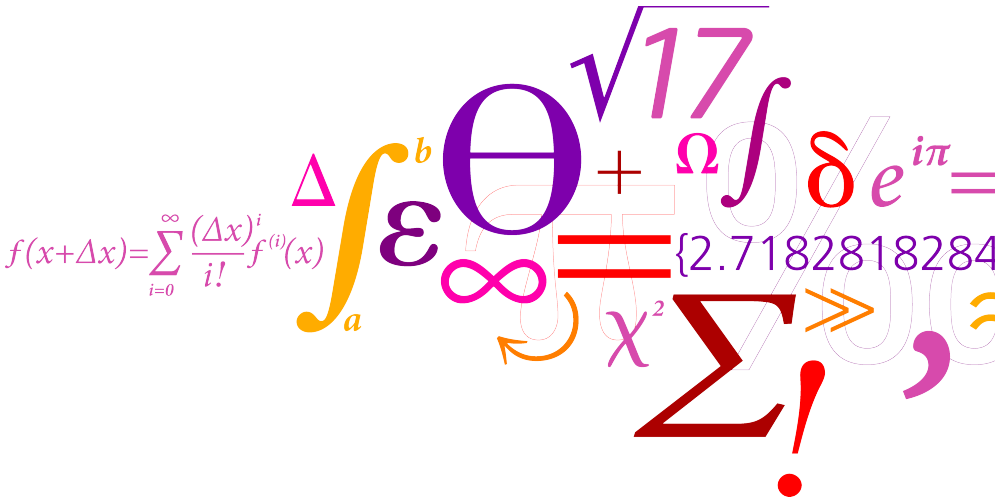
Manpower Planning: Task Scheduling

Anders Høeg Dohn

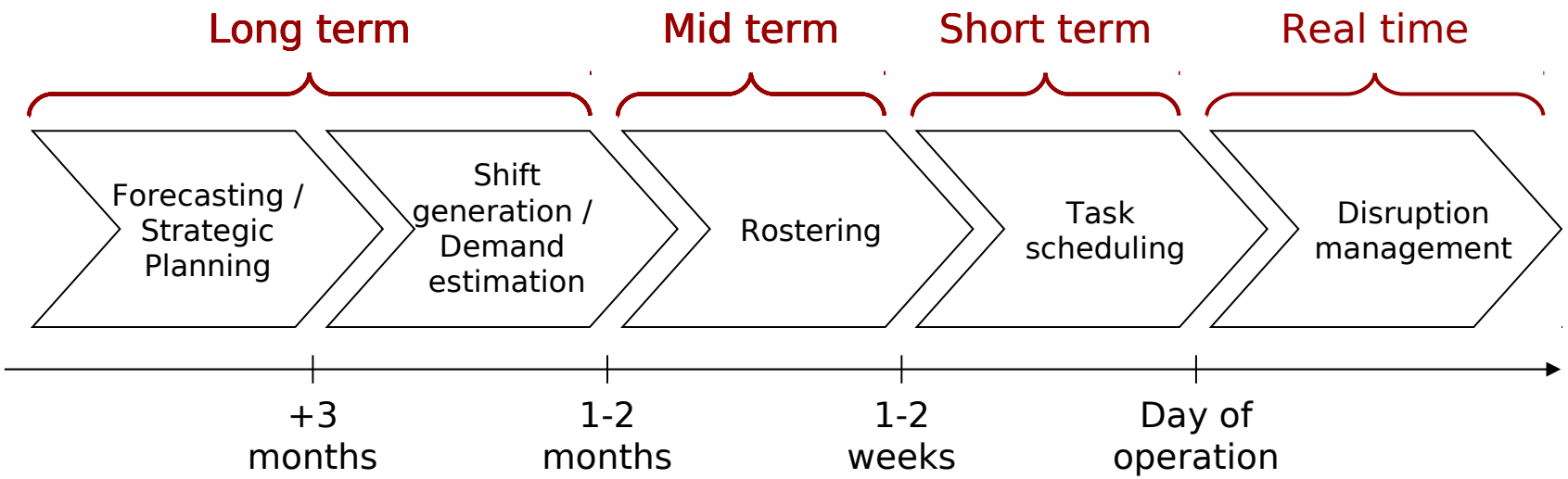


DTU Management Engineering
Department of Management Engineering

Scope

- During these lectures I will:
  - Go over some of the practical problems encountered in manpower planning.
    - Rostering
    - Task Scheduling
  - Propose models that can be used to solve these problems, i.e. present case studies of these methods.
    - Integer Programming
    - Set Partitioning Formulations
    - Column Generation
    - Branch & Price

Task Scheduling



- Allocate employees to tasks
- Assuming that:
  - The roster is fixed
  - The set of tasks is fixed

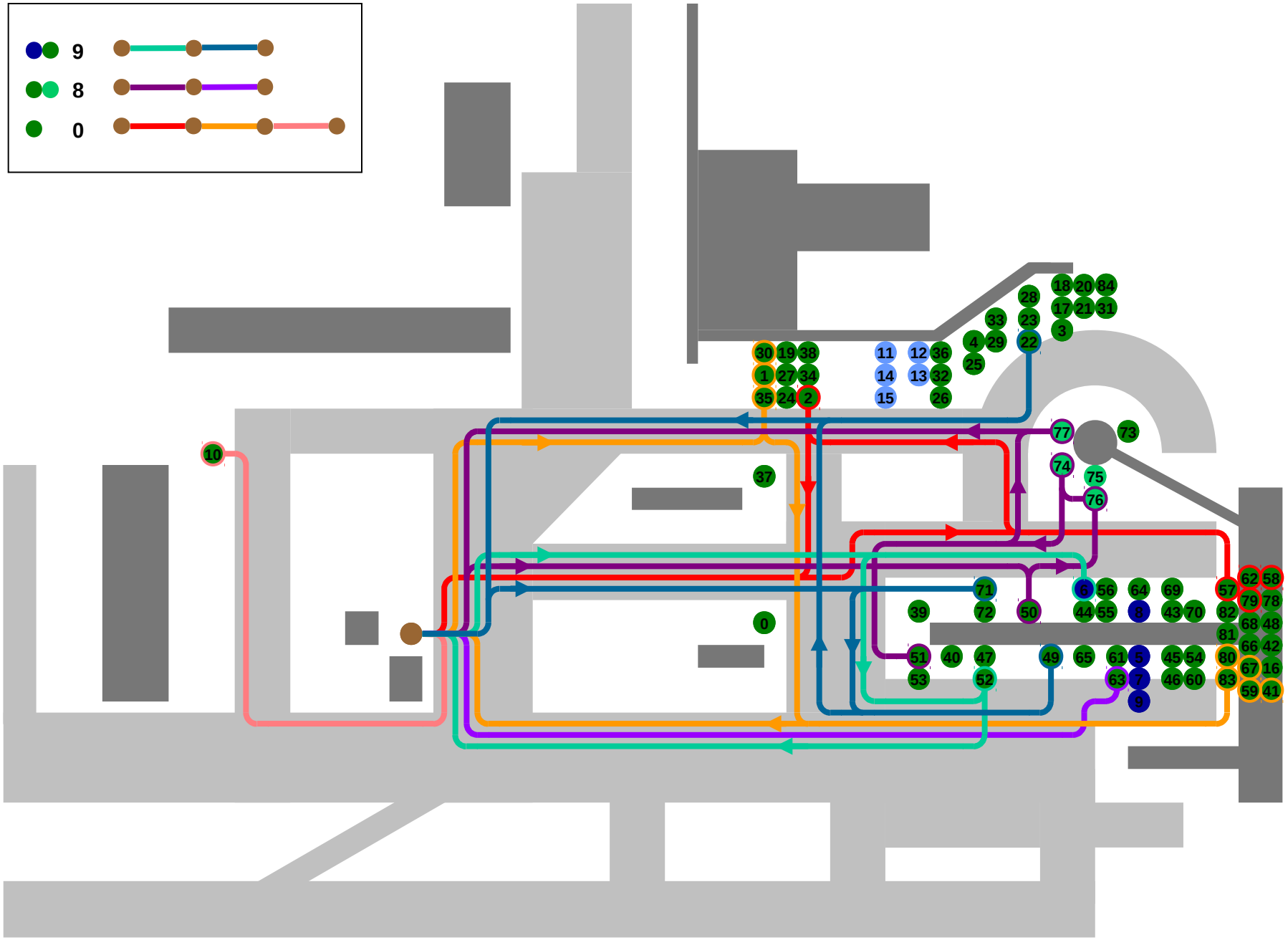
Task Scheduling

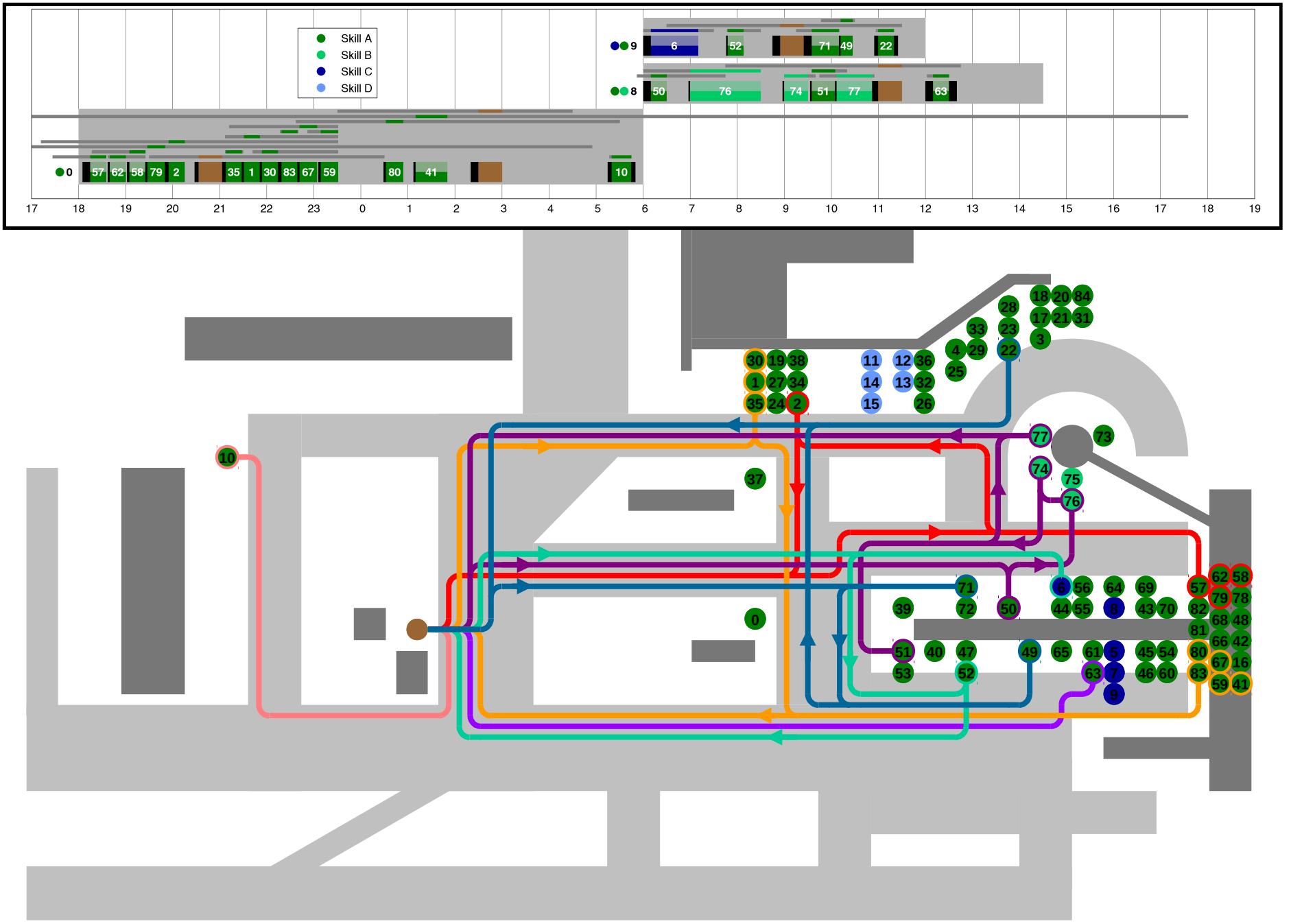
- Task scheduling
  - Consists of:
    - Allocation of tasks
    - Routing of personnel / vehicles between tasks
    - Scheduling of tasks
  - Time horizon is usually at most 24 hours.
  - Shifts may be individual for employees, but have been fixed in advance.
  - May include skills and time windows.
  - May include temporal dependencies between tasks.

