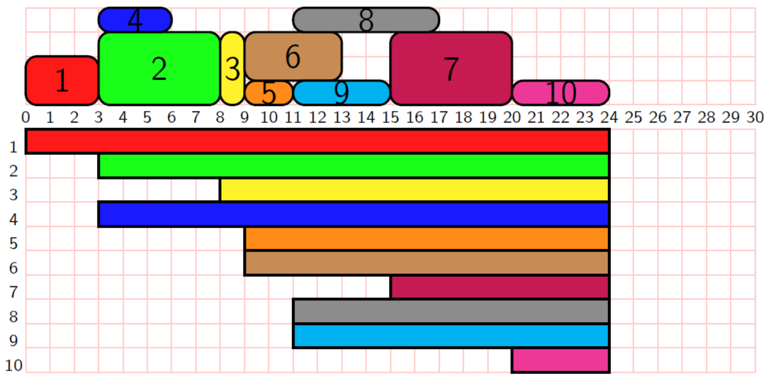
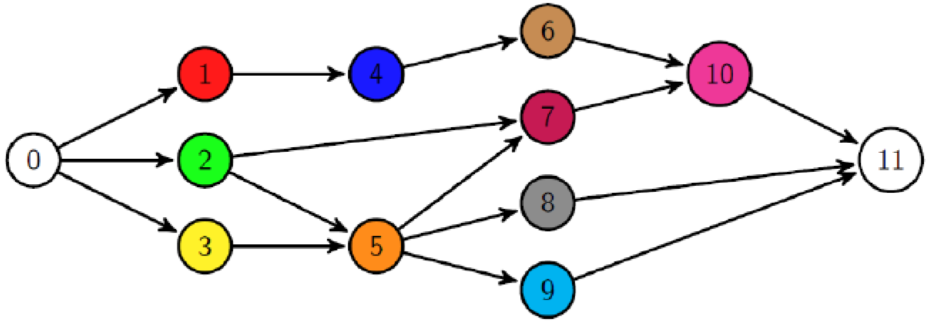
Discrete Optimization and Integer Programming

Integer Programming for Resource-Constraint Project Scheduling Problem





Outline

- RCPSP classic problem statement
- Solving methods overview
    - MILP models
- Network rollout optimization

Outline

- RCPSP classic problem statement
- Solving methods overview
  - MILP models
- Network rollout optimization

RCPSP: classic statement

Set of renewable resources $\boldsymbol{R}$:

- c_r – constant capacity of resource $r \in R$

Set of tasks (activities) $\boldsymbol{N}$:

- p_j – processing time of $j \in N$
- a_{jr} – required amount of renewable resource $r \in R$ during the processing of $j \in N$

Set of precedence constraints $\boldsymbol{E}$:

- $(i, j) \in E$ means that task $i \in N$ must be competed before start of $j \in N$

Scheduling horizon $\boldsymbol{T}$

RCPSP: classic statement

Schedule π

- $S_j(\pi)$ – start time of the task j under π
- $C_j(\pi) = S_j(\pi) + p_j$ – completion time of the task j under π

Set of feasible schedules Π

Consists of schedules π which hold:

- $0 \le S_j(\pi) \le T - p_j$ – each tasks should be processed in the scheduling interval $[0, T)$
- $C_i(\pi) \le S_j(\pi)$ for each $(i, j) \in E$ – precedence relations should be satisfied
- $\sum_{j \in N: S_j(\pi) \le t < C_j(\pi)} a_{jr} \le c_r$ for each $r \in R$ and $t \in [0, T)$ – resource capacities should not be violated

RCPSP: classic statement

Objective

Find feasible schedule with minimal makespan:

$$\min_{\pi \in \Pi} \max_{j \in N} C_j(\pi)$$

NP-hard in a strong sense
(Garey, Johnson 1975)

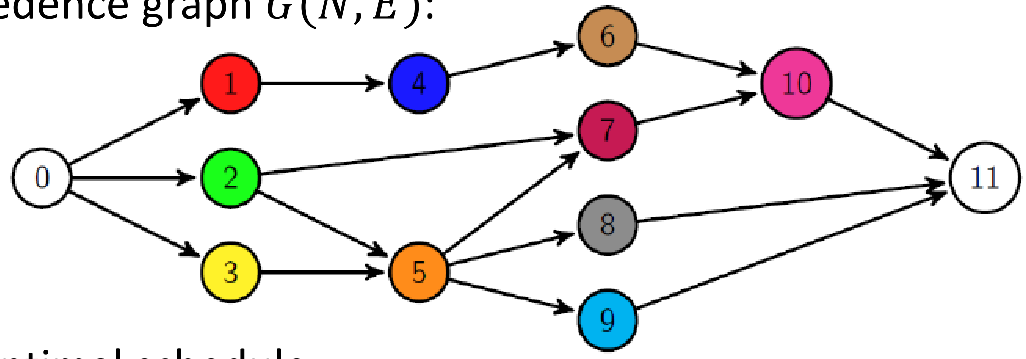
Example

$|R| = 1,\ c = 4,\ T = [0, 30)$

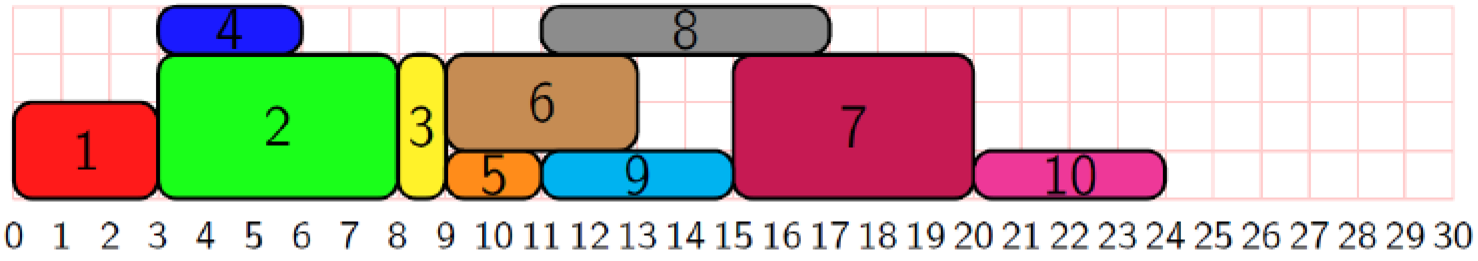
Tasks

i	p_i	a_i
1	3	2
2	5	3
3	1	3
4	3	1
5	2	1
6	4	2
7	5	3
8	6	1
9	4	1
10	4	1

Precedence graph $G(N, E)$:



Optimal schedule:



Outline

- RCPSP classic problem statement
- Solving methods overview
  - MILP models
- Network rollout optimization

RCPSP: solving methods

Comprehensive surveys on formulations and solving methods:

- R. Kolish, R. Padman. *An Integrated Survey of Project Scheduling*, 1997.
- J. Weglarz. *Project Scheduling*, 1999.
- S. Hartmann, R. Kolisch. *Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem,* 2000.
- E. Demeulemeester, W. Herroelen. *Project Scheduling*, 2002.
- C. Schwindt, J. Zimmermann (eds.), *Handbook on Project Management and Scheduling, 2015.*

