Lagrangian Relaxation in CP

Willem-Jan van Hoeve

CPAIOR 2016 Master Class

Overview

1. Motivation for using Lagrangian Relaxations in CP
2. Lagrangian-based domain filtering
   – Example: Traveling Salesman Problem
3. Relaxed Decision Diagrams
   – Example: Disjunctive Scheduling
4. Lagrangian Propagation
   – Improve communication between constraints

Acknowledgements: Presentations from Andre Cire and Hadrien Cambazard

Lagrangian Relaxation

Lagrangian Relaxation

Move subset (or all) of constraints into the objective with 'penalty' multipliers μ:

$$
\begin{aligned}
\min \quad & c^\top x \\
\text{s.t.} \quad & A_1 x = b_1 \\
& A_2 x = b_2 \\
& x \geq 0
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
L(\mu) = \quad \min \quad & c^\top x + \mu^\top (b_2 - A_2 x) \\
\text{s.t.} \quad & A_1 x = b_1 \\
& x \geq 0
\end{aligned}
$$

Lagrangian Relaxation

Move subset (or all) of constraints into the objective with 'penalty' multipliers μ:

$$\begin{aligned} \min \quad & c^\mathsf{T} x \\ \text{s.t.} \quad & A_1 x = b_1 \\ & A_2 x = b_2 \\ & x \geq 0 \end{aligned} \qquad \longrightarrow \qquad \begin{aligned} L(\mu) = \quad \min \quad & c^\mathsf{T} x + \mu^\mathsf{T}(b_2 - A_2 x) \\ \text{s.t.} \quad & A_1 x = b_1 \\ & x \geq 0 \end{aligned}$$

Weak duality: for any choice of μ, Lagrangean $L(\mu)$ provides a lower bound on the original LP

Lagrangian Relaxation

Move subset (or all) of constraints into the objective with ‘penalty’ multipliers μ:

$$
\begin{aligned}
\min \quad & c^\top x \\
\text{s.t.} \quad & A_1 x = b_1 \\
& A_2 x = b_2 \\
& x \geq 0
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
L(\mu) = \quad \min \quad & c^\top x + \mu^\top (b_2 - A_2 x) \\
\text{s.t.} \quad & A_1 x = b_1 \\
& x \geq 0
\end{aligned}
$$

Weak duality: for any choice of μ, Lagrangean $L(\mu)$ provides a lower bound on the original LP

Goal: find optimal μ (providing the best bound) via $\max_{\mu \geq 0} \; L(\mu)$

Lagrangian Relaxations are Awesome

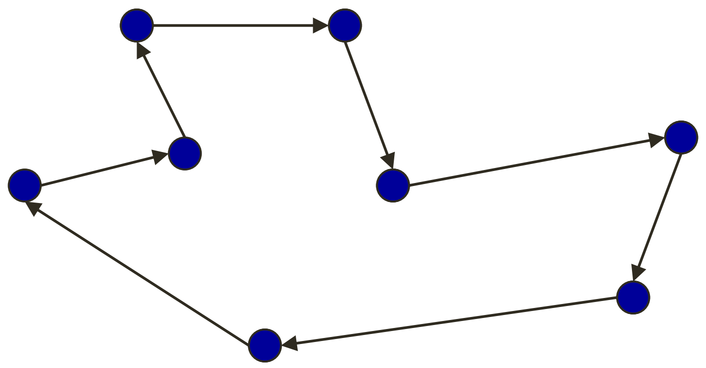
- Can be applied to nonlinear programming problems (NLPs), LPs, and in the context of integer programming
- Can provide better bounds than LP relaxation:

$$z_{LP} \leq z_{Lagr} \leq z_{IP}$$

- Provides domain filtering analogous to that based on LP duality
- Can be efficiently and/or heuristically solved
- Lagrangian relaxation can dualize 'difficult' constraints
  - Can exploit the problem structure, e.g., the Lagrangian relaxation may decouple, or $L(\mu)$ may be very fast to solve combinatorially

Example Application: TSP

- Visit all vertices exactly once, with minimum total distance



Graph G = (V,E) with vertex set V and edge set E

$|V| = n$

$w(i,j)$: distance between i and j

CP model for the TSP (version 1)

- Permutation model
  - variable x_i represents the i-th city to be visited
  - introduce copy of vertex 1: vertex n+1

$$
\begin{array}{ll}
\min & z \\
\text{s.t.} & z = \sum_i w(x_i, x_{i+1}) \\
 & \textit{alldifferent}(x_1, \ldots, x_n) \\
 & x_{n+1} = x_1
\end{array}
$$

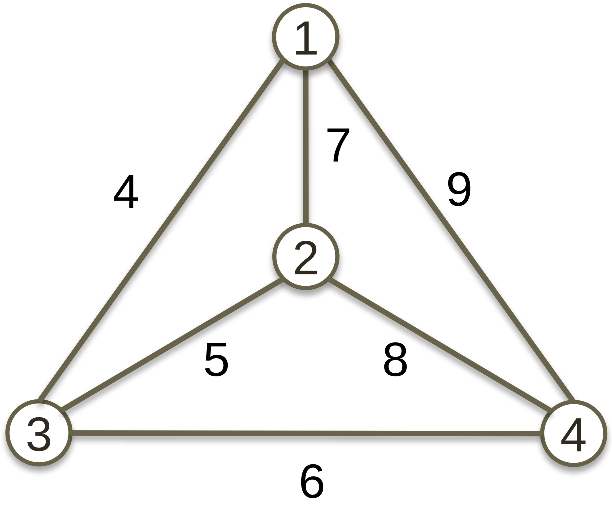
CP Solving

min z

s.t. $z = \sum_{i=1..4} w(x_i, x_{i+1})$ (1)

$alldifferent(x_1, \ldots, x_4)$ (2)

$x_5 = x_1$ (3)



Variable **domains**:

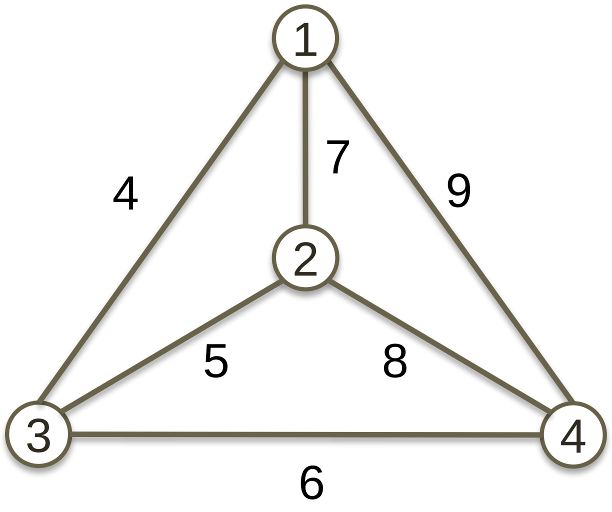
$D(z) = \{ 0..\text{inf} \}$

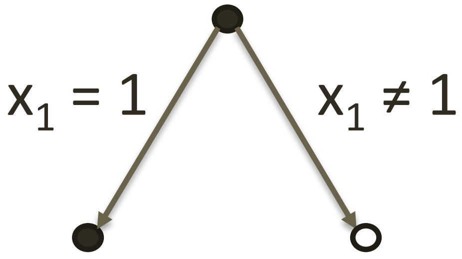
$D(x_i) = \{1,2,3,4\}$ for $i=1,\ldots,5$

Propagate (1) to update $D(z) = \{16,..,36\}$

CP Solving – cont'd

min	z	
s.t.	$z = \sum_{i=1..4} w(x_i, x_{i+1})$	(1)
	$\mathit{alldifferent}(x_1, \ldots, x_4)$	(2)
	$x_5 = x_1$	(3)





Propagate (2) :

$D(x_2) = \{2,3,4\}$

$D(x_3) = \{2,3,4\}$

$D(x_4) = \{2,3,4\}$

Propagate (3) :

$D(x_5) = \{1\}$

Propagate (1) : no updates

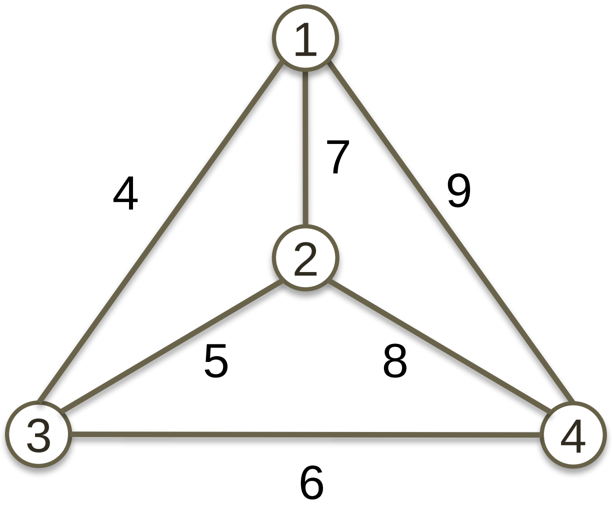
CP Solving – cont'd

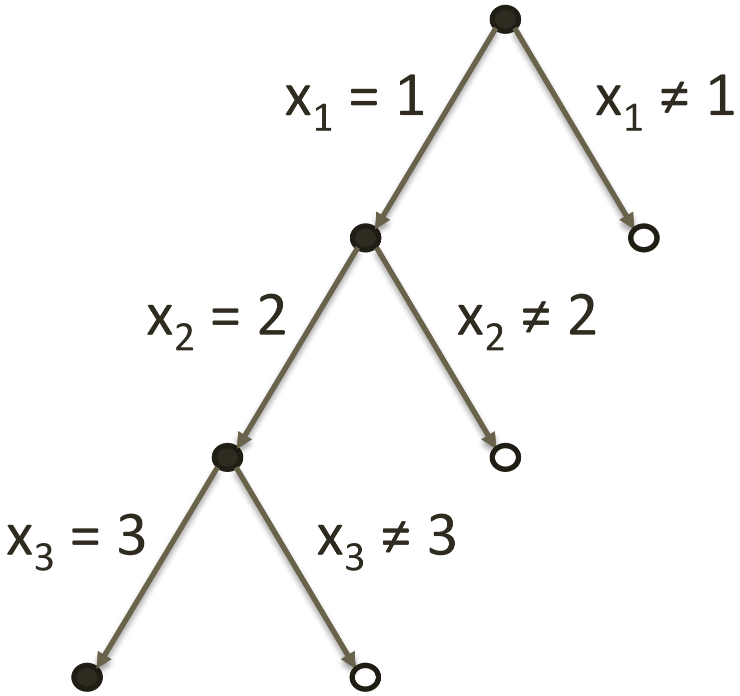
min z

s.t. $z = \sum_{i=1..4} w(x_i, x_{i+1})$ (1)