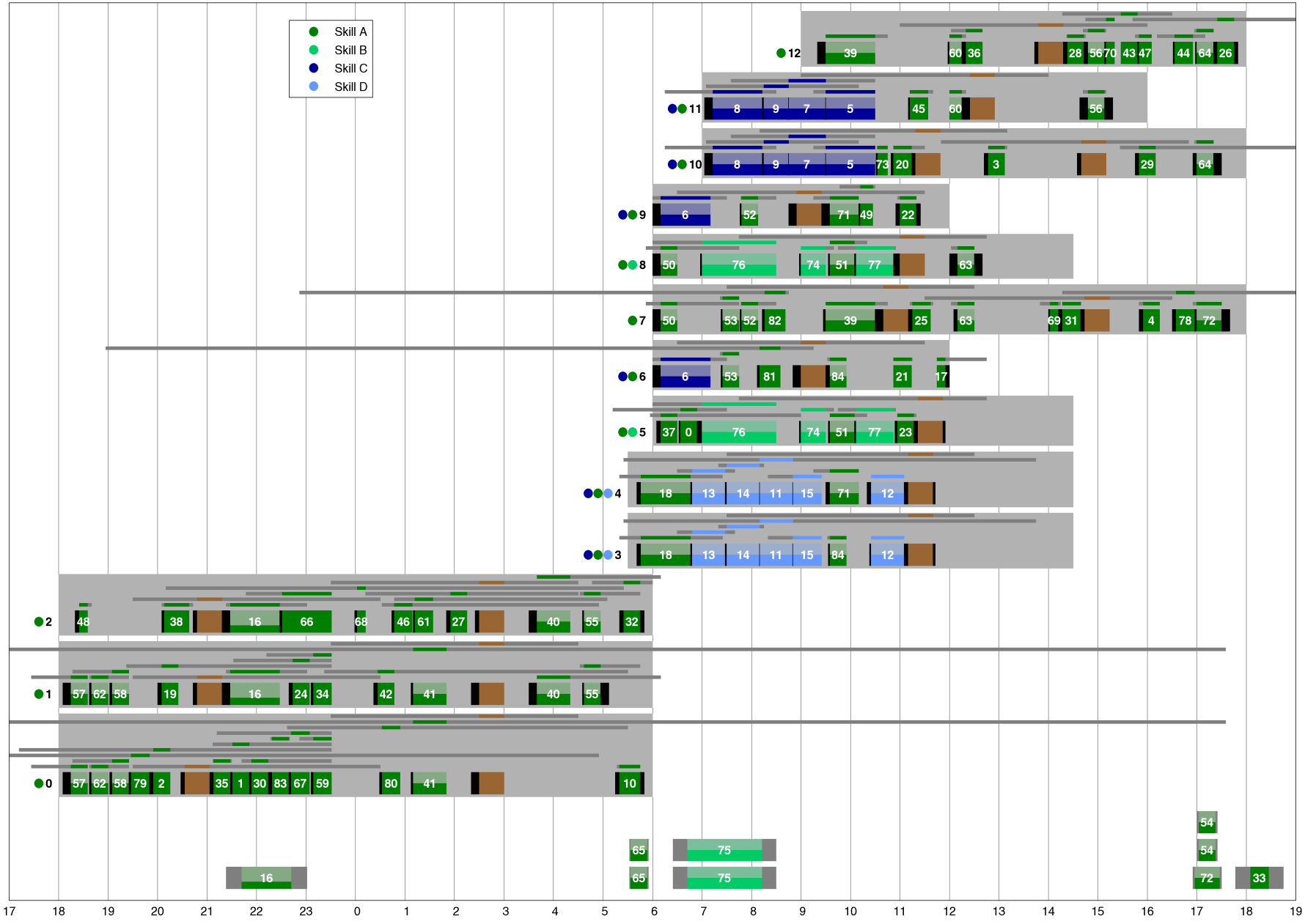
An example

- Example from ground handling in an airport.
- Ground handling tasks:
  - Refueling
  - Luggage handling
  - Garbage collection
  - Cleaning
  - Catering
- Usually outsourced to *ground handling companies*
- A number of teams drive around and carry out tasks at different locations.

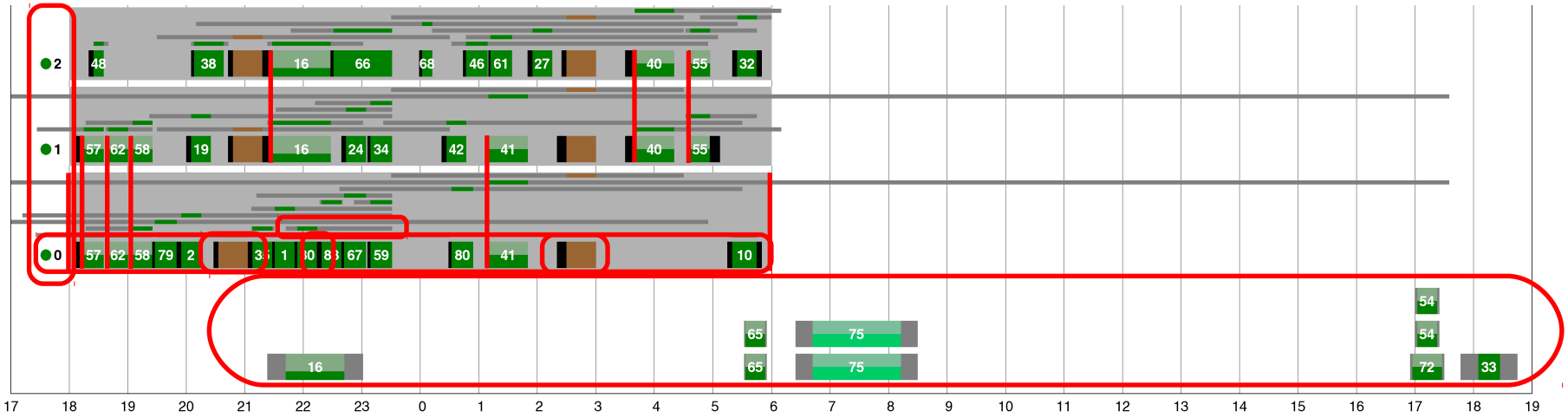

Ordnance Survey
Grid North
PIER 5
NORTH TERMINAL
CARGO CENTRE
PIER 4
MAINT.
AREA
PIER 1
26L


9
8
0
18 20 84
28
17 21 31
33
23
3
4 29
22
30 19 38
11 12 36
25
1 27 34
14 13 32
35 24 2
15
26
77
73
74
75
76
10
37
62 58
79 78
68 48
66 42
67 16
59 41
71
6 56 64 69
57
39
72
50
44 55
8
43 70
82
81
0
51
40
47
49
65
61
5
45 54
80
53
52
63
7
46 60
83
9


Skill A
Skill B
Skill C
Skill D
9
6
52
71
49
22
8
50
76
74
51
77
63
0
57
62
58
79
2
35
1
30
83
67
59
80
41
10
17 18 19 20 21 22 23 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19


Skill A
Skill B
Skill C
Skill D
12 39 60 36 28 56 70 43 47 44 64 26
11 8 9 7 5 45 60 56
10 8 9 7 5 73 20 3 29 64
9 6 52 71 49 22
8 50 76 74 51 77 63
7 50 53 52 82 39 25 63 69 31 4 78 72
6 6 53 81 84 21 17
5 37 0 76 74 51 77 23
4 18 13 14 11 15 71 12
3 18 13 14 11 15 84 12
2 48 38 16 66 68 46 61 27 40 55 32
1 57 62 58 19 16 24 34 42 41 40 55
0 57 62 58 79 2 35 1 30 83 67 59 80 41 10
54
65 75
54
16
65 75
72 33
17 18 19 20 21 22 23 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Problem Description



Master problem

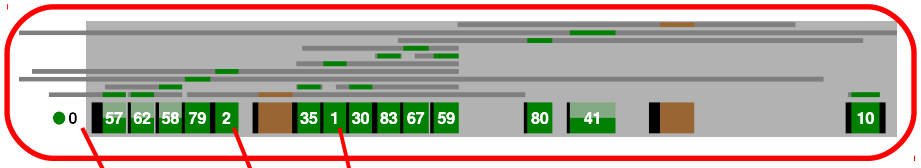
- Minimize the number of unassigned tasks
- Each task must be assigned to exactly one volunteer.
- Temporal dependencies exist between tasks.

Subproblem

- Teams must respect the skill requirement of tasks.
- Teams are only assigned to tasks during their working hours and only to one task at a time.
- Travel times between tasks must be respected.
- A task must be scheduled within its time window.
- Teams must be given the correct amount of breaks.