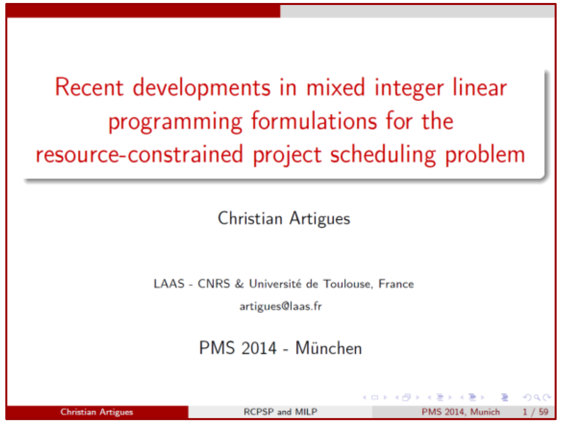
RCPSP: solving methods

Mixed-Integer Linear Programming | Constraint Programming | Branch and bound methods | Heuristics and Metaheuristics

One of the most popular approach for solving RCPSP is Mixed Integer Linear Programming (MILP). There is a plenty of MILP models for RCPSP, most of which are referred in surveys:

- O. Kone et. al. *Event-based MILP models for resource-constrained project scheduling problems,* 2011.
- C. Artigues et. al. *Mixed-Integer Linear Programming Formulations, 2015.*

In the following subsection of presentation the materials of Christian Artigues plenary presentation at PMS’2014 conference will be used.



RCPSP: solving methods

Mixed-Integer Linear Programming

Constraint Programming

Branch and bound methods

Heuristics and Metaheuristics

There is a wide range of existing constraint propagation algorithms which can be applied to find an optimal/suboptimal solution for RCPSP or to make the problem easier to solve. The most comprehensive surveys:

- P. Baptiste, C. Le Pape, W. Nuijten. *Constraint-Based Scheduling,* 2001.
- P. Laborie. *Algorithms for propagation of resource constraints in AI planning and scheduling: Existing approaches and new results,* 2003.
- P. Vilim*. Global Constraints in Scheduling*, 2007.

Outline

- RCPSP classic problem statement
- Solving methods overview
  - MILP models
- Network rollout optimization

RCPSP MILP: trade-offs

Method	Pros	Cons
Pseudo-polynomial or extended formulations	Obtain better LP relaxations, early node pruning in the search tree	Increase of the MILP size (number of binary variables, constraints) towards pseudo-polynomial and even exponential sizes (need of column and cut generation techniques)
Compact formulations (polynomial size)	Fast node evaluation, more nodes explored	Need to generate cuts

RCPSP MILP formulations

Compact formulations

- Time-indexed variables;
- Linear-ordering variables → Strict-order or sequencing variables;
- Positional dates and assignment variables → Event-based formulations.

Pseudo-polynomial or extended formulations

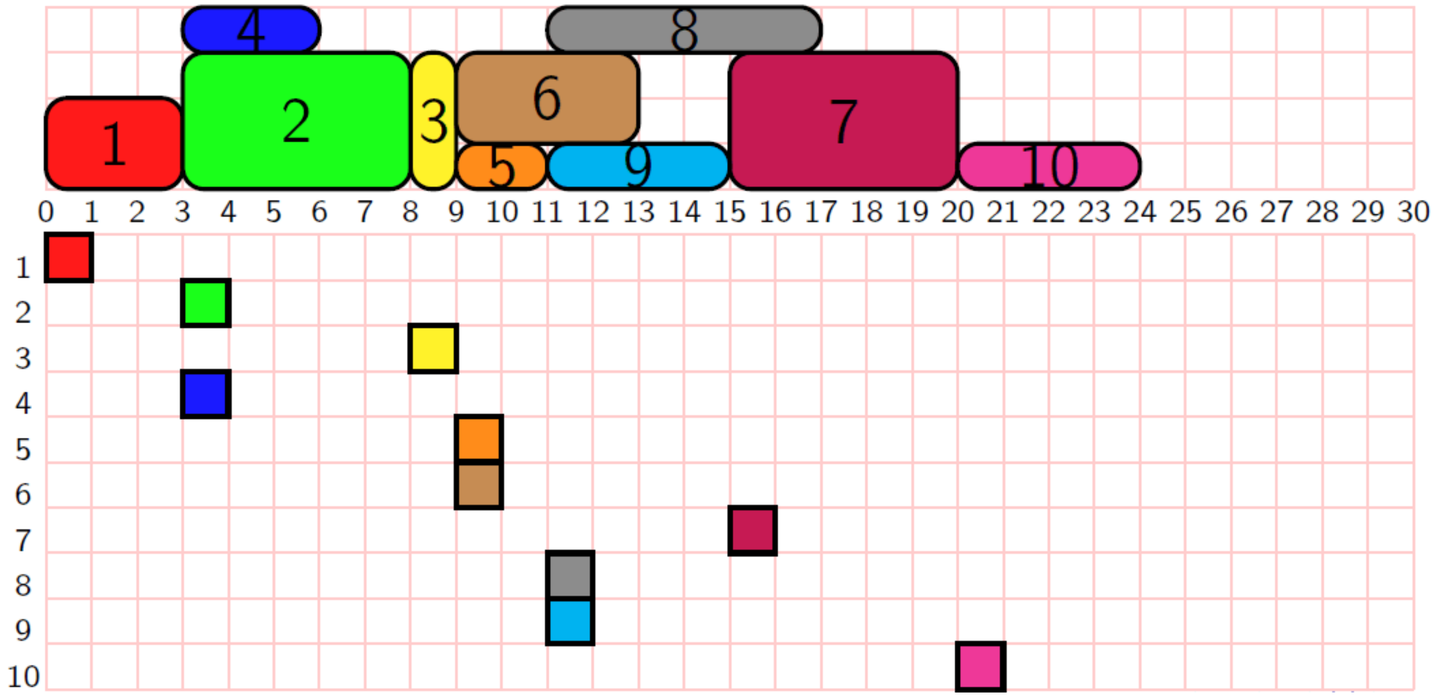
Better relaxations with ... exponential number of variables or constraints.
Cut or column generation techniques are required.

- Minimal forbidden set (MFS) minimal set of activities that cannot be scheduled in parallel (*Hardin, Nemhauser, and Savelsbergh 2008*). Resource constraints can be replaced by MFS. Exp number of inequalities.
- Feasible subsets (FS) a set of activities that can be scheduled in parallel (*Mingozzi et al. 1998*). Resource constraints can be modelled using FS. Exp number of inequalities.

RCPSP MILP formulations

Time-indexed pulse variables

- "Pulse" binary variable $x_{it} = 1 \Leftrightarrow S_i = t$, for $t \in T$
- Pseudo-polynomial number of variables $|N|T$
- Firstly presented by Pritsker, Watters, and Wolfe 1969.