$alldifferent(x_1, \ldots, x_4)$ (2)

$x_5 = x_1$ (3)





$D(x_4) = \{4\}$

$D(z) = \{27\}$

Drawback of first CP Model

- Objective is handled separately from constraints
- Interaction via domain propagation only
- Weak bounds

Other CP models for the TSP

- Successor model
  - variable $next_i$ represents the immediate successor of city i

$$\begin{aligned} \text{min} \quad & z = \textstyle\sum_i w(i, next_i) \\ \text{s.t.} \quad & \textit{alldifferent}(next_1, \ldots, next_n) \\ & \textit{path}(next_1, \ldots, next_n) \end{aligned}$$

Other CP models for the TSP

- Successor model
  - variable $next_i$ represents the immediate successor of city i

$$\begin{array}{ll} \text{min} & z = \sum_i w(i, next_i) \\ \text{s.t.} & \textit{alldifferent}(next_1, \ldots, next_n) \\ & \textit{path}(next_1, \ldots, next_n) \end{array}\Bigg\}$$

objective and constraints still decoupled

Other CP models for the TSP

- Successor model
  - variable $next_i$ represents the immediate successor of city i

$$\begin{array}{ll} \text{min} & z = \sum_i w(i, next_i) \\ \text{s.t.} & \textit{alldifferent}(next_1, \ldots, next_n) \\ & \textit{path}(next_1, \ldots, next_n) \end{array}$$

objective and constraints still decoupled

- Integrated model using 'optimization' constraint

[Focacci et al., 1999, 2002]

$$\begin{array}{lll} \text{min} & z & \\ \text{s.t.} & \textit{alldifferent}(next_1, \ldots, next_n) & \text{[redundant]} \\ & \textit{WeightedPath}(next, w, z) & \end{array}$$

Optimization Constraint

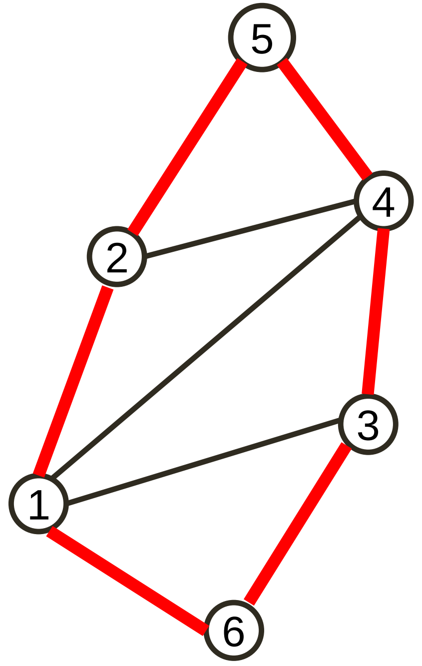
- *WeightedPath*(next, w, z)
  - Ensures that variables $next_i$ represent a Hamiltonian path
    such that the total weight of the path equals variable z

- Benefits:
  - Stronger bounds from constraint structure, e.g., based on LP relaxation
  - 'Cost-based' domain filtering:
    if $next_i$=v leads to path of weight > max(D(z)), remove v from D($next_i$)

Relaxations for TSP

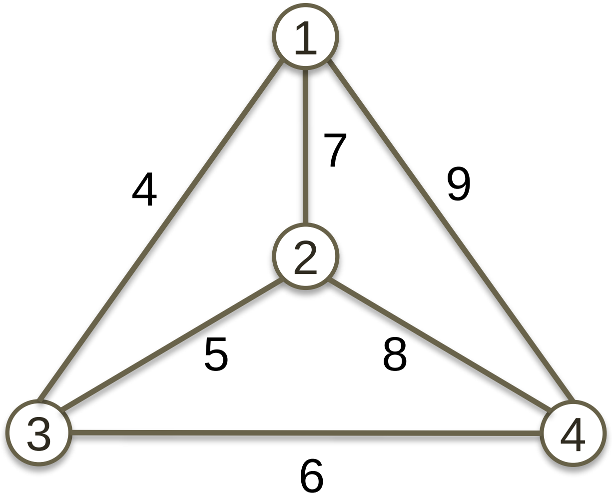
- Observe that the TSP is a combination of two constraints
  - The degree of each node is 2
  - The solution is connected (no subtours)

- Relaxations:
  - relax connectedness: Assignment Problem
  - relax degree constraints: 1-Tree Relaxation



1-Tree Relaxation

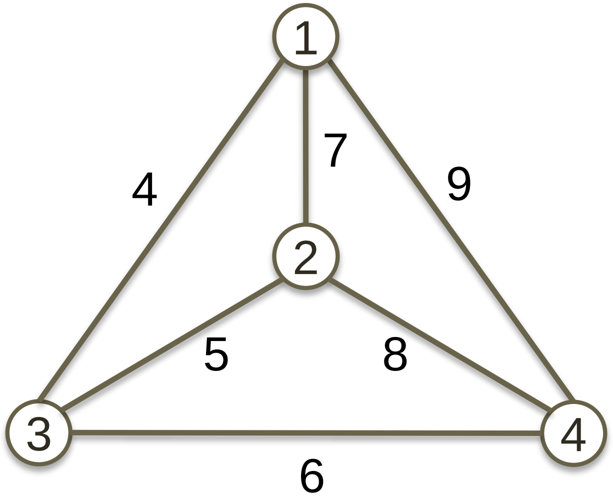
- Relax the degree constraints [Held & Karp, 1970, 1971]
  - E.g., minimum spanning tree (has n-1 edges)



P.S. An MST can be found in $O(m\ \alpha(m,n))$ time

1-Tree Relaxation

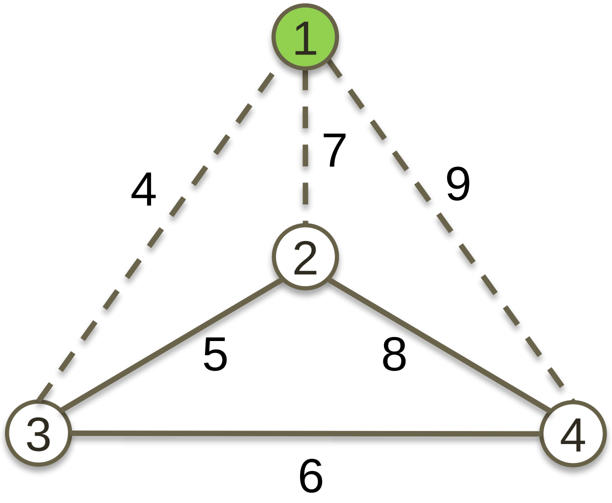
- Relax the degree constraints [Held & Karp, 1970, 1971]
  - E.g., minimum spanning tree (has n-1 edges)
  - 1-Tree extends this with one more edge
    - Choose any node v (which is called the 1-node)
    - Build a minimum spanning tree T on G = (V\{v}, E)
    - Add the smallest two edges linking v to T



P.S. An MST can be found in $O(m\,\alpha(m,n))$ time

1-Tree Relaxation

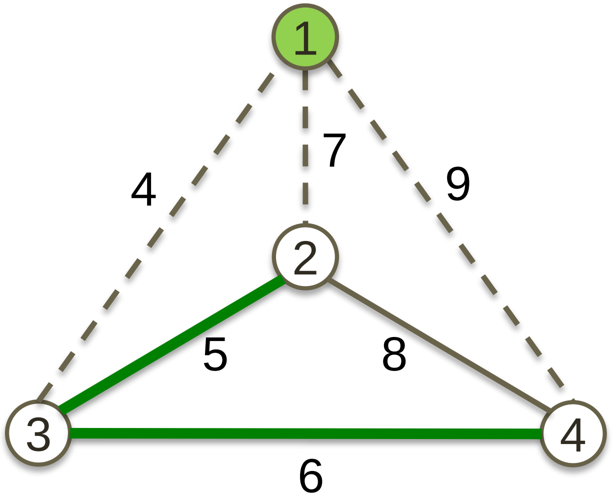
- Relax the degree constraints [Held & Karp, 1970, 1971]
  - E.g., minimum spanning tree (has n-1 edges)
  - 1-Tree extends this with one more edge
    - Choose any node v (which is called the 1-node)
    - Build a minimum spanning tree T on G = (V\{v}, E)
    - Add the smallest two edges linking v to T