Column Generation: Restricted Master Problem

- Linear Programming model
- One roster is one column
- One constraint for each task
- One constraint for each team
- Solve linear programming model
  - Minimize number of unassigned tasks
  - May give fractional solution
  - Temporal dependencies between tasks disregarded (for now)

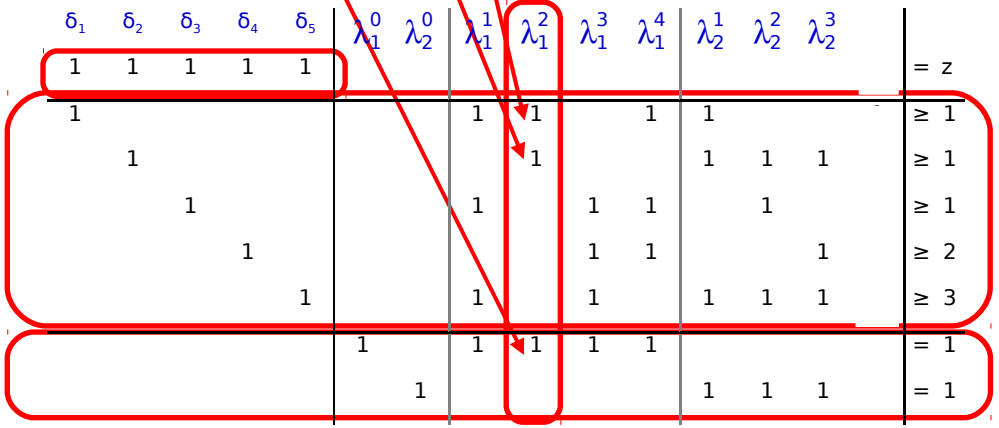


$$\min \sum_{i \in C} \delta_i$$

$$\delta_i + \sum_{k \in V} \sum_{p \in \mathcal{P}'_k} a^p_{ik} \lambda^p_k \geq r_i \qquad \forall i \in C$$

$$\sum_{p \in \mathcal{P}'_k} \lambda^p_k = 1 \qquad \forall k \in V$$

$$\lambda^p_k \geq 0 \qquad \forall k \in V, \forall p \in \mathcal{P}'_k$$
$$\delta_i \geq 0 \qquad \forall i \in C$$



Column Generation: New columns

How did we get those feasible rosters?

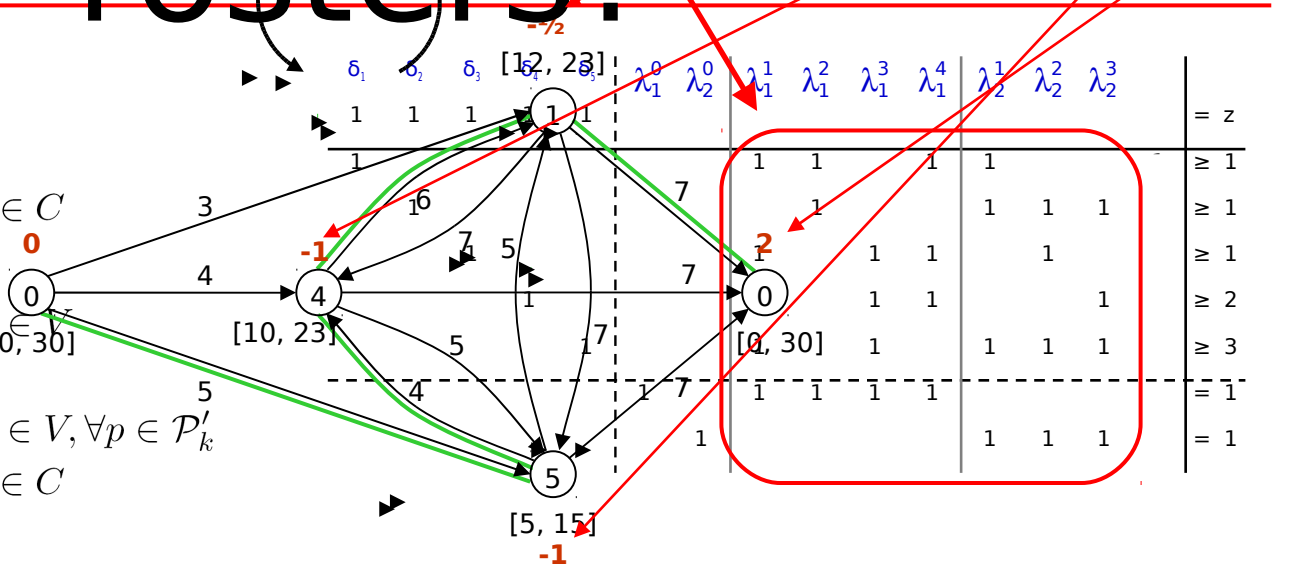
$$\min \sum_{i \in C} \delta_i$$

$$\delta_i + \sum_{k \in V} \sum_{p \in \mathcal{P}'_k} a_{ik}^p \lambda_k^p \geq r_i \qquad \forall i \in C$$

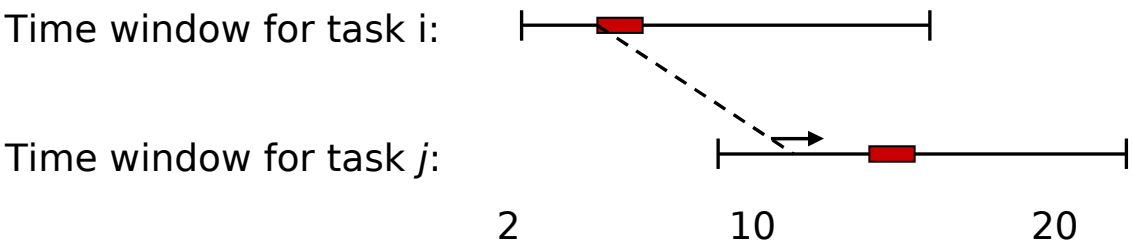
$$\sum_{p \in \mathcal{P}'_k} \lambda_k^p = 1 \qquad \forall k \in V$$

$$\lambda_k^p \geq 0 \qquad \forall k \in V, \forall p \in \mathcal{P}'_k$$

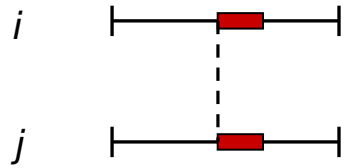
$$\delta_i \geq 0 \qquad \forall i \in C$$



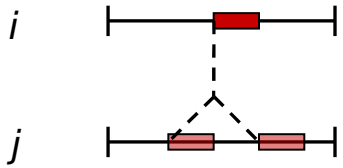
Generalizing Synchronization to Other Temporal Dependencies



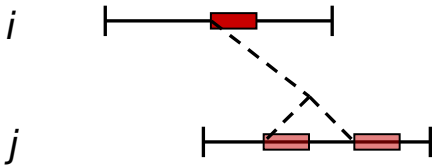
Synchronization:



Overlap:



Min/max gap:



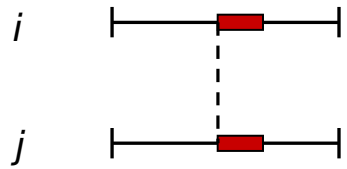
Temporal Dependencies in Practice

- Ground handling in airports
  - Synchronization (Job teaming)
  - Overlap
- Home care crew scheduling
  - Synchronization (Mainly for lifting)
  - Overlap (Lifting)
  - Min and max gap (E.g. medication and laundry)
- Allocation of technicians to service jobs [Li et al. 2005].
- Dial-a-Ride for disabled persons [Rousseau et al. 2003].
- Aircraft fleet assignment and routing [Ioachim et al. 1999].
- Machine scheduling with precedence constraints
  [van den Akker et al. 2006].

The Generalized Precedence Constraint

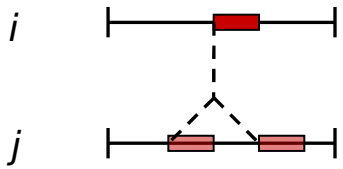
$$t_i + p_{ij} \leq t_j$$

Synchronization:



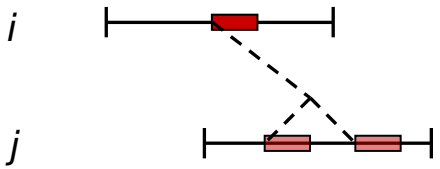
$$p_{ij} = 0 \wedge p_{ji} = 0$$
$$\Rightarrow t_i + 0 \leq t_j \wedge t_j + 0 \leq t_i$$
$$\Leftrightarrow \quad t_j = t_i$$

Overlap:



$$p_{ij} = -dur_j \wedge p_{ji} = -dur_i$$
$$\Rightarrow t_i - dur_j \leq t_j \wedge t_j - dur_i \leq t_i$$
$$\Leftrightarrow t_i - dur_j \leq t_j \leq t_i + dur_i$$

Min/max gap:



$$p_{ij} = mingap \wedge p_{ji} = -maxgap$$
$$\Rightarrow t_i + mingap \leq t_j$$
$$\wedge t_j - maxgap \leq t_i$$
$$\Leftrightarrow t_i + mingap \leq t_j \leq t_i + maxgap$$

Compact formulation

$$\min w_1 \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{N}^k} \sum_{j\in\mathcal{N}^k} c_{ij}^k x_{ij}^k + w_2 \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{N}^k} \delta_i^k x_{ij}^k + \quad w_3 \sum_{i\in\mathcal{C}} \gamma_i y_i$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{N}^k} x_{ij}^k + y_i = 1 \qquad \forall i \in \mathcal{C}$$

$$\sum_{j\in\mathcal{N}^k} x_{ij}^k \leq \rho_i^k \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{C}$$

$$\sum_{j\in\mathcal{N}^k} x_{0^k,j}^k = 1 \qquad \forall k \in \mathcal{K}$$

$$\sum_{i\in\mathcal{N}^k} x_{i,n^k}^k = 1 \qquad \forall k \in \mathcal{K}$$

$$\sum_{i\in\mathcal{N}^k} x_{ih}^k - \sum_{j\in\mathcal{N}^k} x_{hj}^k = 0 \qquad \forall k \in \mathcal{K}, \forall h \in \mathcal{C}$$

$$\alpha_i \sum_{j\in\mathcal{N}^k} x_{ij}^k \leq t_i^k \leq \beta_i \sum_{j\in\mathcal{N}^k} x_{ij}^k \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{C} \cup \{0^k\}$$

$$\alpha_{n^k} \leq t_{n^k}^k \leq \beta_{n^k} \qquad \forall k \in \mathcal{K}$$

$$t_i^k + s_{ij}^k x_{ij}^k \leq t_j^k + \beta_i (1 - x_{ij}^k) \qquad \forall k \in \mathcal{K}, \forall i,j \in \mathcal{N}^k$$