RCPSP MILP formulations

Objective:

$$\min \sum_{t\in[0,T)} t x_{n+1,t}$$

Constraints:

- Resource capacity not violated:

$\forall t \in [0,T), r \in R: \sum_{i\in N} \sum_{\tau=t-p_i+1}^{t} a_{ir}\, x_{i\tau} \le c_r$

- All tasks should be processed:

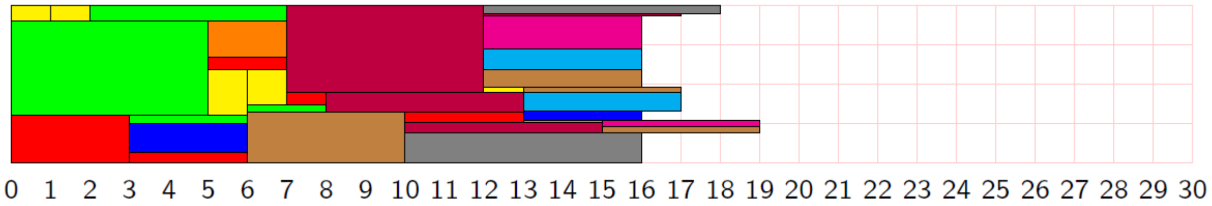
$\forall i \in N: \sum_{t\in[0,T)} x_{it} = 1$

Precedence constraints modeling:

Aggregated:

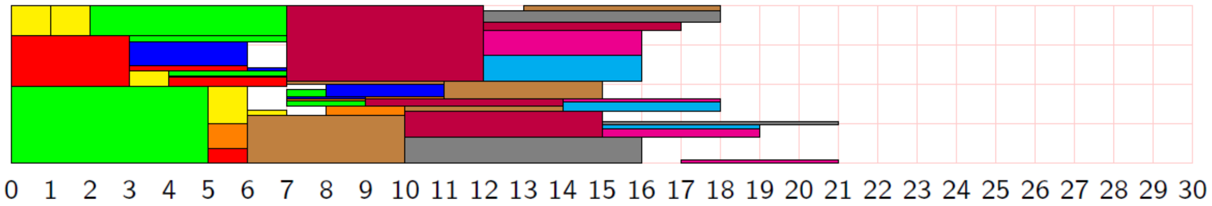
$\sum_{t\in T} t x_{jt} - \sum_{t\in T} t x_{it} \ge p_i$ for each $(i,j) \in E$

LP relaxation:



Disaggregated:

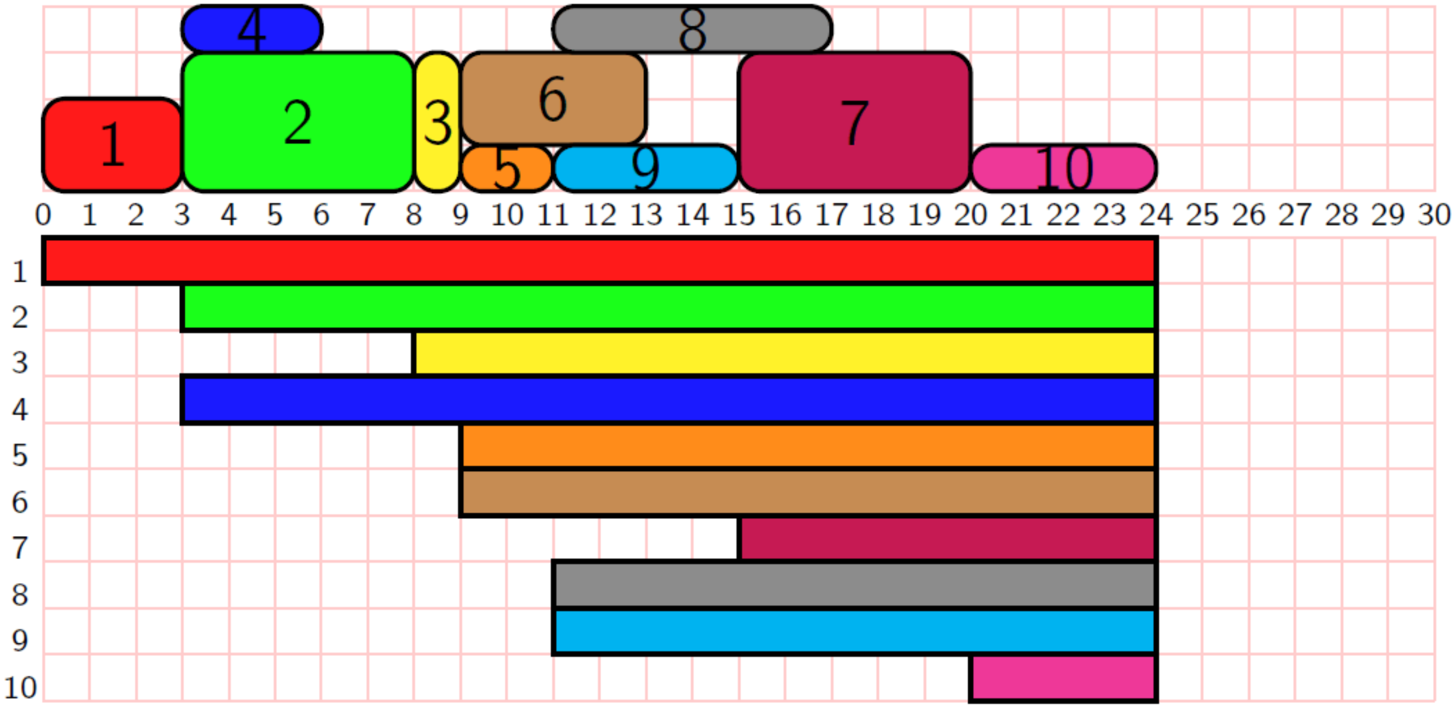
$\sum_{\tau=0}^{t-p_i} x_{i\tau} - \sum_{\tau=0}^{t} x_{j\tau} \ge 0$ for each $(i,j) \in E, t \in [0,T)$



RCPSP MILP formulations

Time-indexed step variables

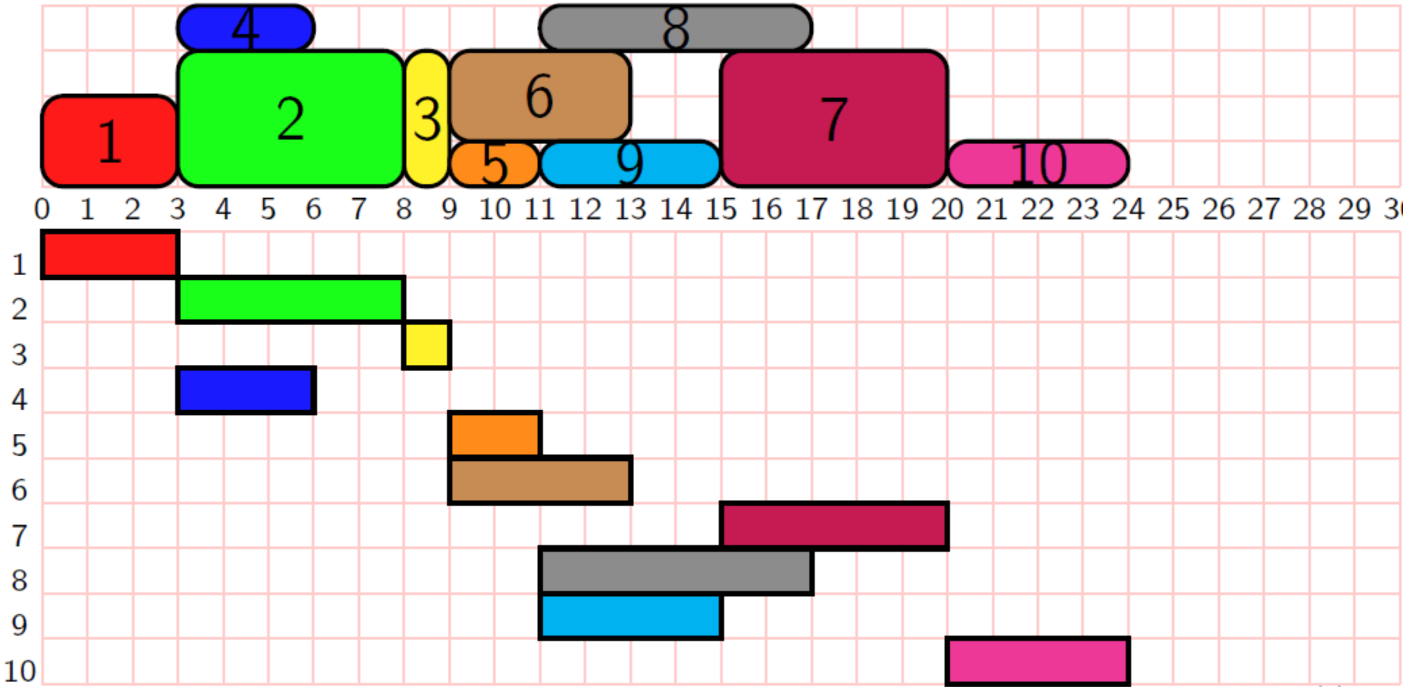
- “Step" binary variable $\xi_{it} = 1 \Leftrightarrow S_i \leq t$, for $t \in T$
- Equivalent to time-indexed formulation
- Firstly presented by Pritsker and Watters 1968.