P.S. An MST can be found in O(m α(m,n)) time

1-Tree Relaxation

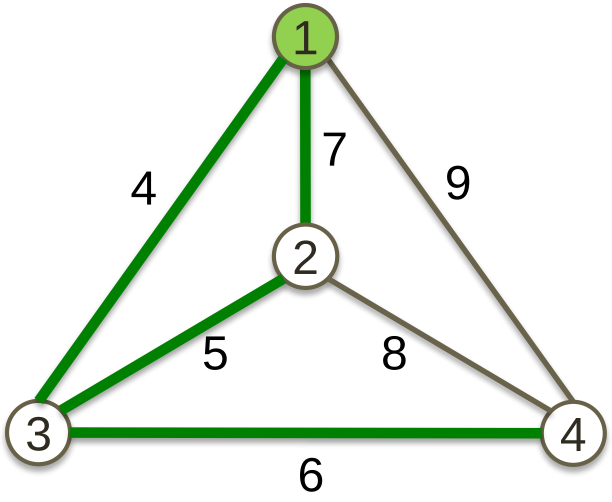
- Relax the degree constraints [Held & Karp, 1970, 1971]
  - E.g., minimum spanning tree (has n-1 edges)
  - 1-Tree extends this with one more edge
    - Choose any node v (which is called the 1-node)
    - Build a minimum spanning tree T on G = (V\{v}, E)
    - Add the smallest two edges linking v to T



P.S. An MST can be found in $O(m\ \alpha(m,n))$ time

1-Tree Relaxation

- Relax the degree constraints [Held & Karp, 1970, 1971]
  - E.g., minimum spanning tree (has n-1 edges)
  - 1-Tree extends this with one more edge
    - Choose any node v (which is called the 1-node)
    - Build a minimum spanning tree T on G = (V\{v}, E)
    - Add the smallest two edges linking v to T



P.S. An MST can be found in $O(m\ \alpha(m,n))$ time

Improved 1-Tree: Lagrangian relaxation

Let binary variable x_e represent whether edge e is used

$$\min \sum_{e \in E} w(e) x_e$$

$$\text{s.t.} \sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V$$

$$\sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V, |S| \geq 3$$

$$x_e \in \{0, 1\} \qquad \forall e \in E$$

Improved 1-Tree: Lagrangian relaxation

Let binary variable x_e represent whether edge e is used

$$\begin{aligned}
\min \sum_{e \in E} & w(e) x_e \\
\text{s.t.} \quad & \boxed{\sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V} \\
& \sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V, |S| \geq 3 \\
& x_e \in \{0, 1\} \qquad \forall e \in E
\end{aligned}$$

Improved 1-Tree: Lagrangian relaxation

Let binary variable x_e represent whether edge e is used

$$\min \sum_{e \in E} w(e) x_e$$

$$\text{s.t.} \quad \boxed{\sum_{e \in \delta(i)} x_e = 2 \qquad \forall i \in V}$$

$$\sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V, |S| \geq 3$$

$$x_e \in \{0,1\} \qquad \forall e \in E$$

Lagrangian multipliers π_i (penalties for node degree violation)

Held-Karp Bound

$$\min \sum_{e \in E} w(e) x_e + \sum_{i \in V \setminus \{1\}} \pi_i (2 - \sum_{e \in \delta(i)} x_e)$$

$$\text{s.t.} \quad \sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V \setminus \{1\}, |S| \geq 3$$

$$x_e \in \{0, 1\}$$

Held-Karp Bound

$$\min \sum_{e \in E} w(e) x_e + \sum_{i \in V \setminus \{1\}} \pi_i (2 - \sum_{e \in \delta(i)} x_e)$$

$$\text{s.t.} \quad \sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V \setminus \{1\}, |S| \geq 3$$

$$\sum_{e \in \delta(1)} x_e = 2$$

$$\sum_{e \in E} x_e = |V|$$

$$x_e \in \{0, 1\}$$

Held-Karp Bound

$$\min \sum_{e \in E} w(e)x_e + \sum_{i \in V \setminus \{1\}} \pi_i \left(2 - \sum_{e \in \delta(i)} x_e\right) = \sum_{(i,j) \in E} (w(i,j) - \pi_i - \pi_j) x_{(i,j)} + 2 \sum_{i \in V} \pi_i$$

$$\text{s.t.} \quad \sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V \setminus \{1\}, |S| \geq 3$$

$$\sum_{e \in \delta(1)} x_e = 2$$

$$\sum_{e \in E} x_e = |V|$$

$$x_e \in \{0, 1\}$$

Held-Karp Bound

$$\min \sum_{e \in E} w(e) x_e + \sum_{i \in V \setminus \{1\}} \pi_i \Big(2 - \sum_{e \in \delta(i)} x_e\Big) = \sum_{(i,j) \in E} (w(i,j) - \pi_i - \pi_j) x_{(i,j)} + 2 \sum_{i \in V} \pi_i$$

$$\text{s.t.} \quad \sum_{i,j \in S, i<j} x_{(i,j)} \leq |S| - 1 \qquad \forall S \subset V \setminus \{1\}, |S| \geq 3$$

$$\sum_{e \in \delta(1)} x_e = 2$$

$$\sum_{e \in E} x_e = |V|$$

$$x_e \in \{0, 1\}$$

How to find the best penalties π ?

- In this case, we can exploit a combinatorial interpretation
- No need to solve an LP

Held-Karp Iteration

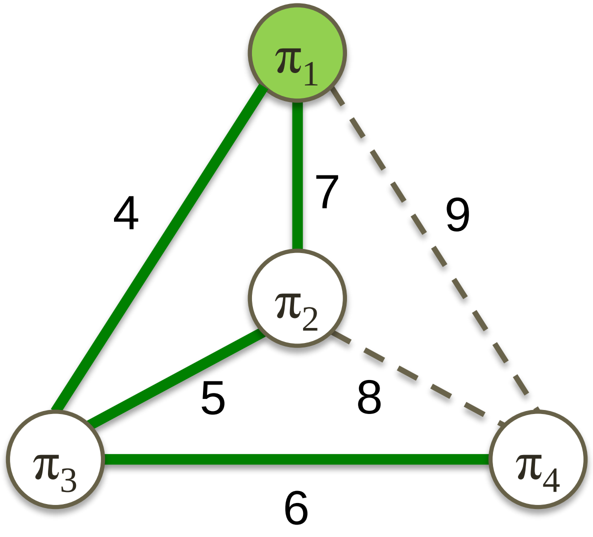
$$\min \sum_{(i,j)\in E} (w(i,j) - \pi_i - \pi_j)x_{(i,j)} + 2\sum_{i\in V} \pi_i$$

- Find minimum 1-tree T w.r.t. $w'(i,j) = w(i,j) - \pi_i - \pi_j$
- Lower bound: cost(T) + 2 $\sum_i \pi_i$
- If T is not a tour, update multipliers as

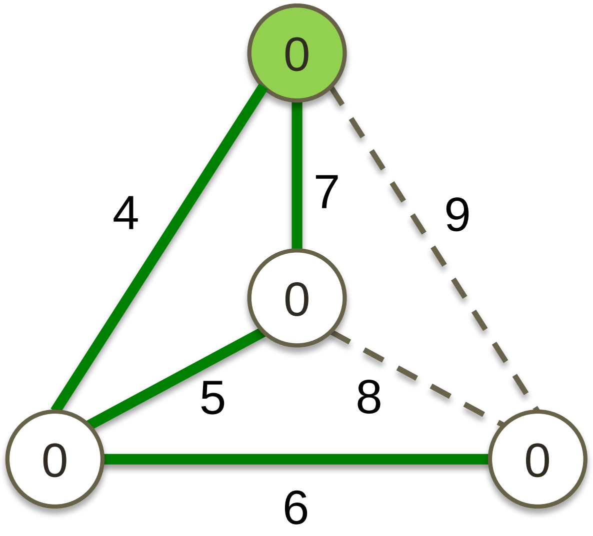
π_i += (2-degree(i))*β

and repeat

(step size β different per iteration)