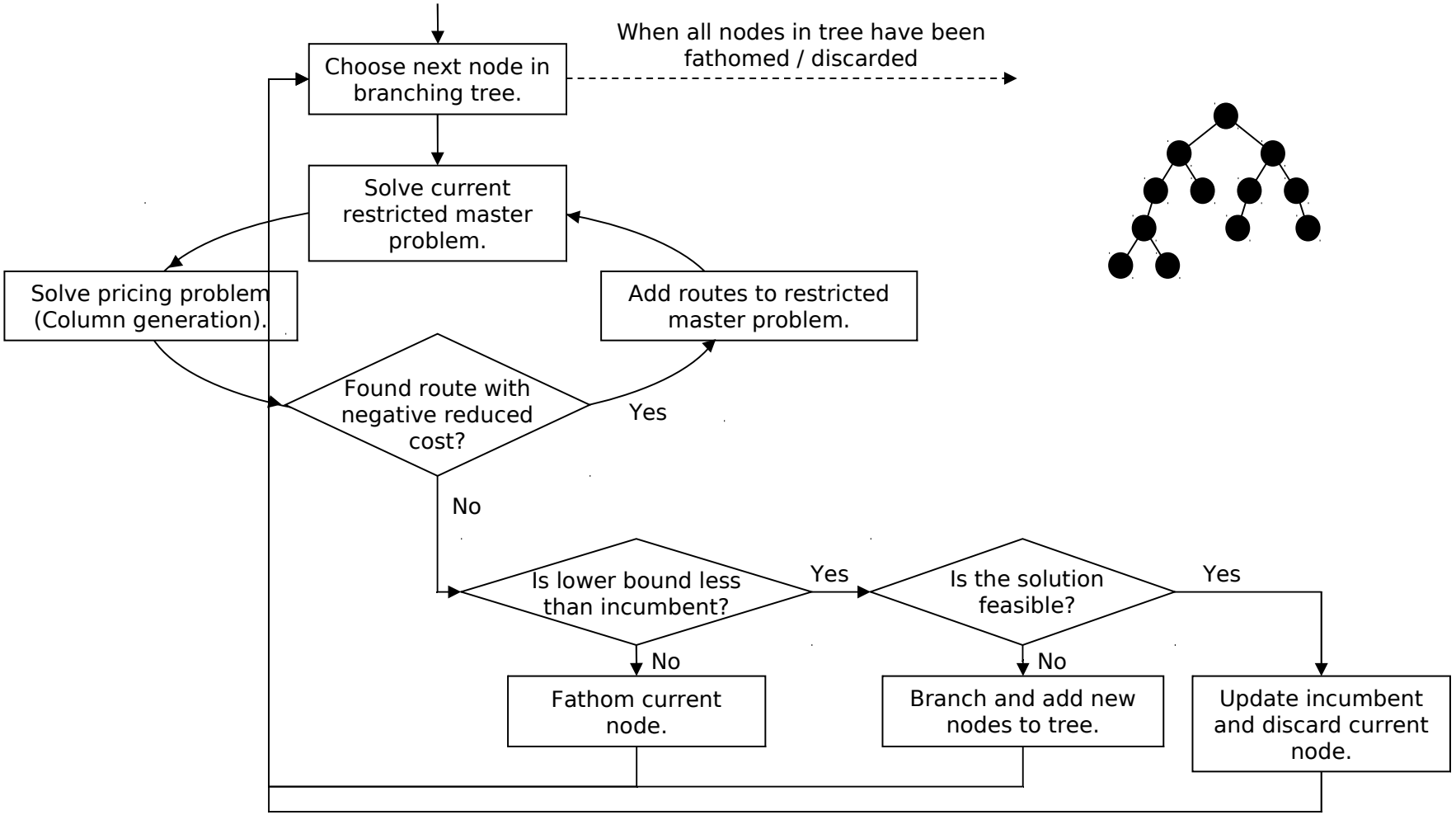
$$\alpha_i y_i + \sum_{k\in\mathcal{K}} t_i^k + p_{ij} \leq \sum_{k\in\mathcal{K}} t_j^k + \beta_j y_j \qquad \forall (i,j) \in \mathcal{P}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall i,j \in \mathcal{N}^k$$

$$t_i^k \in \mathbb{Z}_+ \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}^k$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{C}$$

Branch & Price – Overview



Branch & Price

- Necessary considerations:
  - How do we model and solve the master problem?
  - How do we model and solve the subproblem?
  - How do we ensure integrality in the master problem?

The Master Problem

- Solving the set partitioning problem with generalized precedence constraints:
  - The master problem is a Set Partitioning Problem with additional non-binary constraints.
  - The subproblem is an Elementary Shortest Path Problem with Time Windows and Linear Node Costs.
    - Only the acyclic case has been considered in the literature [Ioachim et al. 1997].
  - Gives a harder subproblem.
  - Leads to highly fractional solutions for the master problem.

The Master Problem

$$\min \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} c_r^k \lambda_r^k + \sum_{i \in \mathcal{C}} c_i \Lambda_i$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} a_{ir}^k \lambda_r^k + \Lambda_i = 1 \qquad \forall i \in \mathcal{C}$$

$$\sum_{r \in \mathcal{R}^k} \lambda_r^k = 1 \qquad \forall k \in \mathcal{K}$$

$$\alpha_i \Lambda_i + \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} t_{ir}^k \lambda_r^k + p_{ij} \leq \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} t_{jr}^k \lambda_r^k + \beta_j \Lambda_j \qquad \forall (i,j) \in \mathcal{P}$$

$$\lambda_r^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall r \in \mathcal{R}^k$$

$$\Lambda_i \in \{0,1\} \qquad \forall i \in \mathcal{C}$$

Variables:

$=\lambda_r^k$ { 1 if route *r* is chosen for team *k*; 0 otherwise

$=\Lambda_i$ { 1 if task *i* is uncovered; 0 otherwise

Sets:

C	Tasks
K	Teams / Vehicles
R^k	Routes
P	Temporal Dependencies

The Master Problem

- Solving a time index model.
  - Common in solution of machine scheduling problems.
    - Change the coefficient a_{ir}^{k} to $a_{i\tau r}^{k}$
    - $a_{i\tau r}^{k} = 1$ if team *k* is allocated to task *i* at time τ in route *r*.
  - Each generalized precedence constraint introduces a set of new constraints in the master problem.
  - The master problem becomes a Set Partitioning Problem with a huge amount of constraints.
    - However, it can be proven that the formulation is stronger than the master problem formulation with a continuous time-variable.
    - All constraints cannot be generated a priori:
      Branch & Cut & Price

The Master Problem

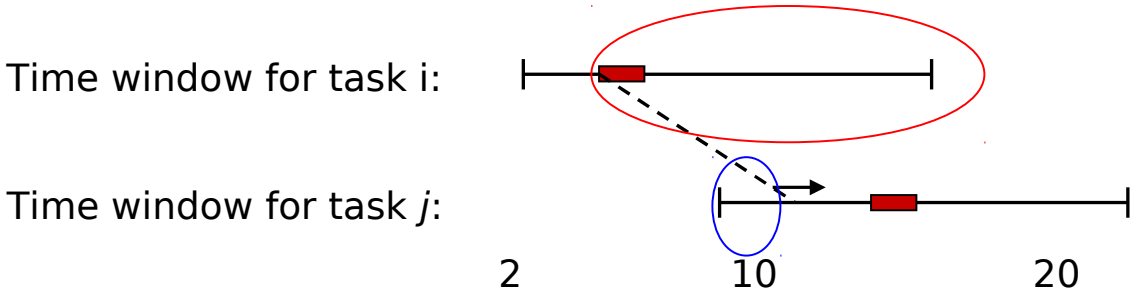
- Solving a time index model.
- Each generalized precedence constraint introduces a set of new constraints in the master problem:

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}'} t^k_{ir}\lambda^k_r + p_{ij} \le \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}'} t^k_{jr}\lambda^k_r \qquad \forall (i,j)\in\mathcal{P}$$

$$\downarrow$$

$$\sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}'}\sum_{\tau'=\tau,\ldots,\beta_0} a^k_{ir\tau'}\lambda^k_r + \sum_{k\in\mathcal{K}}\sum_{r\in\mathcal{R}'}\sum_{\tau'\in\mathcal{T}^p_{\tau ij}} a^k_{jr\tau'}\lambda^k_r \le 1 \qquad \forall (i,j)\in\mathcal{P}, \forall \tau\in\mathcal{T}$$

$$\mathcal{T}^p_{\tau ij} = \{\alpha_0,\ldots,\min\{\beta_0,\tau+p_{ij}-1\}\}$$



The Master Problem

- Relaxing the generalized precedence constraints:
  - The master problem is a Set Partitioning Problem.
  - The subproblem is an Elementary Shortest Path Problem with Time Windows.
  - Temporal dependencies are enforced by branching.
- This is the simplest approach to solving the set partitioning problem with generalized precedence constraints.
  - We use this approach in the following.

The Master Problem

$$\min \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} c_r^k \lambda_r^k + \sum_{i \in \mathcal{C}} c_i \Lambda_i$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} a_{ir}^k \lambda_r^k + \Lambda_i = 1 \qquad \forall i \in \mathcal{C}$$

$$\sum_{r \in \mathcal{R}^k} \lambda_r^k = 1 \qquad \forall k \in \mathcal{K}$$

$$\alpha_i \Lambda_i + \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} t_{ir}^k \lambda_r^k + p_{ij} \leq \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}^k} t_{jr}^k \lambda_r^k + \beta_j \Lambda_j \qquad \forall (i,j) \in \mathcal{P}$$

$$\lambda_r^k \in \{0,1\} \qquad \forall k \in \mathcal{K}, \forall r \in \mathcal{R}^k$$

$$\Lambda_i \in \{0,1\} \qquad \forall i \in \mathcal{C}$$

Variables:

- $=\lambda_r^k$ { 1 if route *r* is chosen for team *k*; 0 otherwise
- $=\Lambda_i$ { 1 if task *i* is uncovered; 0 otherwise

Sets:

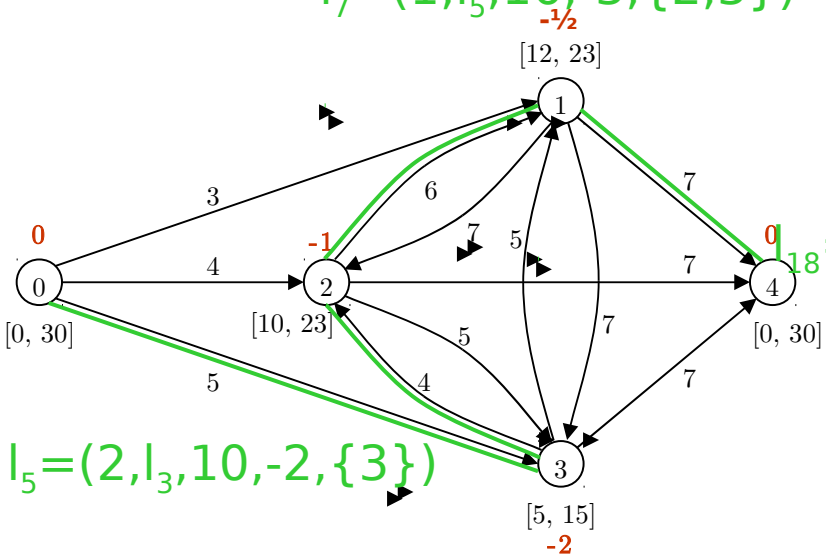
- *C* Tasks
- *K* Teams / Vehicles
- R^k Routes
- *P* Temporal Dependencies