RCPSP MILP formulations

On/off time-indexed step variables

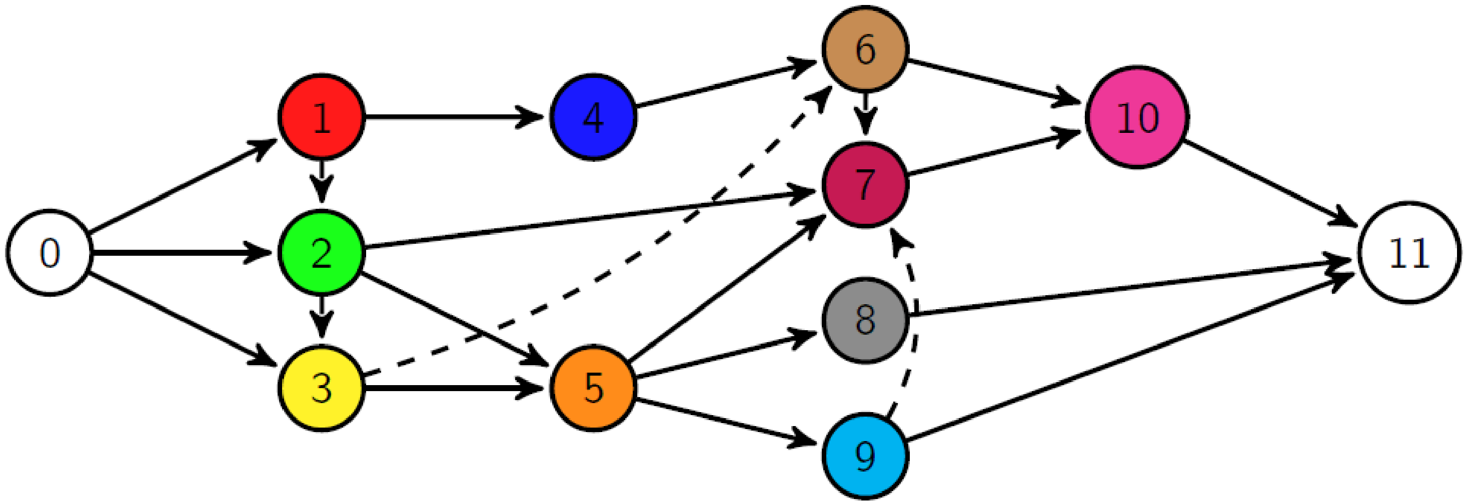
- “On/off" binary variable $\mu_{it} = 1 \Leftrightarrow t \in [S_i, S_i + p_i)$, for $t \in T$
- Introduced by Lawler 1964 for preemptive problems and Klein 2000 for the RCPSP. Improved in Artigues et al. 2013.
- In fact weaker or equivalent to other time-indexed formulations.



RCPSP MILP: formulations

Sequencing or strict ordering

- Principle: adding precedence constraints such that all resource conflicts are resolved.
- Any schedule satisfying these new precedence constraints is feasible.
- Sequencing variable $z_{ij} = 1 \Leftrightarrow S_j \geq S_i + p_i$



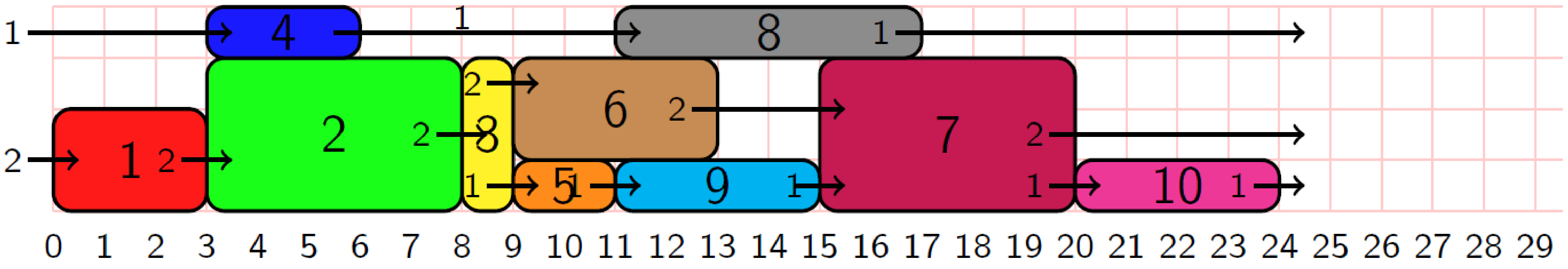
RCPSP MILP formulations

Resource flow variables

- $\phi_{ij}^{r} \geq 0$ – amount of resource $r \in R$ transferred from i to j.
- Enforcing sequencing variables to be compatible with the flow:

$$\phi_{ij}^{r} > 0 \Rightarrow z_{ij} = 1.$$

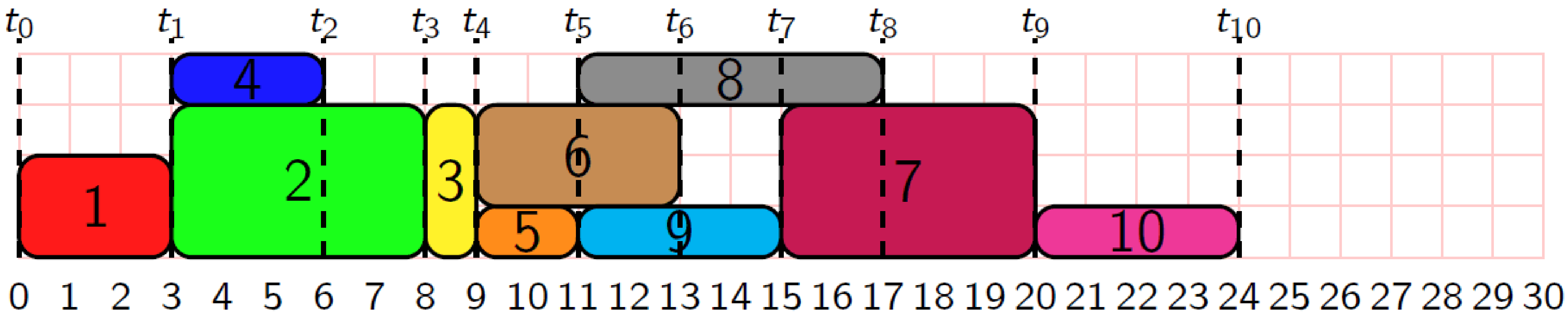
- Compact formulation! $O(|N|^2 R)$ additional continuous variables. Allows to replace MFS constraints by $O(|N|^2 R)$ flow constraints (Artigues 2013).