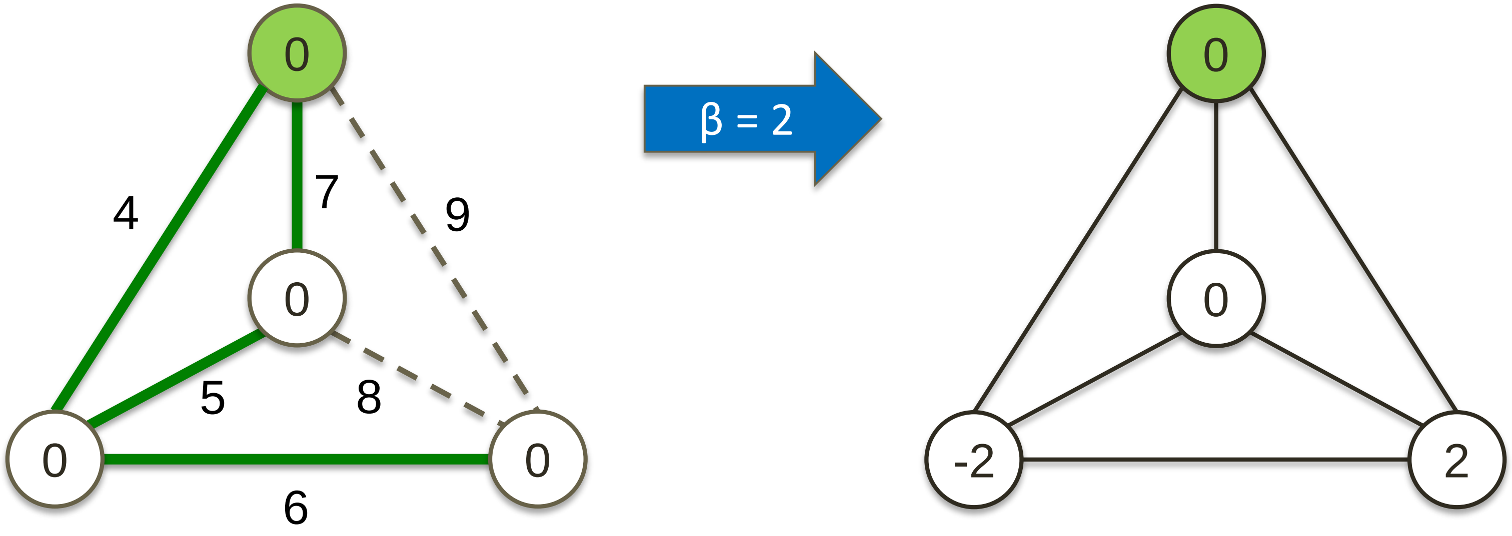
Example



β = 2

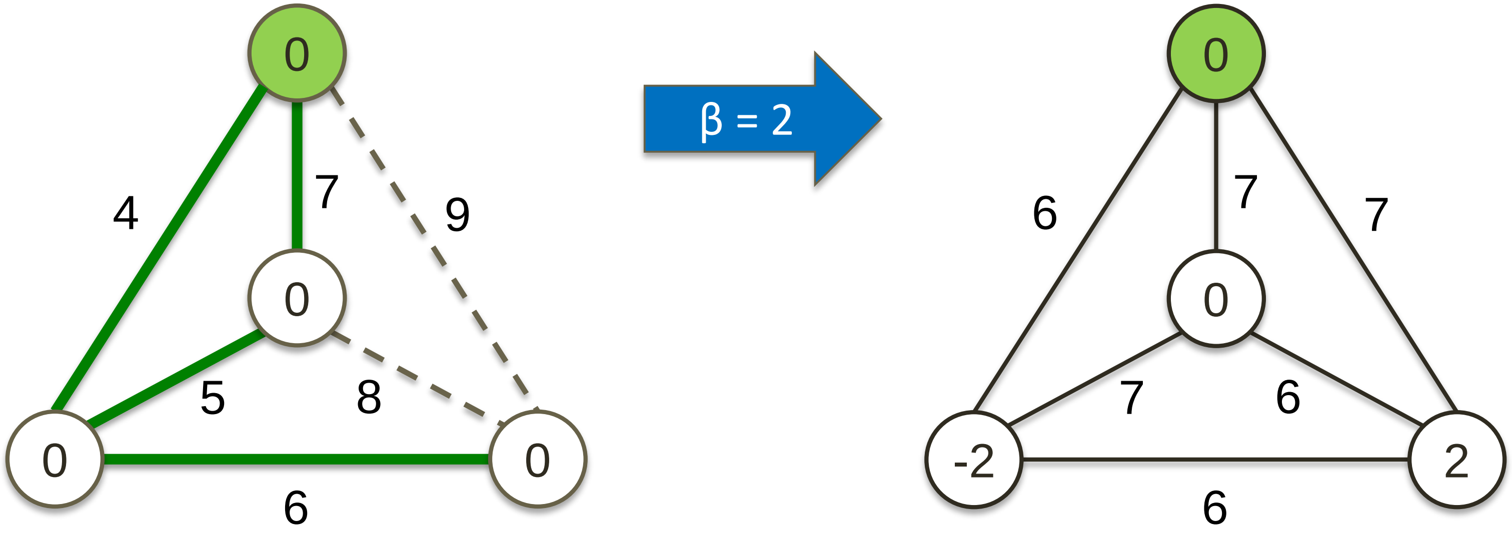
bound: 22

Example



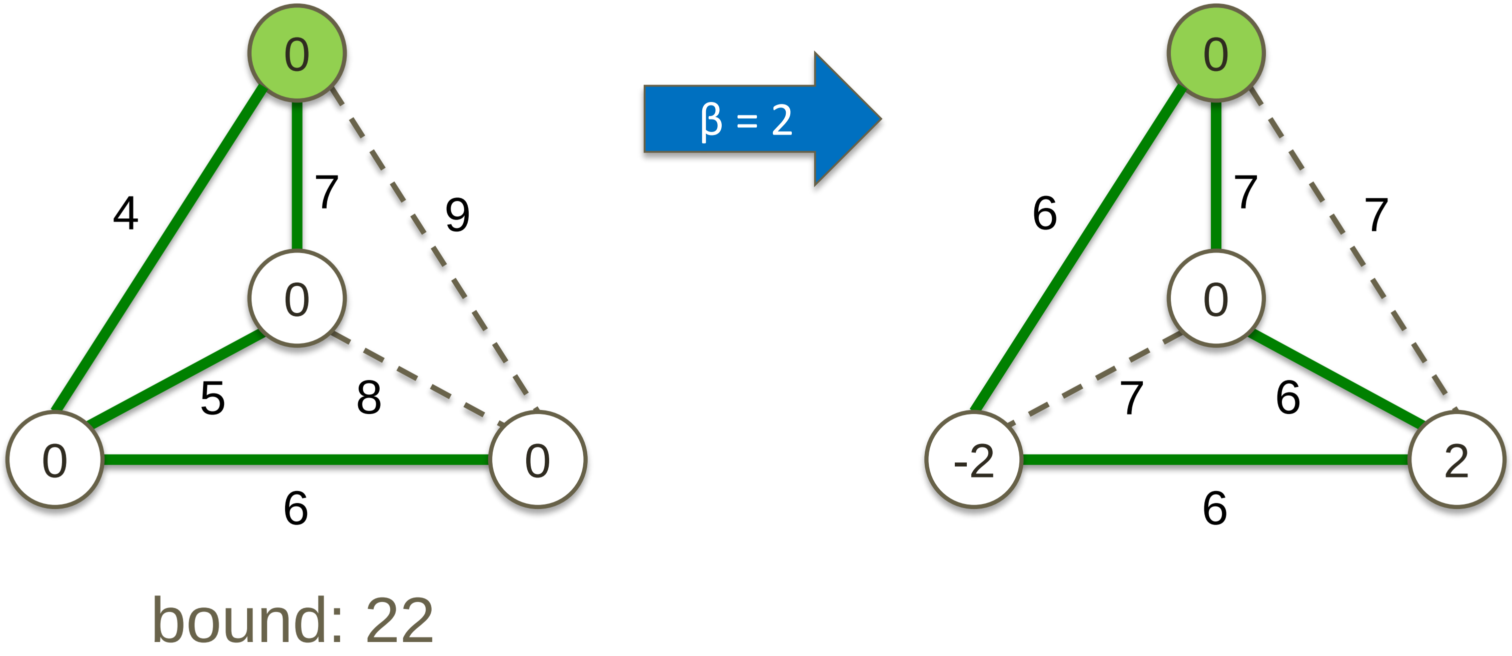
bound: 22

Example

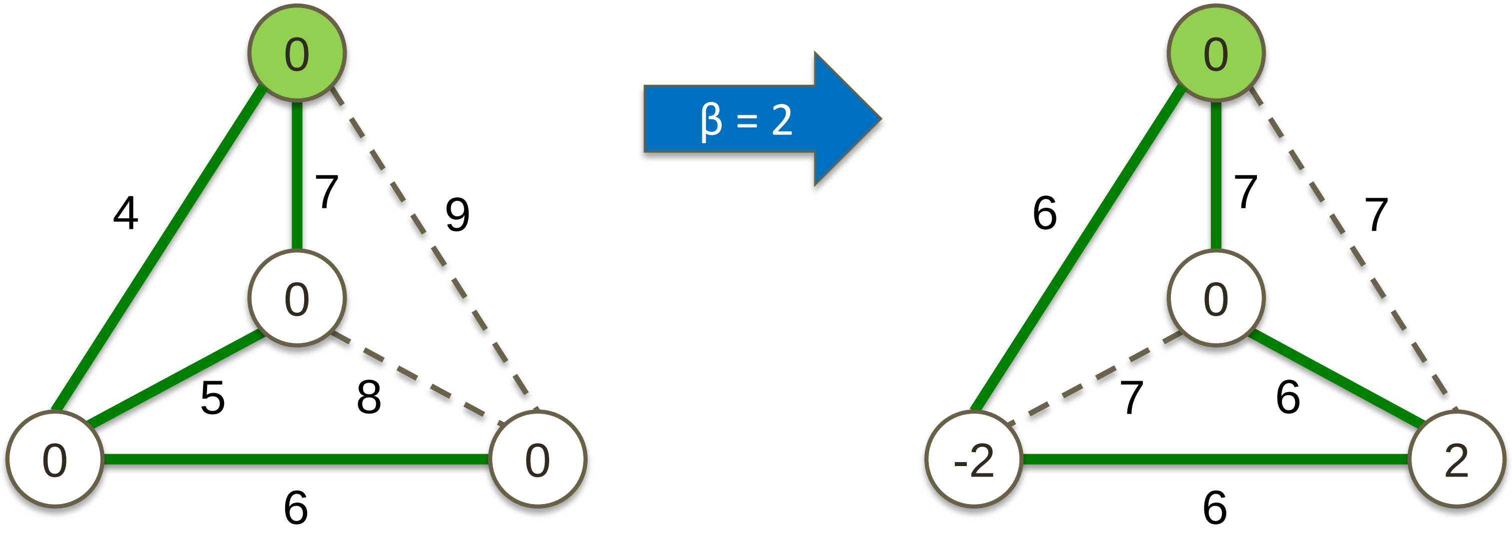


bound: 22

Example



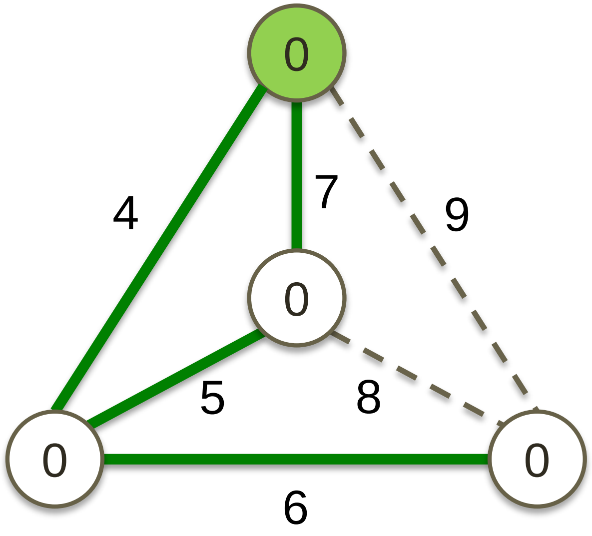
Example



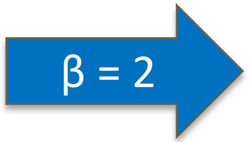
bound: 22

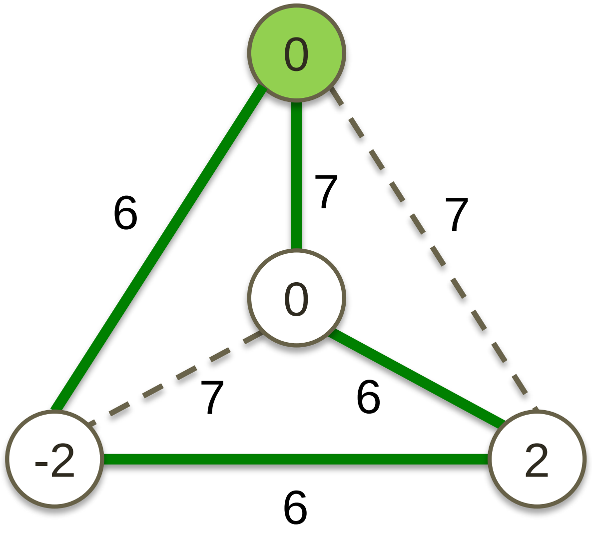
bound: 25
(optimal)

Example



bound: 22





bound: 25
(optimal)

Cost-based propagation?

Embed 1-Tree in CP

- We need to reason on the graph structure
  - manipulate the graph, remove costly edges, etc.
- Not easily done with ‘next’ and ‘position’ variables
  - e.g., how can we enforce that a given edge e=(i,j) is mandatory?
- Ideally, we want to have access to the graph rather than local successor/predecessor information
  - modify definition of our global constraint

One more CP model for the TSP

Integrated model based on graph representation

min $\quad z$

s.t. $\quad$ *weighted-circuit*(X, G, z)

- G=(V,E,w) is the graph with vertex set V, edge set E, weights w
- Set of binary variables X = { x_e | e ∊ E } representing the tour
  - In CP, X can be modeled using a 'set variable'
- Variable z represents the total weight

[Benchimol et al. 2012]

Propagation

- Given constraint

  weighted-circuit(X, G, z)

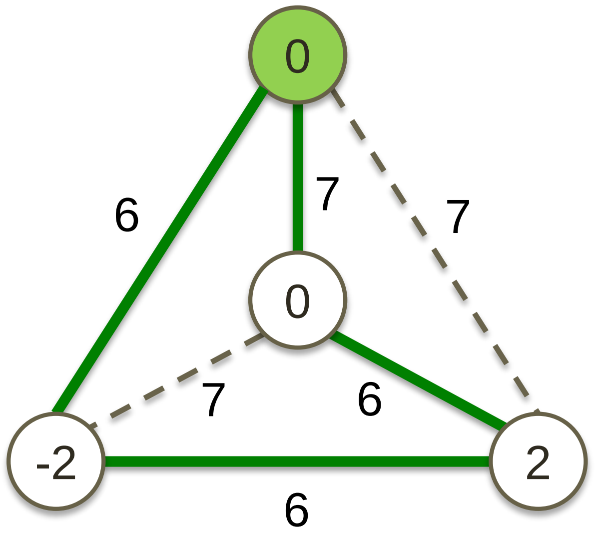
- Apply the Held-Karp relaxation to
  – remove sub-optimal edges ($x_e = 0$), i.e., total weight > max(D(z))
  – force mandatory edges ($x_e = 1$)
  – update bounds of z

Propagation