Branch & Price

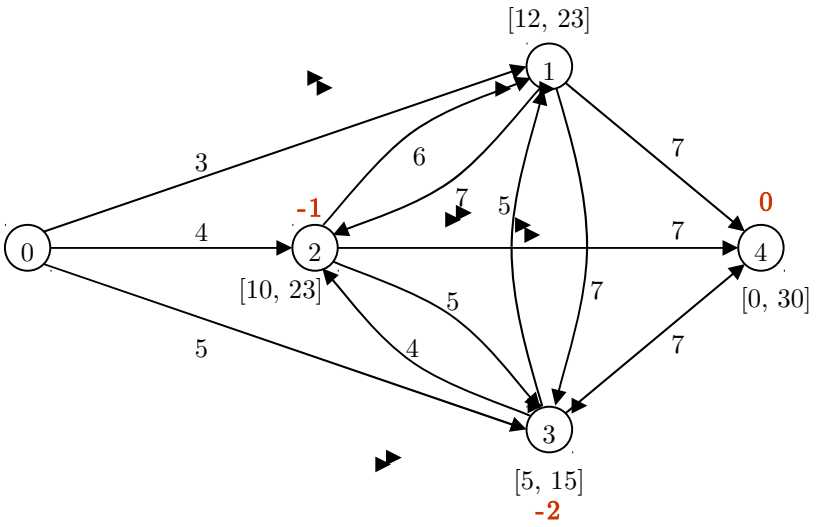
- Necessary considerations:
  - How do we model and solve the master problem?
  - How do we model and solve the subproblem?
  - How do we ensure integrality in the master problem?

The label setting algorithm

- Label: l = $(v, p, \tilde{t}, \tilde{c}, W)$
- Pseudo code:
  1. Create initial label: l_0
  2. Pick label with minimum *t*
  3. Extend label to all possible successors
  4. If more unprocessed labels exist: Go to 2
  5. Return best path



The label setting algorithm



Extend	New label: $(v, p, \tilde{t}, \tilde{c}, W)$	U	$ULPQ$
	$l_0 = (0, l_\varnothing, 0, 0, \varnothing)$	$\varnothing$	$[l_0]$
l_0:	$l_1 = (1, l_0, \mathbf{12}, 0, \varnothing)$	$\{3\}$	$[l_1]$
	$l_2 = (2, l_0, \mathbf{10}, 0, \varnothing)$	$\varnothing$	$[l_2, l_1]$
	$l_3 = (3, l_0, 5, 0, \varnothing)$	$\varnothing$	$[l_3, l_2, l_1]$
l_3:	$l_4 = (1, l_3, \mathbf{12}, -2, \{3\})$	$\{3\}$	$[l_2, l_4, l_1]$
	$l_5 = (2, l_3, \mathbf{10}, -2, \{3\})$	$\{3\}$	$[l_5, l_2, l_4, l_1]$
	$l_6^* = (4, l_3, 12, -2, \{3\})$	$\{3\}$	$[l_5, l_2, l_4, l_1]$
l_5:	$l_7 = (1, l_5, 16, -3, \{2, 3\})$	$\{2, 3\}$	$[l_2, l_4, l_1, l_7]$
	$l_8^* = (4, l_5, 17, -3, \{2, 3\})$	$\{2, 3\}$	$[l_2, l_4, l_1, l_7]$
l_2:	$l_9 = (1, l_2, 16, -1, \{2\})$	$\{2, 3\}$	$[l_4, l_1, l_7, l_9]$
	$l_{10} = (3, l_2, 15, -1, \{2\})$	$\{2\}$	$[l_4, l_1, l_{10}, l_7, l_9]$
	$l_{11}^* = (4, l_2, 17, -1, \{2\})$	$\{2, 3\}$	$[l_4, l_1, l_{10}, l_7, l_9]$
l_4:	$l_{12} = (2, l_4, 19, -\frac{5}{2}, \{1, 3\})$	$\{1, 3\}$	$[l_1, l_{10}, l_7, l_9, l_{12}]$
	$l_{13}^* = (4, l_4, 19, -\frac{5}{2}, \{1, 3\})$	$\{1, 3\}$	$[l_1, l_{10}, l_7, l_9, l_{12}]$
l_1:	$l_{14} = (2, l_1, 19, -\frac{1}{2}, \{1\})$	$\{1, 3\}$	$[l_{10}, l_7, l_9, l_{12}, l_{14}]$
	$l_{15}^* = (4, l_1, 19, -\frac{1}{2}, \{1\})$	$\{1, 3\}$	$[l_{10}, l_7, l_9, l_{12}, l_{14}]$
l_{10}:	$l_{16} = (1, l_{10}, 20, -3, \{2, 3\})$	$\{2, 3\}$	$[l_7, l_9, l_{12}, l_{14}, l_{16}]$
	$l_{17}^* = (4, l_{10}, 22, -3, \{2, 3\})$	$\{2, 3\}$	$[l_7, l_9, l_{12}, l_{14}, l_{16}]$
l_7:	$l_{18}^* = (4, l_7, 23, -\frac{7}{2}, \{1, 2, 3\})$	$\{1, 2, 3\}$	$[l_9, l_{12}, l_{14}, l_{16}]$
l_9:	$l_{19}^* = (4, l_9, 23, -\frac{3}{2}, \{1, 2\})$	$\{1, 2, 3\}$	$[l_{12}, l_{14}, l_{16}]$
l_{12}:	$l_{20}^* = (4, l_{12}, 26, -\frac{7}{2}, \{1, 2, 3\})$	$\{1, 2, 3\}$	$[l_{14}, l_{16}]$
l_{14}:	$l_{21}^* = (4, l_{14}, 26, -\frac{3}{2}, \{1, 2\})$	$\{1, 2, 3\}$	$[l_{16}]$
l_{16}:	$l_{22}^* = (4, l_{16}, 27, -\frac{7}{2}, \{1, 2, 3\})$	$\{1, 2, 3\}$	$[]$

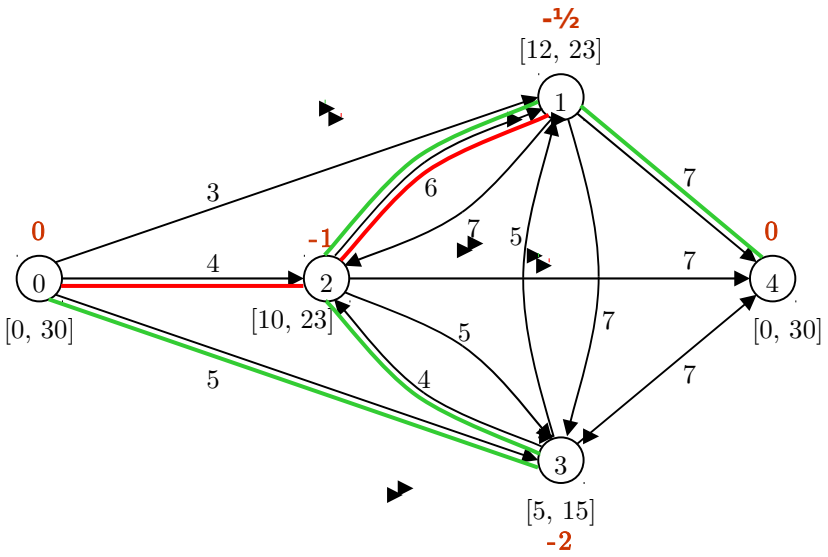
Dominance

- The idea: By applying dynamic programming techniques, labels can be removed while still ensuring optimality:

$$l_a \preceq l_b \quad \Leftrightarrow \quad \begin{array}{rcl} v_a & = & v_b \\ \tilde{t}_a & \leq & \tilde{t}_b \\ \tilde{c}_a & \leq & \tilde{c}_b \\ U_a & \subseteq & U_b \end{array}$$

l_7=(1,l_5,16,-3,{2,3}) $\Rightarrow l_a$ l_b

l_9=(1,l_2,16,-1,{2,3})



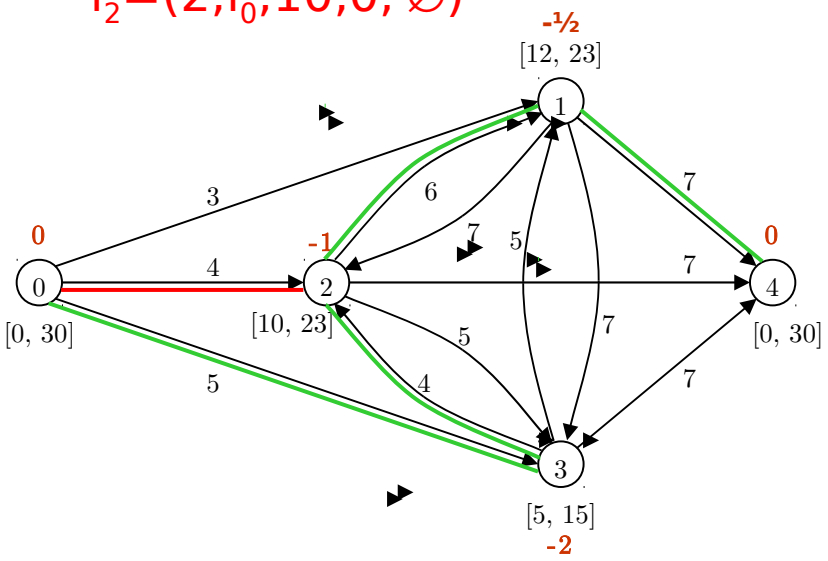
Dominance

- The dominance may be strengthened further:

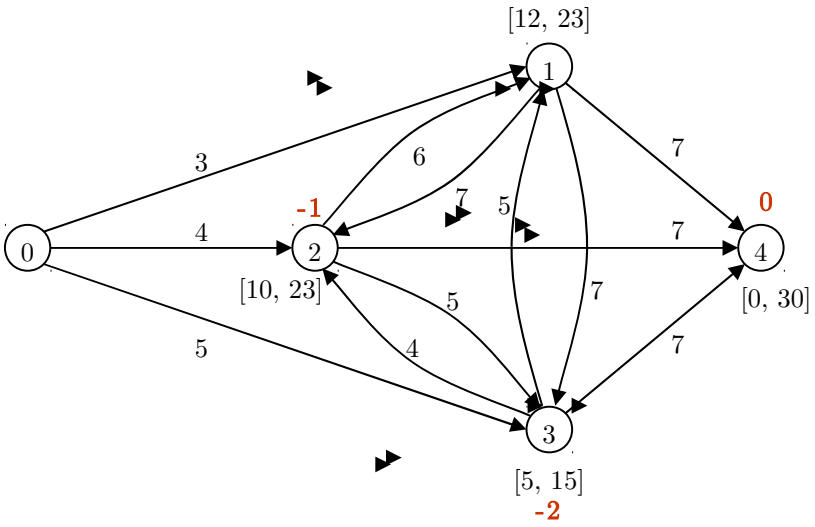
$$l_a \preceq l_b \Leftrightarrow \begin{array}{rcl} v_a & = & v_b \\ \tilde{t}_a & \leq & \tilde{t}_b \\ \tilde{c}_a & \leq & \tilde{c}_b \\ U_a & \subseteq & U_b \end{array}$$

$$l_a \preceq l_b \Leftrightarrow \begin{array}{rcl} v_a & = & v_b \\ \tilde{t}_a & \leq & \tilde{t}_b \\ \tilde{c}_a & \leq & \tilde{c}_b + \sum_{m \in U_a \setminus U_b} \bar{c}_m \end{array}$$

l_5=(2,l_3,10,-2,{3})
l_2=(2,l_0,10,0, ∅)
$\Rightarrow l_a \; | l_b$



The label setting algorithm



Extend	New label: $(v, p, \tilde{t}, \tilde{c}, U)$	Dominance	$ULPQ$
	$l_0 = (0, l_\varnothing, 0, 0, \varnothing)$		$[l_0]$
l_0:	$l_1 = (1, l_0, \mathbf{12}, 0, \{\mathbf{3}\})$		$[l_1]$
	$l_2 = (2, l_0, \mathbf{10}, 0, \varnothing)$		$[l_2, l_1]$
	$l_3 = (3, l_0, 5, 0, \varnothing)$		$[l_3, l_2, l_1]$
l_3:	$l_4 = (1, l_3, \mathbf{12}, -2, \{3\})$	$l_4 \preceq l_1$	$[l_2, l_4, \sout{l_1}]$
	$l_5 = (2, l_3, \mathbf{10}, -2, \{3\})$	$l_5 \preceq l_2$	$[l_5, \sout{l_2}, l_4]$
	$l_6^* = (4, l_3, 12, -2, \{3\})$		$[l_5, l_4]$
l_5:	$l_7 = (1, l_5, 16, -3, \{2, 3\})$		$[l_4, l_7]$
	$l_8^* = (4, l_5, 17, -3, \{2, 3\})$		$[l_4, l_7]$
l_4:	$l_{12} = (2, l_4, 19, -\frac{5}{2}, \{1, 3\})$		$[l_7, l_{12}]$
	$l_{13}^* = (4, l_4, 19, -\frac{5}{2}, \{1, 3\})$		$[l_7, l_{12}]$
l_7:	$l_{18}^* = (4, l_7, 23, -\frac{7}{2}, \{1, 2, 3\})$		$[l_{12}]$
l_{12}:	$l_{20}^* = (4, l_{12}, 26, -\frac{7}{2}, \{1, 2, 3\})$		$[]$

The label setting algorithm

Extend	New label: $(v, p, \tilde{t}, \tilde{c}, W)$	U	$ULPQ$
	$l_0 = (0, l_\varnothing, 0, 0, \varnothing)$	$\varnothing$	$[l_0]$
l_0:	$l_1 = (1, l_0, \mathbf{12}, 0, \varnothing)$	$\{3\}$	$[l_1]$
	$l_2 = (2, l_0, \mathbf{10}, 0, \varnothing)$	$\varnothing$	$[l_2, l_1]$
	$l_3 = (3, l_0, 5, 0, \varnothing)$	$\varnothing$	$[l_3, l_2, l_1]$
l_3:	$l_4 = (1, l_3, \mathbf{12}, -2, \{3\})$	$\{3\}$	$[l_2, l_4, l_1]$
	$l_5 = (2, l_3, \mathbf{10}, -2, \{3\})$	$\{3\}$	$[l_5, l_2, l_4, l_1]$
	$l_6^* = (4, l_3, 12, -2, \{3\})$	$\{3\}$	$[l_5, l_2, l_4, l_1]$
l_5:	$l_7 = (1, l_5, 16, -3, \{2, 3\})$	$\{2, 3\}$	$[l_2, l_4, l_1, l_7]$
	$l_8^* = (4, l_5, 17, -3, \{2, 3\})$	$\{2, 3\}$	$[l_2, l_4, l_1, l_7]$
l_2:	$l_9 = (1, l_2, 16, -1, \{2\})$	$\{2, 3\}$	$[l_4, l_1, l_7, l_9]$
	$l_{10} = (3, l_2, 15, -1, \{2\})$	$\{2\}$	$[l_4, l_1, l_{10}, l_7, l_9]$
	$l_{11}^* = (4, l_2, 17, -1, \{2\})$	$\{2, 3\}$	$[l_4, l_1, l_{10}, l_7, l_9]$
l_4:	$l_{12} = (2, l_4, 19, -\frac{5}{2}, \{1, 3\})$	$\{1, 3\}$	$[l_1, l_{10}, l_7, l_9, l_{12}]$
	$l_{13}^* = (4, l_4, 19, -\frac{5}{2}, \{1, 3\})$	$\{1, 3\}$	$[l_1, l_{10}, l_7, l_9, l_{12}]$
l_1:	$l_{14} = (2, l_1, 19, -\frac{1}{2}, \{1\})$	$\{1, 3\}$	$[l_{10}, l_7, l_9, l_{12}, l_{14}]$
	$l_{15}^* = (4, l_1, 19, -\frac{1}{2}, \{1\})$	$\{1, 3\}$	$[l_{10}, l_7, l_9, l_{12}, l_{14}]$
l_{10}:	$l_{16} = (1, l_{10}, 20, -3, \{2, 3\})$	$\{2, 3\}$	$[l_7, l_9, l_{12}, l_{14}, l_{16}]$
	$l_{17}^* = (4, l_{10}, 22, -3, \{2, 3\})$	$\{2, 3\}$	$[l_7, l_9, l_{12}, l_{14}, l_{16}]$
l_7:	$l_{18}^* = (4, l_7, 23, -\frac{7}{2}, \{1, 2, 3\})$	$\{1, 2, 3\}$	$[l_9, l_{12}, l_{14}, l_{16}]$
l_9:	$l_{19}^* = (4, l_9, 23, -\frac{3}{2}, \{1, 2\})$	$\{1, 2, 3\}$	$[l_{12}, l_{14}, l_{16}]$
l_{12}:	$l_{20}^* = (4, l_{12}, 26, -\frac{7}{2}, \{1, 2, 3\})$	$\{1, 2, 3\}$	$[l_{14}, l_{16}]$
l_{14}:	$l_{21}^* = (4, l_{14}, 26, -\frac{3}{2}, \{1, 2\})$	$\{1, 2, 3\}$	$[l_{16}]$
l_{16}:	$l_{22}^* = (4, l_{16}, 27, -\frac{7}{2}, \{1, 2, 3\})$	$\{1, 2, 3\}$	$[]$

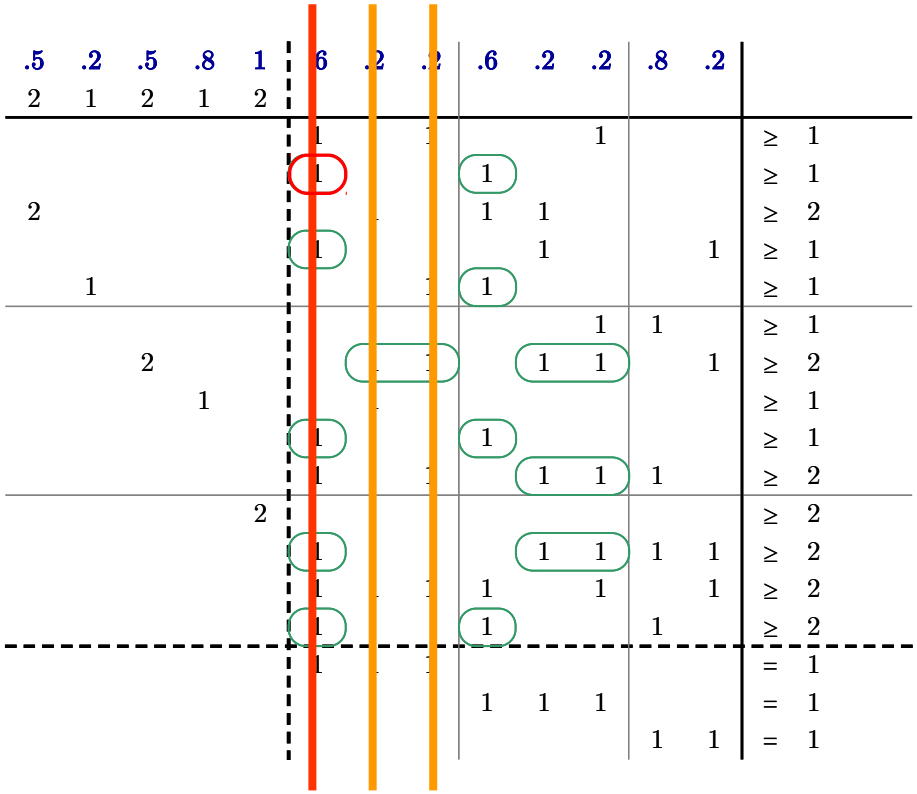
Extend	New label: $(v, p, \tilde{t}, \tilde{c}, U)$	Dominance	$ULPQ$
	$l_0 = (0, l_\varnothing, 0, 0, \varnothing)$		$[l_0]$
l_0:	$l_1 = (1, l_0, \mathbf{12}, 0, \{\mathbf{3}\})$		$[l_1]$
	$l_2 = (2, l_0, \mathbf{10}, 0, \varnothing)$		$[l_2, l_1]$
	$l_3 = (3, l_0, 5, 0, \varnothing)$		$[l_3, l_2, l_1]$
l_3:	$l_4 = (1, l_3, \mathbf{12}, -2, \{3\})$	$l_4 \preceq l_1$	$[l_2, l_4, \sout{l_1}]$
	$l_5 = (2, l_3, \mathbf{10}, -2, \{3\})$	$l_5 \preceq l_2$	$[l_5, \sout{l_2}, l_4]$
	$l_6^* = (4, l_3, 12, -2, \{3\})$		$[l_5, l_4]$
l_5:	$l_7 = (1, l_5, 16, -3, \{2, 3\})$		$[l_4, l_7]$
	$l_8^* = (4, l_5, 17, -3, \{2, 3\})$		$[l_4, l_7]$
l_4:	$l_{12} = (2, l_4, 19, -\frac{5}{2}, \{1, 3\})$		$[l_7, l_{12}]$
	$l_{13}^* = (4, l_4, 19, -\frac{5}{2}, \{1, 3\})$		$[l_7, l_{12}]$
l_7:	$l_{18}^* = (4, l_7, 23, -\frac{7}{2}, \{1, 2, 3\})$		$[l_{12}]$
l_{12}:	$l_{20}^* = (4, l_{12}, 26, -\frac{7}{2}, \{1, 2, 3\})$		$[]$

Branch & Price

- Necessary considerations:
  - How do we model and solve the master problem?
  - How do we model and solve the subproblem?
  - How do we ensure integrality in the master problem?