RCPSP MILP formulations

Start and End Event variables

- E – set of events (possible tasks start and end times).
- Start binary assignment variable $a_{ie}^{-} = 1 \Leftrightarrow S_i = t_e$
- End binary assignment variable $a_{ie}^{+} = 1 \Leftrightarrow S_i = t_e$
- $2|N||\mathrm{E}|$ variables
- Example of application of process scheduling Zapata, Hodge, and Reklaitis 2008. On/off variables model presented in Kone et al. 2011.



RCPSP: solving methods

Constraint programming

Efficient statements are depends on used solver and global constraints it accepts. Example: in IBM CP Optimizer model with interval variables and cumulative resource constraints are very efficient.

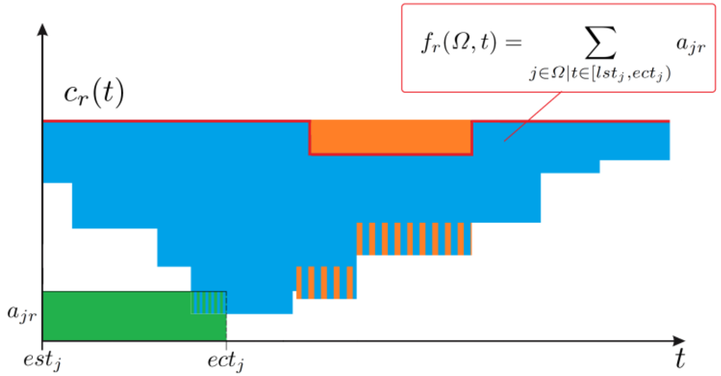
MILP & CP overview

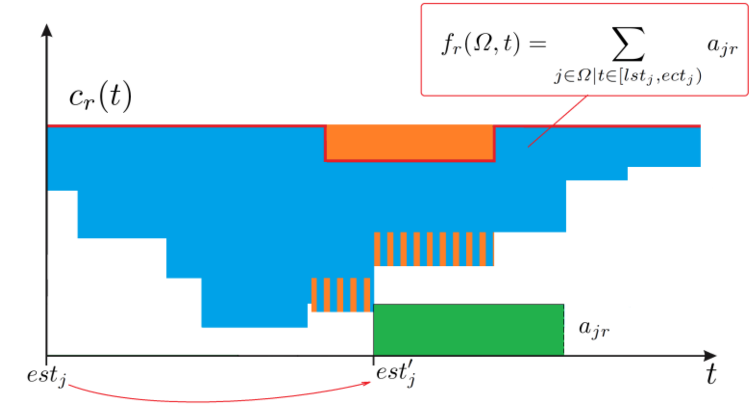
- Time-indexed formulations have the best LP relaxations.
- Compact formulations have poor relaxations, but can be applied for instances with large horizons. The efficiency depends on considered instance:
  - highly disjunctive instances flow-based models are efficient;
  - highly cumulative instances event-based models more efficient.
- For most instances except highly-disjunctive ones CP (i.e. Laborie 2005) and hybrid CP/SAT (Schutt, Feydy, and Stuckey 2013) methods outperform MILP.

RCPSP: pre-solve

Task domain propagation

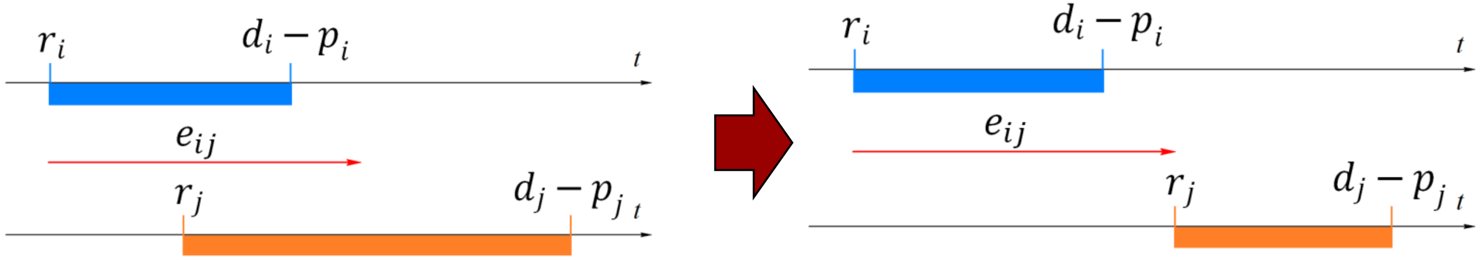
For each task $j \in N$ we can define its processing domains $[est_j, lct_j)$, initially equals to $[r_j, d_j)$. Then apply procedures to make task domains tighter subject to resource, time lags and other task domains. See Schwindt, Zimmermann 2015.

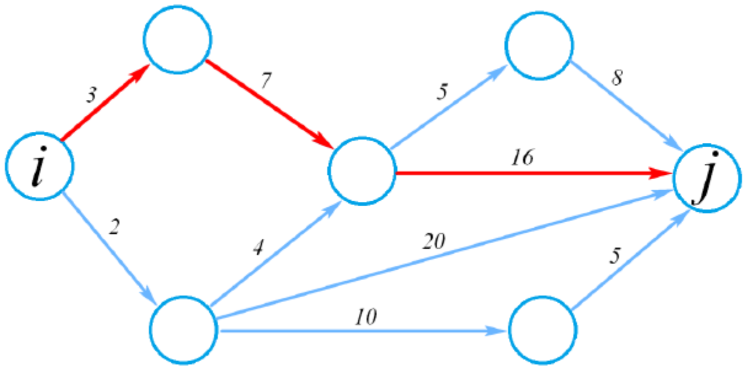




Reconciling precedence relations and task domains

Sometimes task domains strengthening precedence constraint time lags, sometimes vice versa. Polynomial-time procedure can be applied to reconcile domains and time lags.





RCPSP: popular generalizations

Task release times and deadlines

In some statements release time r_j and deadline d_j are defined for task $j \in N$. In classic statement we have $r_j = 0$ and $d_j = T$ for each $j \in N$.

Precedence relations with time lags

Precedence constraints could be generalized by introducing time lag l_{ij} for each constraint $(i, j) \in E$.

Precedence constraint changes to: $S_j \geq S_j + l_{ij}$.

Resource generalizations

- Piecewise-constant renewable resource capacities.
- Multi-skills human resources (MSRCPSP). See De Bruecker et. al. 2015, Almeida et. al. 2019.
- Introducing non-renewable resources (i.e. money).

Objective functions

Uncertainty

Multi-criteria RCPSP

Bi-Criteria problem

Example:

Suppose that for each resource $r \in R$ cost of each available unit is defined by w_r. Let $u_{rt} \in Z_+$ - amount of resource $r \in R$ used by tasks at time t. The objective is to optimize schedule subject to minimal makespan and total resource cost

$$\min C_{\max}, \sum_{r \in R} w_r \, u_{rt}$$

Trade-off

How to consider both objectives?