- Given constraint

  weighted-circuit(X, G, z)

- Apply the Held-Karp relaxation to
  - remove sub-optimal edges ($x_e = 0$), i.e., total weight > max(D(z))
  - force mandatory edges ($x_e = 1$)
  - update bounds of z



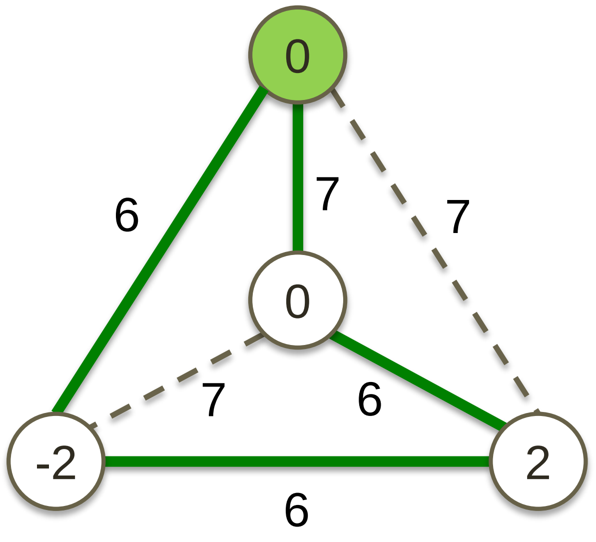
bound: 25

Propagation

- Given constraint

  weighted-circuit(X, G, z)

- Apply the Held-Karp relaxation to
  - remove sub-optimal edges ($x_e = 0$), i.e., total weight > max(D(z))
  - force mandatory edges ($x_e = 1$)
  - update bounds of z



bound: 25

Suppose D(z) = {25}

Propagation

- Given constraint

  weighted-circuit(X, G, z)

- Apply the Held-Karp relaxation to
  - remove sub-optimal edges (x_e = 0),
    i.e., total weight > max(D(z))
  - force mandatory edges (x_e = 1)
  - update bounds of z
- Effectiveness depends on multipliers!

  [Sellmann, 2004]

Suppose D(z) = {25}

Propagation

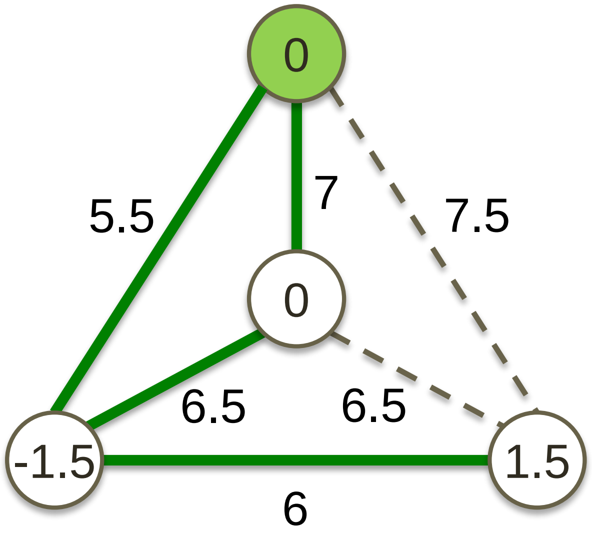
- Given constraint

  weighted-circuit(X, G, z)

- Apply the Held-Karp relaxation to
  – remove sub-optimal edges ($x_e = 0$), i.e., total weight > max(D(z))
  – force mandatory edges ($x_e = 1$)
  – update bounds of z

- Effectiveness depends on multipliers!

  [Sellmann, 2004]



bound: 25

Suppose D(z) = {25}

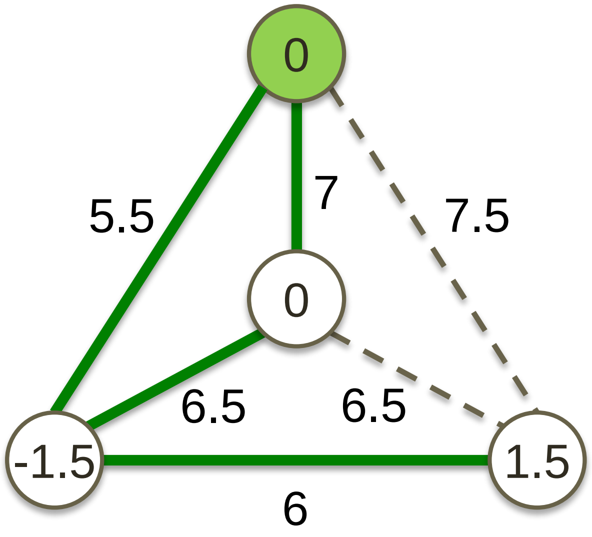
Propagation

- Given constraint

  weighted-circuit(X, G, z)

- Apply the Held-Karp relaxation to
  – remove sub-optimal edges ($x_e = 0$), i.e., total weight > max(D(z))
  – force mandatory edges ($x_e = 1$)
  – update bounds of z
- Effectiveness depends on multipliers!