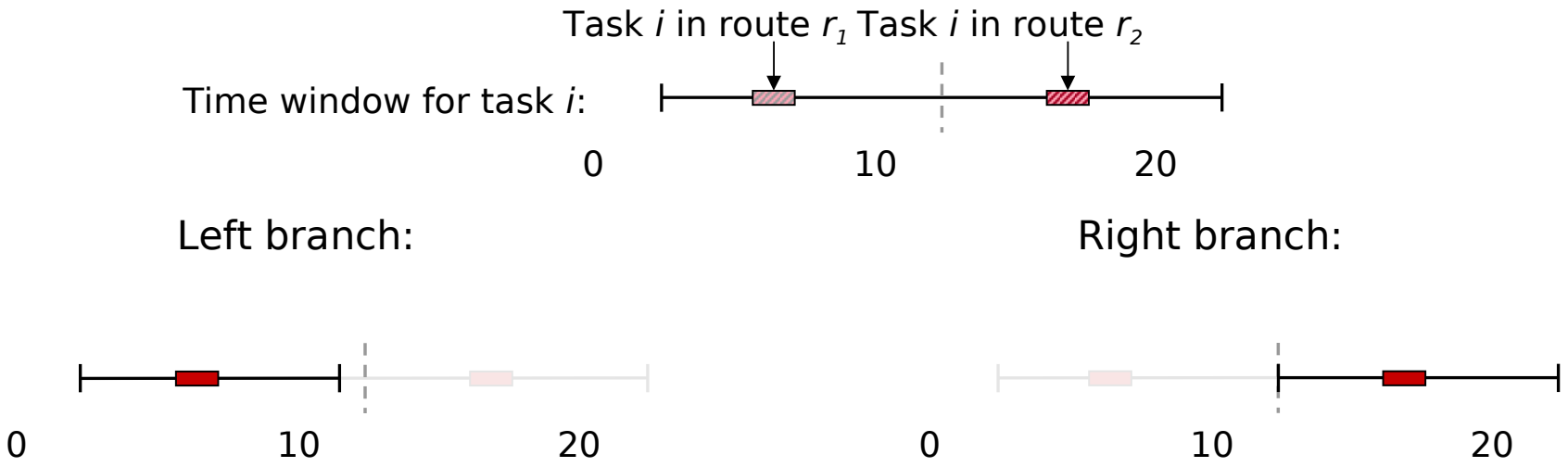
Branching

- Branching on task allocation (sum of fractions)
  - "How much of task *i* is assigned to team *k*" ?
  - Fractional variables ⇒ At least one S_{ik} is fractional
  - Branch on S_{ik}: $S_{ik} = \sum_{p \in \mathcal{P}'_k} a^p_{ik} \lambda^p_k$
    - 0-branch: Team *k* cannot do task *i*
    - 1-branch: Team *k* must do task *i*
  - Remove infeasible columns
  - Force / remove task in solution of pricing problem

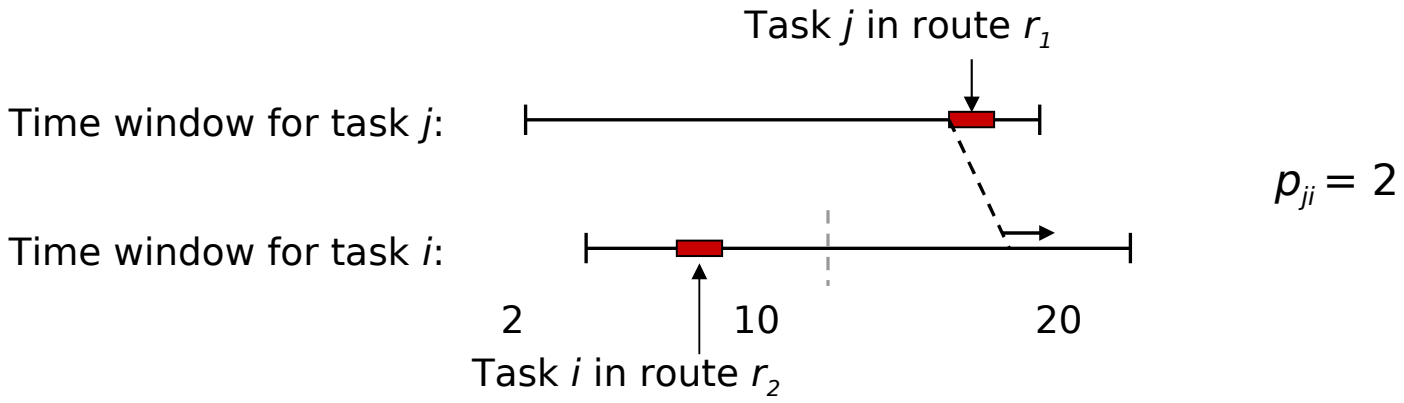


Branching on Time Windows

- Will remove most fractional values.
- Will enforce all temporal dependencies.
- Proposed as branching strategy to solely remove fractional values in traditional VRPTW [Gélinas et al. 1995].

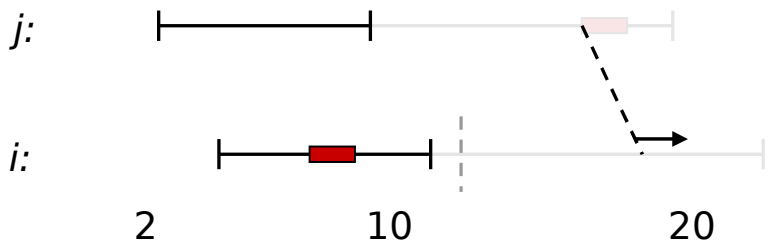


Branching on Time Windows

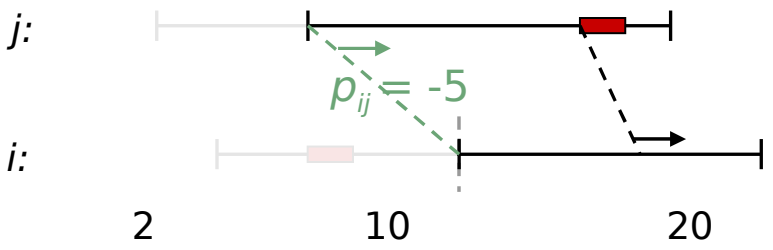


$p_{ji} = 2$

Left branch:



Right branch:

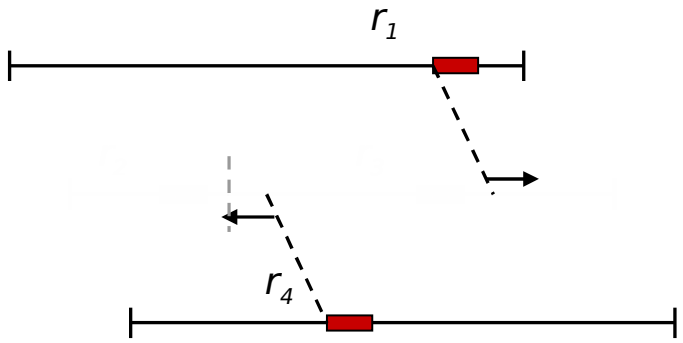


Branching on Time Windows

Task j in route r_1:

Task i in route r_2 and route r_3:

Task k in route r_4:



$p_{ji} = 2$

$p_{ik} = 2$

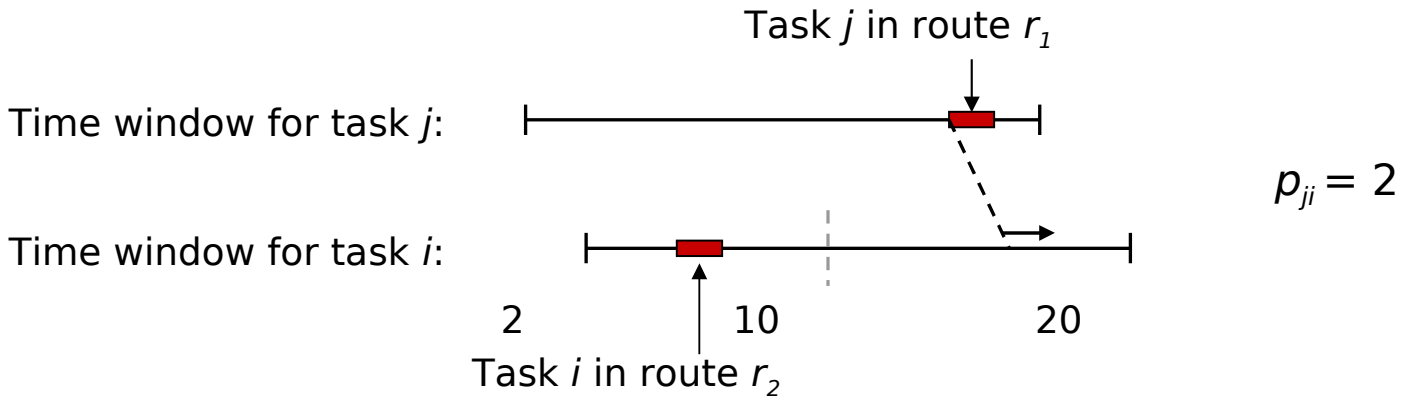
Infeasible routes:

	Left branch:	Right branch:
Branching candidate 1:	r_3, r_1	r_2
Branching candidate 2:	r_3, r_1	r_2, r_4
Branching candidate 3:	r_1	r_2, r_3, r_4

Branching on Time Windows

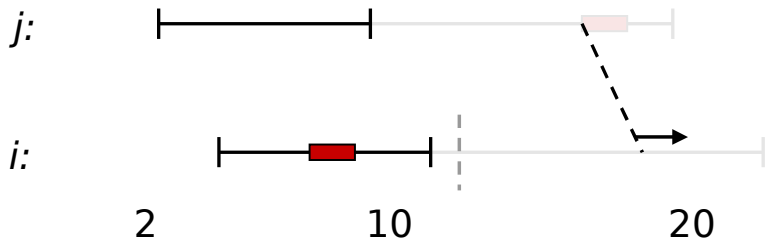
- Choosing the best branching candidate:
  - Create a balanced branching tree.
  - Choose a candidate that has the largest impact on both branches.
  - In this case, choose the right-most point in the time window.