- Priority
- Combining in objective function
- Pareto-Front

Multi-criteria RCPSP

Priority

If makespan minimization is more important, then

1. Find optimal value $\min C_{\max} = C^*$
2. Add constraint $C_{\max} = C^*$
3. Find $\min \sum_{r \in R} w_r\, u_{rt}$

Combination

Set objective function

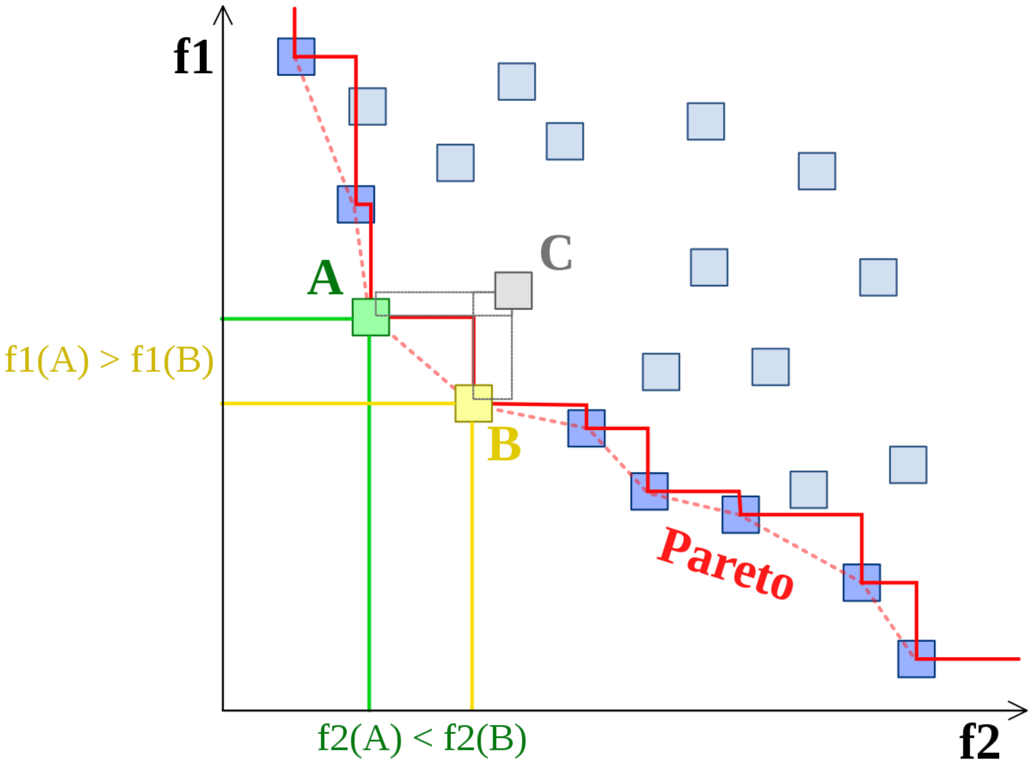
$$\min A \cdot C_{\max} + B \sum_{r \in R} w_r\, u_{rt}$$

where A, B – constants related to components value (importance, weight). Sometimes finding constant values to obtain the desired result is not so easy.

Pareto front

Find the set of solutions not majored by others.

Pareto front



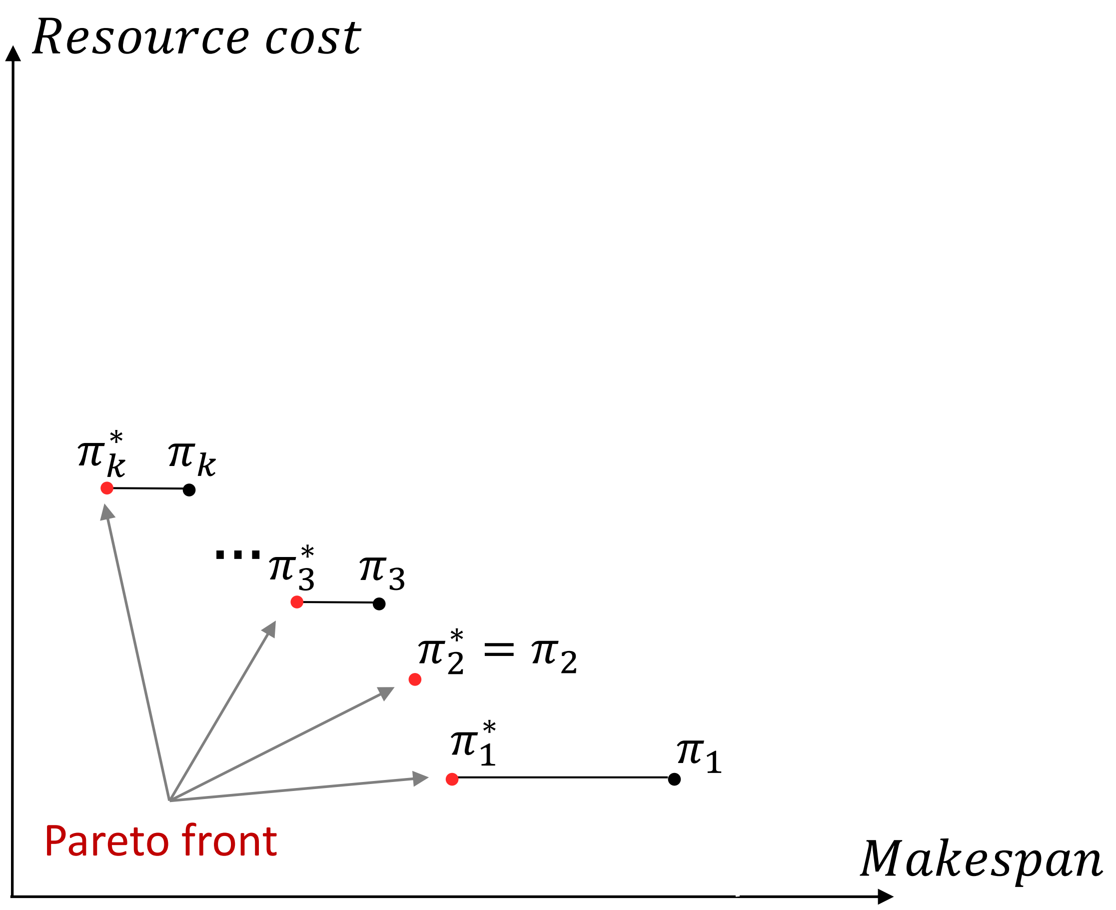
Multi-criteria RCPSP

Enumerate schedules in Pareto front Π^*

1. Set $i = 1, \Pi^* = \emptyset$
2. Look for solution π_i with respect to objective $\min \sum_{r \in R} w_r \, u_{rt}$.
   a) If solution found, look for solution π_i^* with respect to $\min C_{\max}$ and $\sum_{r \in R} w_r \, u_{rt} = \sum_{r \in R} w_r \, u_{rt}(\pi_i)$. Add π_i^* to Π^*.
   b) Otherwise return Π^*.
3. Set $T = C_{\max}(\pi_i) - 1$. Increment i, go to step 2.