  [Sellmann, 2004]

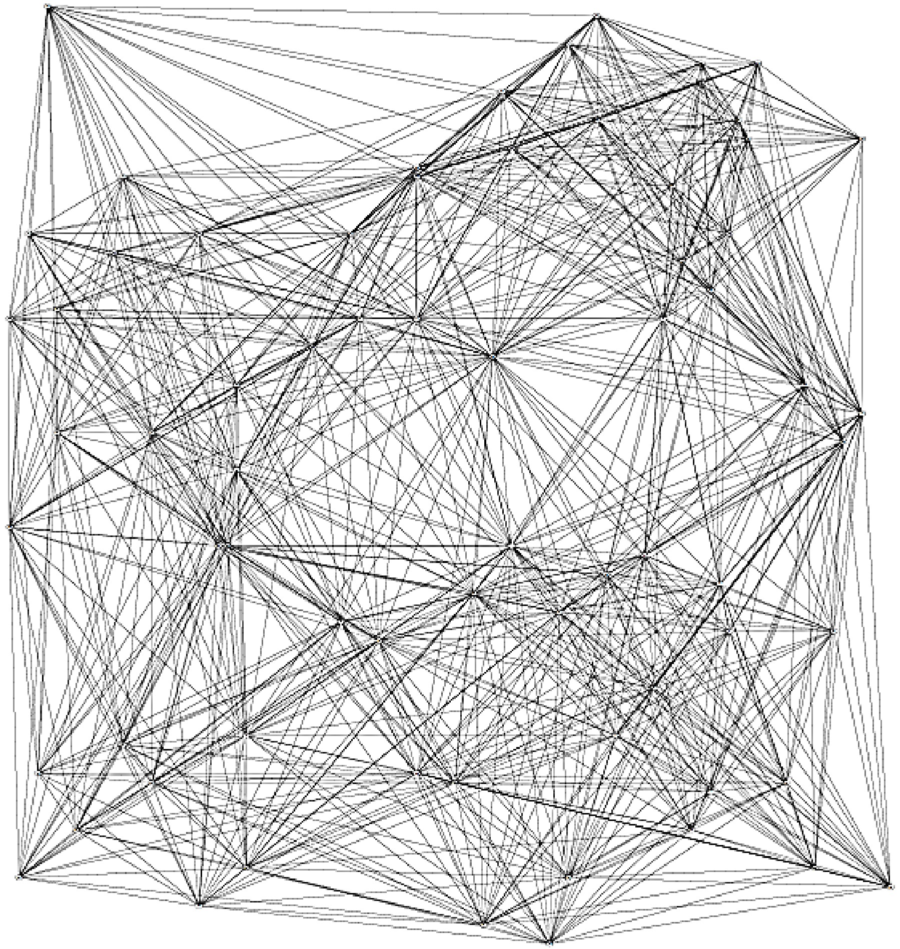


bound: 25

Suppose D(z) = {25}

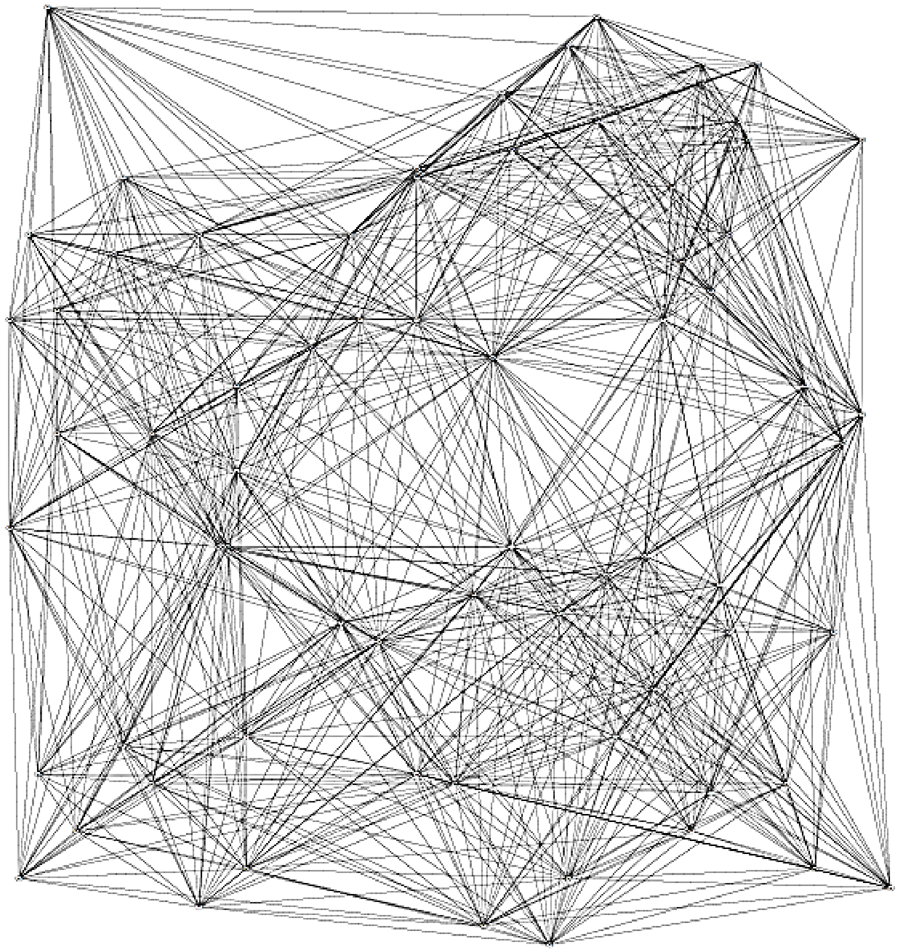
[Benchimol et al. 2010, 2012] [Regin et al. 2010]

Impact of Propagation (st70 from TSPLIB)

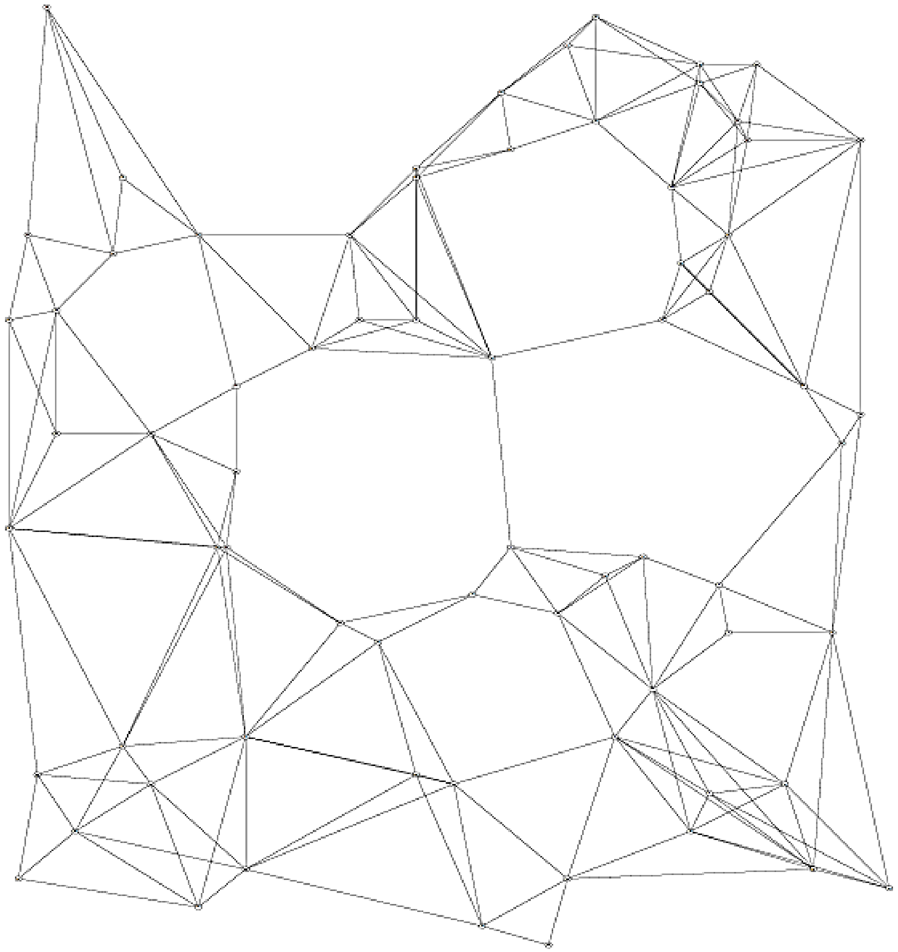


upper bound = 700

Impact of Propagation (st70 from TSPLIB)



upper bound = 700



upper bound = 675

Impact on Complete (Exact) Solver

	1-tree no filtering			1-tree with filtering			Concorde		
size	solved	time	nodes/s	solved	time	nodes/s	solved	time	nodes/s
50	1.00	0.13	299.26	1.00	0.03	712.39	1.00	0.18	19.59
100	1.00	3.19	55.10	1.00	0.34	160.65	1.00	0.31	6.10
150	1.00	18.31	13.83	1.00	1.42	46.91	1.00	0.59	4.52
200	1.00	132.30	5.16	1.00	4.68	33.00	1.00	0.97	3.18
250	0.97	409.88	2.13	1.00	10.98	25.76	1.00	1.98	2.83
300	0.80	770.67	1.38	1.00	24.35	20.29	1.00	2.32	2.15
350	0.67	1,239.25	0.61	1.00	39.54	15.96	1.00	3.74	1.92
400	0.33	1,589.71	0.42	0.97	108.45	11.04	1.00	4.57	1.64
450	0.17	1,722.56	0.34	1.00	121.08	12.16	1.00	4.99	1.68
500	0.00	1,800.00	0.21	0.97	194.32	8.81	1.00	6.42	1.38
550	0.00	1,800.00	0.20	0.97	206.99	7.98	1.00	5.00	1.00

randomly generated symmetric TSPs, time limit 1800s

average over 30 instances per size class

Impact on Complete (Exact) Solver

	1-tree no filtering			1-tree with filtering			Concorde		
size	solved	time	nodes/s	solved	time	nodes/s	solved	time	nodes/s
50	1.00	0.13	299.26	1.00	0.03	712.39	1.00	0.18	19.59
100	1.00	3.19	55.10	1.00	0.34	160.65	1.00	0.31	6.10
150	1.00	18.31	13.83	1.00	1.42	46.91	1.00	0.59	4.52
200	1.00	132.30	5.16	1.00	4.68	33.00	1.00	0.97	3.18
250	0.97	409.88	2.13	1.00	10.98	25.76	1.00	1.98	2.83
300	0.80	770.67	1.38	1.00	24.35	20.29	1.00	2.32	2.15
350	0.67	1,239.25	0.61	1.00	39.54	15.96	1.00	3.74	1.92
400	0.33	1,589.71	0.42	0.97	108.45	11.04	1.00	4.57	1.64
450	0.17	1,722.56	0.34	1.00	121.08	12.16	1.00	4.99	1.68
500	0.00	1,800.00	0.21	0.97	194.32	8.81	1.00	6.42	1.38
550	0.00	1,800.00	0.20	0.97	206.99	7.98	1.00	5.00	1.00