Branching on Time Windows

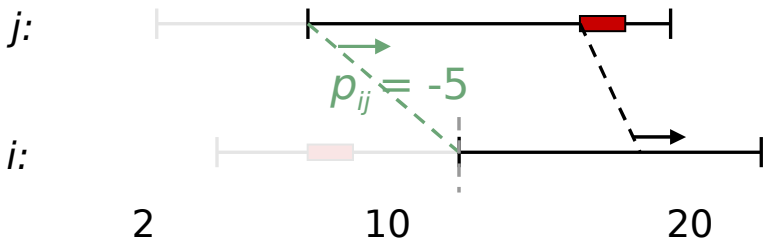


$p_{ji} = 2$

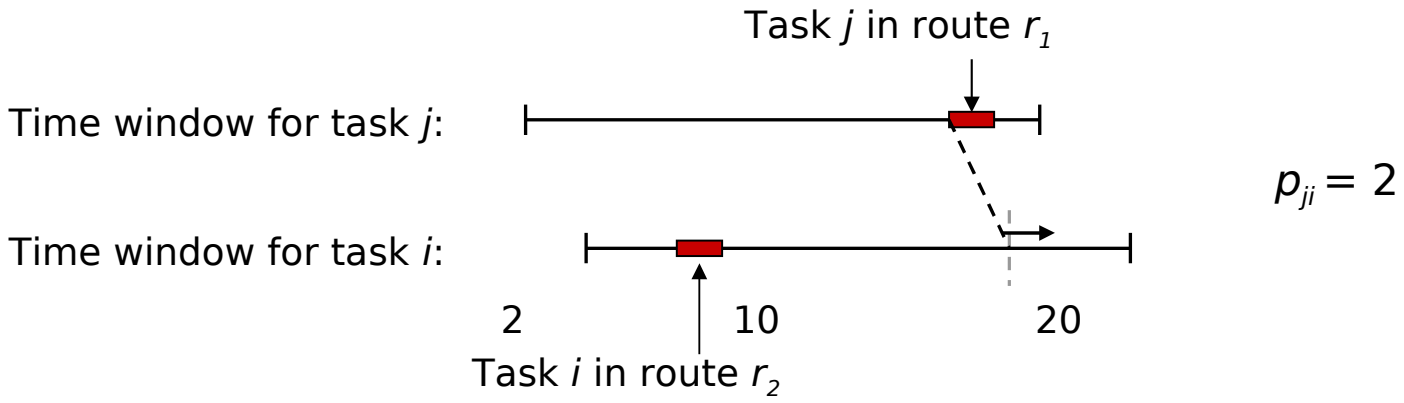
Left branch:



Right branch:

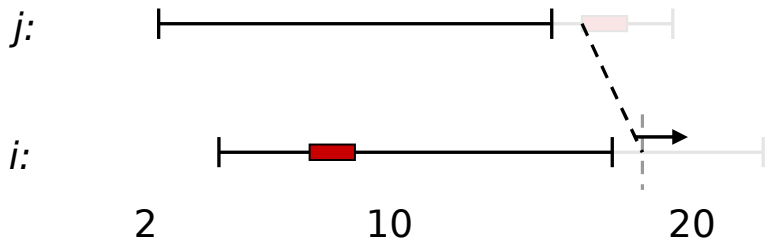


Branching on Time Windows

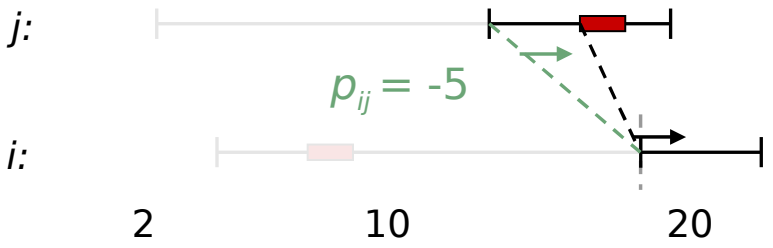


$p_{ji} = 2$

Left branch:



Right branch:

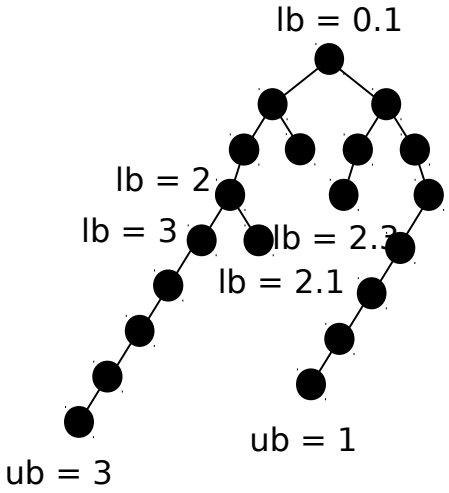


Branch & Price

- Necessary considerations:
  - How do we model and solve the master problem?
  - How do we model and solve the subproblem?
  - How do we ensure integrality in the master problem?
- Other considerations:
  - Master problem
    - Solve it to optimality every time?
    - Use dual stabilization?
  - Subproblem
    - Use heuristics?
    - In what order should the individual subproblems be solved?
  - Branching
    - How do we search the branch-and-bound tree?

Granularity

- The smallest possible difference in solution value between two feasible solutions.
  - All feasible solutions have integer solution values ⇒ granularity = 1
- Utilization in the branching tree:
  - The current node can be removed if the difference between the incumbent and the current lower bound is less than the granularity.



Granularity

- Further utilization of granularity:

  $+ \kappa\bar{c}^* \leq z^*_{MP} \leq \bar{z}$ on to the current Restricted Master Problem.

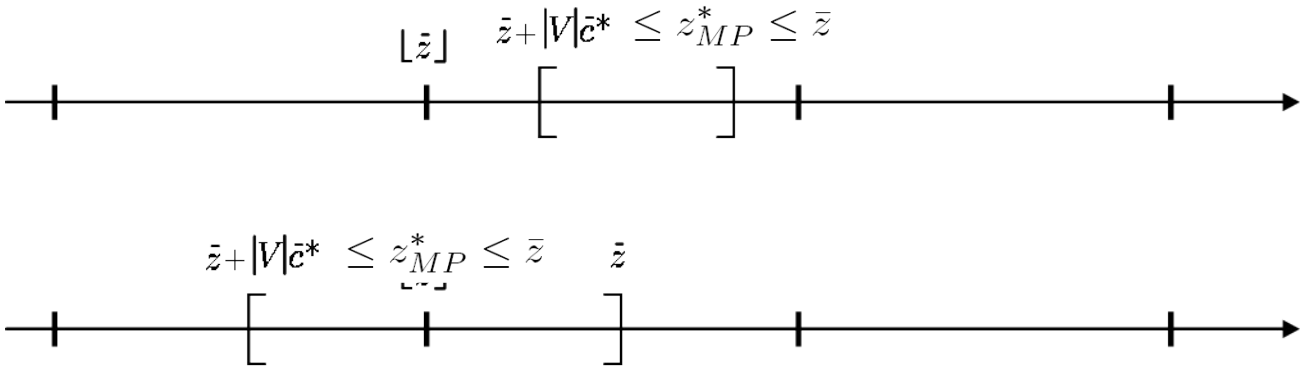
  $\kappa\bar{c}^* \leq \quad \leq \bar{z}$: optimal solution to the Master Problem (this value is only known when no more columns can be added in the current node).

  $\bar{z} + \kappa \quad \leq z^*_{MP} \leq \bar{z}$ nal solution to the current subproblem.

  $\bar{z} + \quad \bar{c}^* \leq z^*_{MP} \leq \bar{z}$ f employees

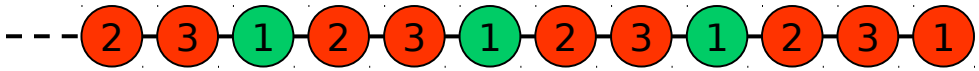
$$\bar{z} + \kappa\bar{c}^* \leq z^*_{MP} \leq \bar{z}$$

- If next integer cannot be reached: branch/fathom immediately

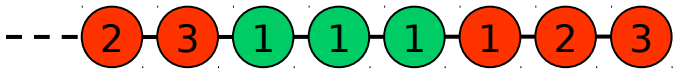


Multiple subproblems

- Usual approach:
  - Solve subproblems in a Round-robin fashion:



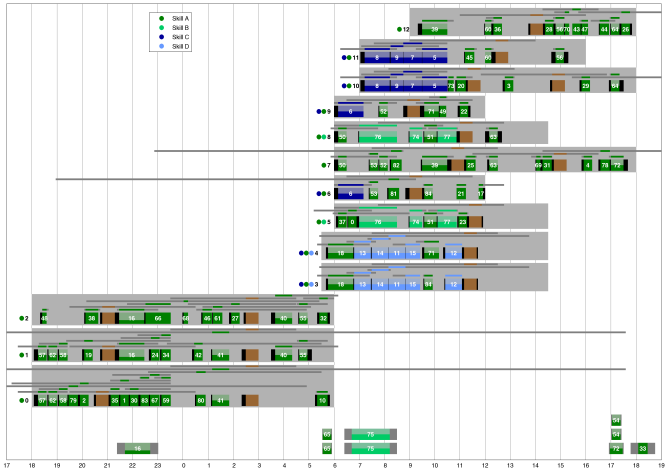
  - This is inefficient if a few subproblems produce better columns than the rest.
- An alternative approach:
  - Prioritize subproblems according to an expected performance.



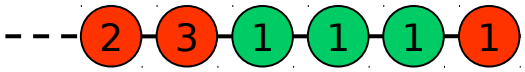
  - Requires a performance measure. The most recent solution value may be used.
  - All subproblems must be solved with the most recent dual values in order to declare the solution of the master problem optimal.

Multiple subproblems

- The solution of some subproblems may be avoided by considering only “relevant” dual variables.
- Some problems are highly segregated.



- If none of the values of the “relevant” variables have changed, the optimal solution to that subproblem has already been found.



Summary

- What you should be able to remember from this lecture:
  - Task Scheduling
    - The practical problem
    - Synchronization
    - Temporal Dependencies in general
      - Generalized Precedence Constraints
  - Solution method
    - Branch & Price
    - Various master problem formulations
    - Solving the subproblem by Label Setting
    - Branching on Time Windows
    - Granularity
    - Prioritizing subproblems

The assignment

- Central Security Control (CSC) at Copenhagen Airport.
- The largest “single” task in the airport.
- Find optimal number of shifts of each type.
  - Given:
    - A set of possible shifts.
    - An estimated demand for one day.
- Questions:
  - What is the optimal cover, if all demand is covered?
  - What is the optimal cover, if undercoverage can be accepted at a certain price?
  - How can breaks be included?
  - How can robustness be included?

The assignment

Shift Name	Time
A0	04:00-14:00
A1	05:00-14:00
C	06:00-18:00
D	10:00-20:00
F0	13:00-21:00
F1	14:00-23:00
H3	20:30-06:30
H4	18:00-04:00
K2	08:00-16:00

The assignment