Analysis

- On step we obtain solution with better makespan and worse resource cost than on previous one.
- Number of points in Pareto front depends on the problem.
- Integrality of at least one criterion is very important for set enumeration algorithms.



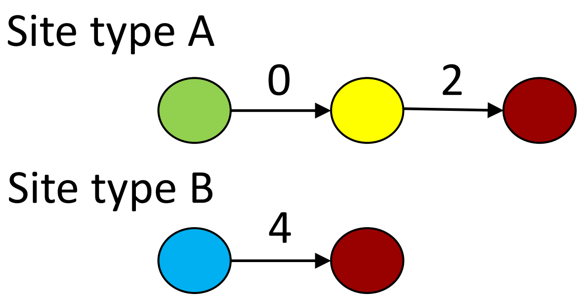
Outline

- RCPSP classic problem statement
- Solving methods overview
  - MILP models
- Network rollout optimization

Network rollout optimization

Problem statement

- There is a set of sites types I to build up in T days.
- For each site $k \in I$ type there is a set of sites U_k on which a chain of tasks (workflow) should be completed subject to time lags. Won the sites of the same type are equivalent. Set of all tasks is denoted by N.
- Each task $j \in N$ can be processed by work team $w \in W_{s_j}$ with defined skill s_j in p_j days. Each team has one skill.
- Some tasks $j \in N$ require non-renewable resource (material) $m_j \in M$ at the start time of their processing. For each non-renewable resource replenishment times $H_m = \{t_{1m}, \dots, T\}$ and amounts $G_m = \{g_{t_1 m}, \dots, 0\}$ are given (including zero - replenishment at time T).
- For each work team of the set W the costs of task processing $cost_{jw}$ are defined for all tasks which this.

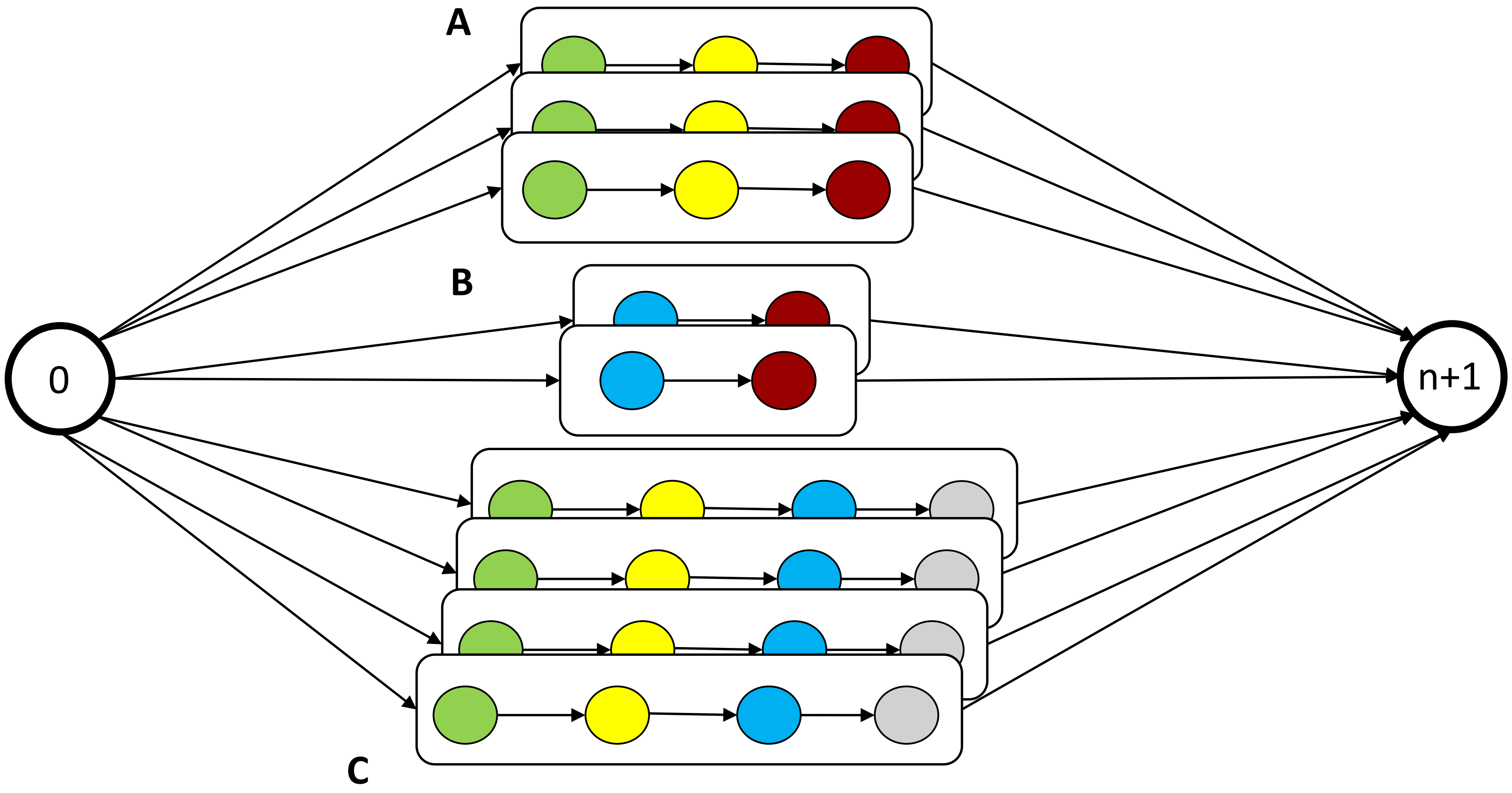


Goal

Process all tasks subject to all resource and time lag constraints with minimal total cost.

Network rollout optimization: RCPSP

Precedence relations series-parallel graph



Network rollout optimization: MILP model

Classic RCPSP MILP formulations application weaknesses

- Thousands of tasks for industry problem instances.
- Number of variables is $|W|$ times larger because of task to team assignment.
- Larger number of variables means larger number of more complicated constraints.
- Precedence relations graph don’t enforce good domain propagation because of short critical path.
- Resource capacity is enough to process single task at any time. No direct domain propagation by resource constraints.

Network rollout optimization: MILP model

RCPSP MILP formulations adaptation

Tasks aggregation

Tasks of the same site types with the same ordinal number are equivalent.
Let $N_k \subseteq N$ – subset of equivalent tasks (type $k \in K$).

Decomposition

The problem can be splat into two stages:

1. Volumetric scheduling. For each task type $i \in K$ and team $w \in W$ find the number of tasks starting processing at time $t \in [0, T)$.
2. Tasks assignment. Find start times of each task.

Statement: optimal solution of the stage two can be found by greedy algorithm.

Network rollout optimization: MILP model

Variables:

- $q_{iwt} \in [0, \min\{|N_i|, |W_{s_i}|\}]$ – number of tasks of type i started by teams with the skill s at day t.
- $x_{n+1,t} \in \{0,1\}$ – if task n+1 started or not at time t.