randomly generated symmetric TSPs, time limit 1800s

average over 30 instances per size class

Impact on Complete (Exact) Solver

	1-tree no filtering			1-tree with filtering			Concorde		
size	solved	time	nodes/s	solved	time	nodes/s	solved	time	nodes/s
50	1.00	0.13	299.26	1.00	0.03	712.39	1.00	0.18	19.59
100	1.00	3.19	55.10	1.00	0.34	160.65	1.00	0.31	6.10
150	1.00	18.31	13.83	1.00	1.42	46.91	1.00	0.59	4.52
200	1.00	132.30	5.16	1.00	4.68	33.00	1.00	0.97	3.18
250	0.97	409.88	2.13	1.00	10.98	25.76	1.00	1.98	2.83
300	0.80	770.67	1.38	1.00	24.35	20.29	1.00	2.32	2.15
350	0.67	1,239.25	0.61	1.00	39.54	15.96	1.00	3.74	1.92
400	0.33	1,589.71	0.42	0.97	108.45	11.04	1.00	4.57	1.64
450	0.17	1,722.56	0.34	1.00	121.08	12.16	1.00	4.99	1.68
500	0.00	1,800.00	0.21	0.97	194.32	8.81	1.00	6.42	1.38
550	0.00	1,800.00	0.20	0.97	206.99	7.98	1.00	5.00	1.00

randomly generated symmetric TSPs, time limit 1800s

average over 30 instances per size class

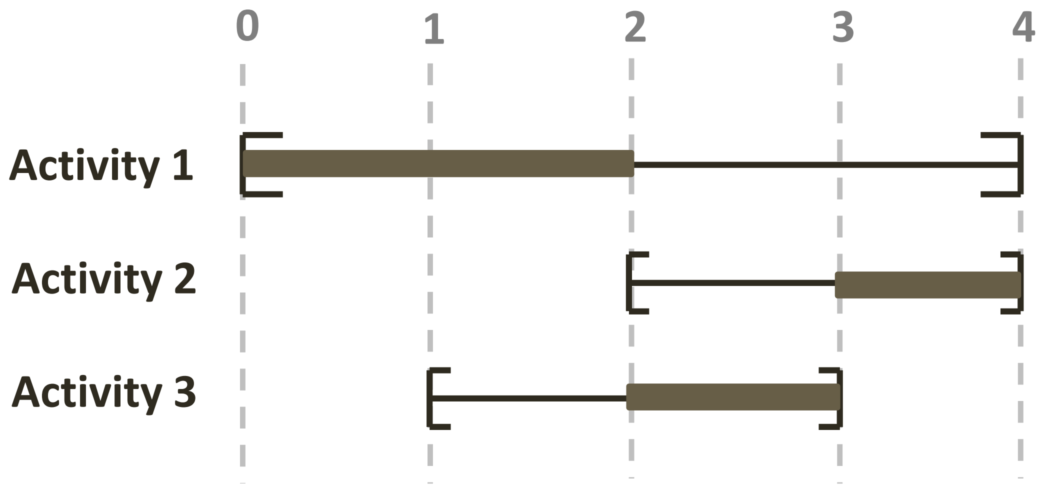
Extended to ATSP [Fages & Lorca, 2012] and degree-constraint MST [Fages et al., 2016]

Many More Applications

- TSP [Caseau et al., 1997]
- TSPTW [Focacci et al., 2000, 2002]
- Traveling Tournament Problem [Benoist et al., 2001]
- Capacitated Network Design [Sellmann et al., 2002]
- Automated Recording Problem [Sellmann et al, 2003]
- Network Design [Cronholm et al, 2004]
- Resource-Constrained Shortest Path Problem [Gellermann et al., 2005] [Gualandi, 2012]
- Personnel Scheduling [Menana et al., 2009]
- Multileaf Sequencing Collimator [Cambazard et al., 2009]
- Parallel Machine Scheduling [Edis et al., 2011]
- Traveling Purchaser Problem [Cambazard, 2012]
- Empirical Model Learning [Lombardi et al., 2013]
- NPV in Resource-Constrained Projects [Gu et al., 2013]

Disjunctive Scheduling / Sequencing

- Sequencing and scheduling of activities on a resource
- *Activities*
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i
- *Resource*
  - Nonpreemptive
  - Process one activity at a time



Variants and Extension

- Precedence relations between activities
- Sequence-dependent setup times
- Various objective functions
  - Makespan
  - Sum of setup times
  - (Weighted) sum of completion times
  - (Weighted) tardiness
  - number of late jobs
  - …

Variants and Extension

- Precedence relations between activities
- Sequence-dependent setup times
- Various objective functions
  - Makespan
  - Sum of setup times
  - (Weighted) sum of completion times
  - (Weighted) tardiness
  - number of late jobs
  - …

Includes:
- TSP with time windows
- single-machine scheduling
- sequential ordering problem
- …

Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

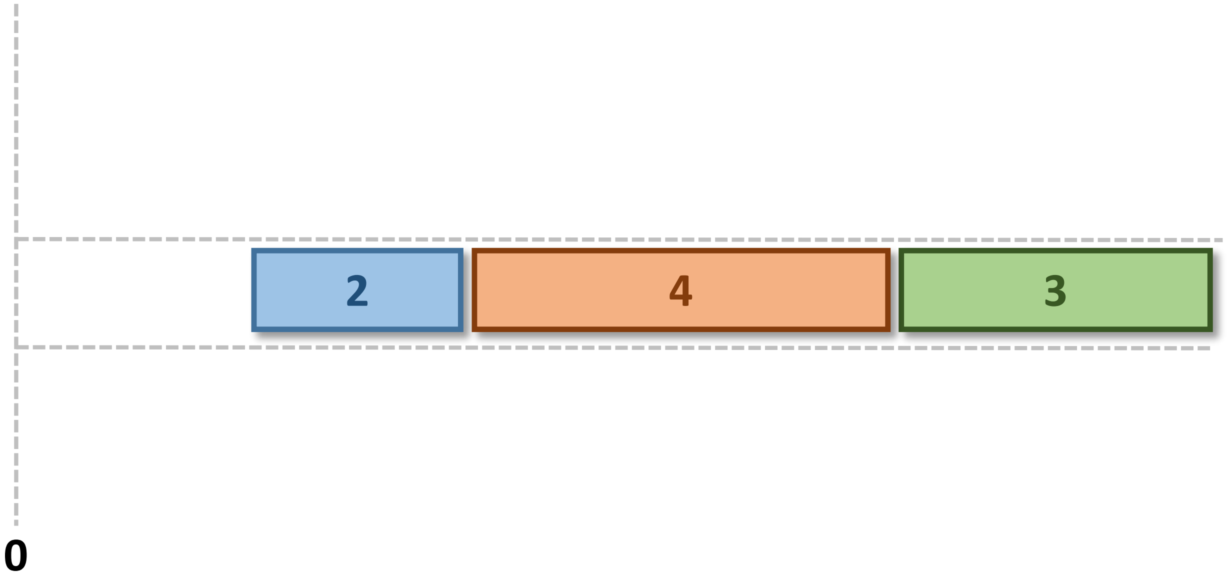
- 3 tasks to perform on a machine
- Each task has a processing time and a release time
- The machine can perform at most one task at a time, non-preemptively
- **Objective: minimize completion time**

Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

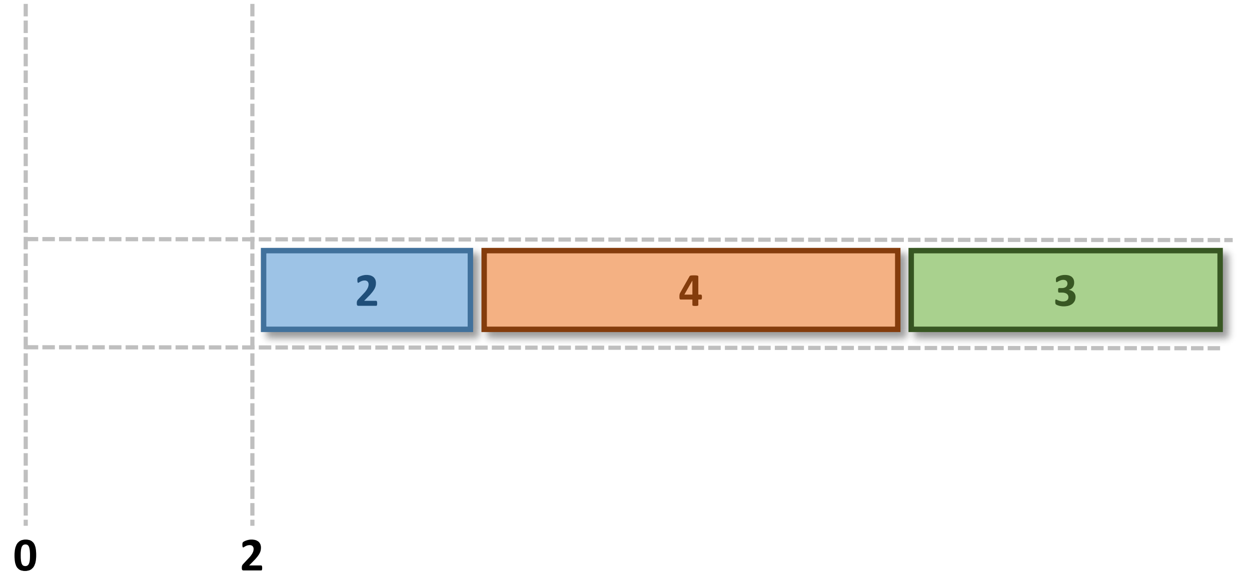
Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