Objective: cost minimization

$$\min \sum_{i \in K} \sum_{w \in W_{s_i}} \sum_{t \in [0,T)} q_{iwt}$$

Constraints:

- Teams capacity not violated:

$$\forall t \in [0,T), s \in S: \sum_{i \in N} \sum_{w \in W_{s_i}} \sum_{\tau = t - p_i + 1}^{t} q_{iw\tau} \leq |W_{s_i}|$$

Network rollout optimization: MILP model

- All tasks should be processed:

$$\forall i \in K: \sum_{w \in W_{s_i}} \sum_{t=0}^{T-p_j} q_{iwt} = |N_i|$$

- Precedence relations should be satisfied (disaggregated):

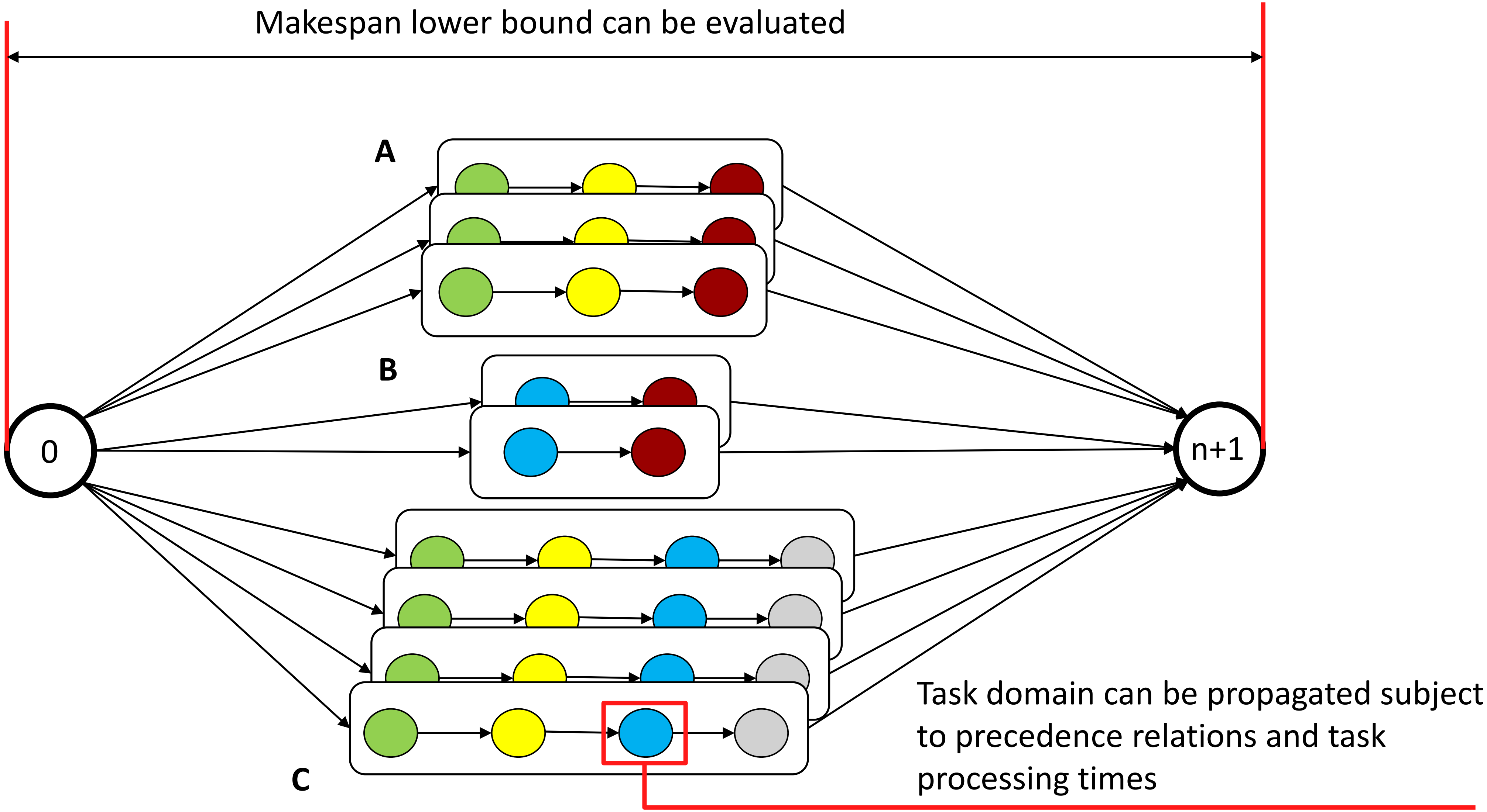
$$\forall (i,j) \in E, t \in [0,T): \sum_{w \in W_{s_i}} \sum_{\tau=0}^{t-p_i} q_{iw\tau} - \sum_{w \in W_{s_j}} \sum_{\tau=0}^{t} q_{j\tau} \geq 0$$

$$\forall (i, n+1) \in E, t \in [0,T): \sum_{w \in W_{s_i}} \sum_{\tau=0}^{t-p_i} q_{i\tau} - \sum_{\tau=0}^{t} |N_i| x_{n+1,\tau} \geq 0$$

- There are enough renewable resources:

$$\forall m \in M, \qquad t \in H_m \sum_{j \in N | m_j = m} \sum_{w \in W_{s_i}} \sum_{\tau=0}^{t} q_{j\tau} \leq \sum_{\tau < t} g_{\tau m}$$

Network rollout optimization: pre-solves & cuts



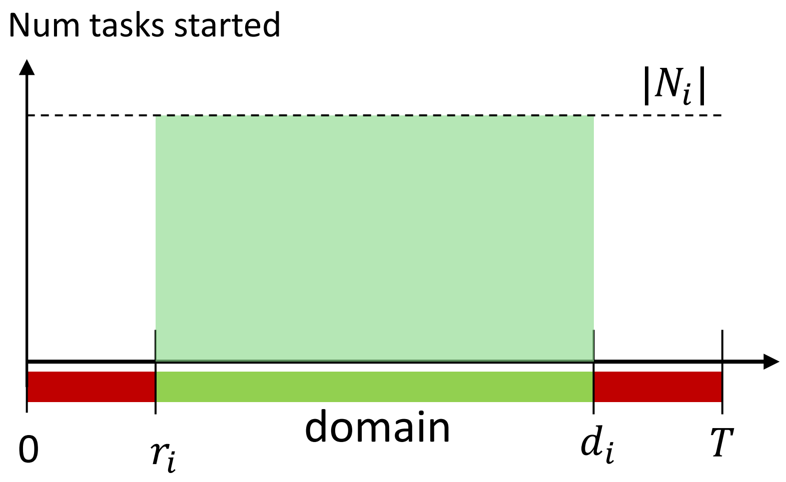
Network rollout optimization: pre-solves & cuts

Variable domain propagation

RCPSP task (of the type $i \in K$) domain propagation techniques could detect $r_i, d_i \in [0, T)$ such that holds:

$$\sum_{w \in W_{s_i}} \sum_{t=0}^{r_i - 1} q_{iwt} = 0$$

$$\sum_{w \in W_{s_i}} \sum_{t=d_i-p_i+1}^{T-p_i+1} q_{iwt} = 0 \Leftrightarrow \sum_{w \in W_{s_i}} \sum_{t=0}^{d_i-p_i+1} q_{iwt} = |N_i|$$



This can be improved by evaluating Q_{iwt}^{LB} and Q_{iwt}^{UB} – upper and lower bounds on total number of tasks of the type $i \in K$ which starts processing by team $w \in W$ in time interval $[0, t]$.

Following cuts could be added to the statement:

$$Q_{iwt}^{LB} \leq \sum_{w \in W_{s_i}} \sum_{\tau=0}^{t} q_{iwt} \leq Q_{iwt}^{UB}$$