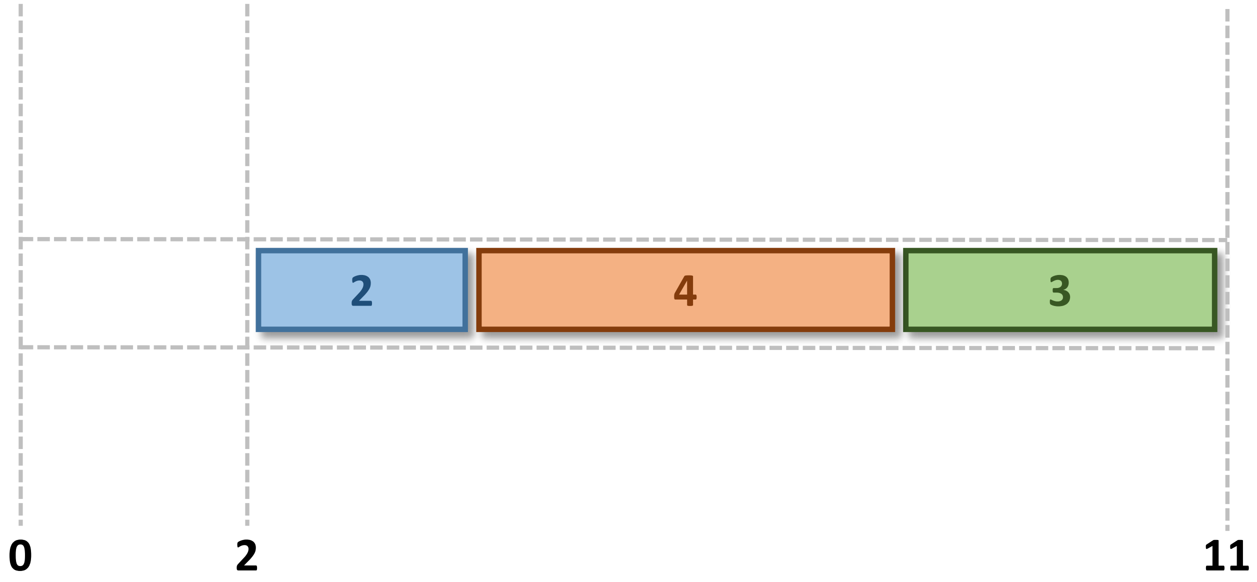
Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



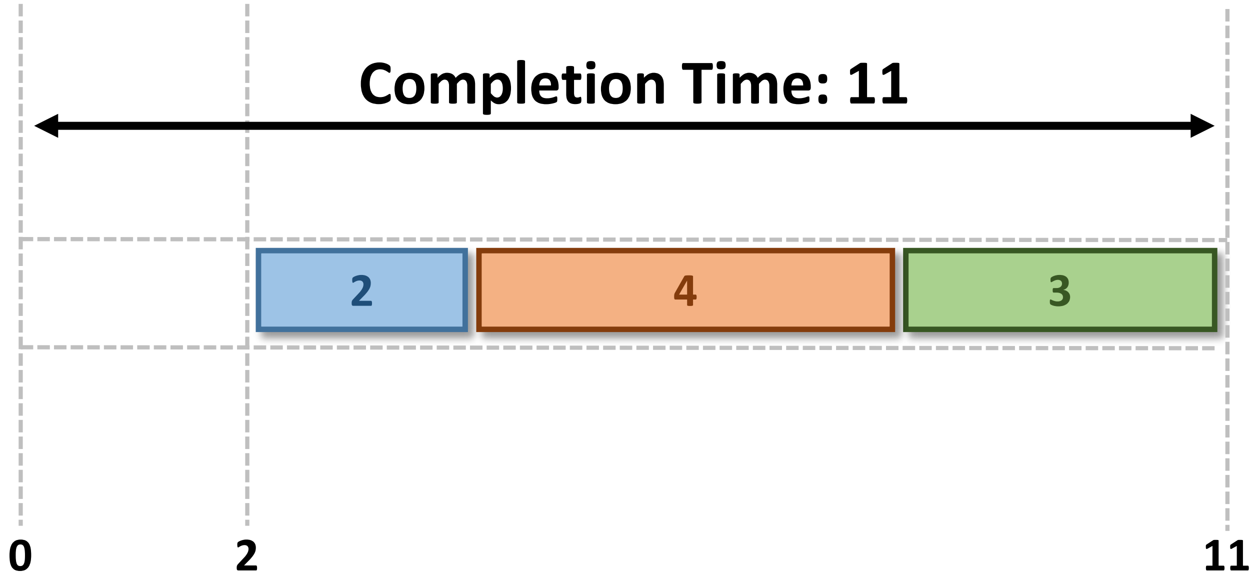
Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

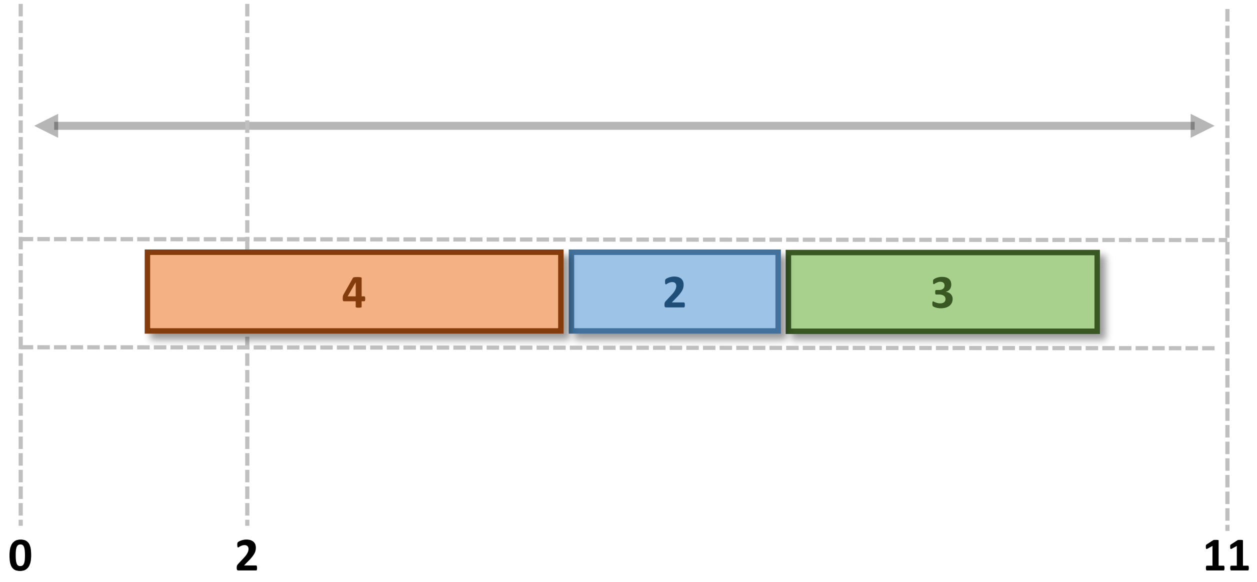


Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

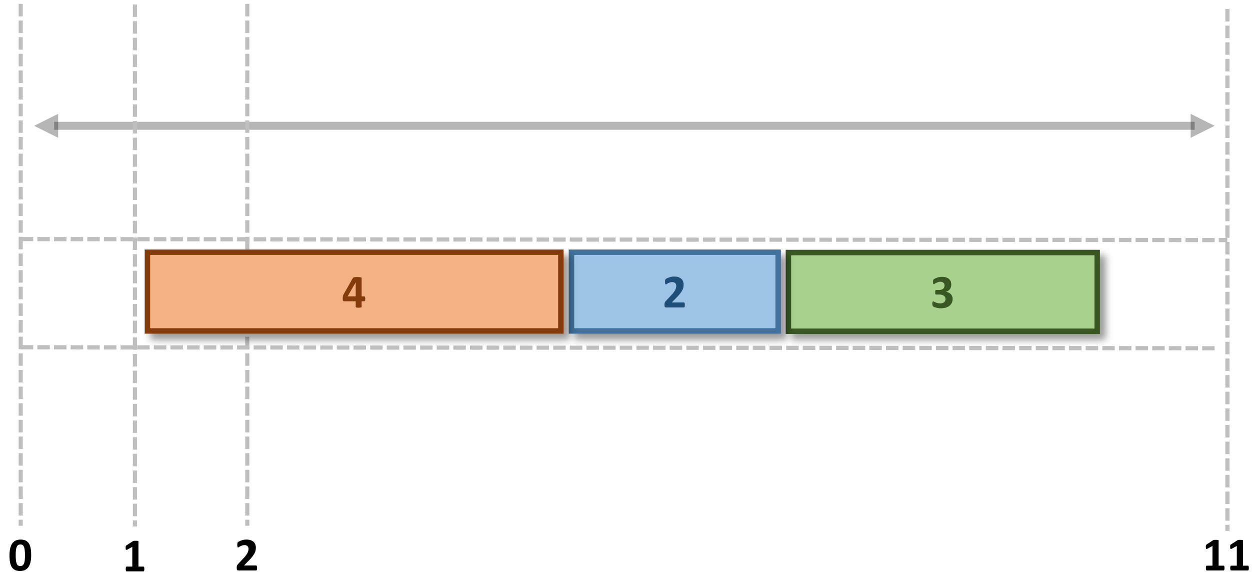
Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



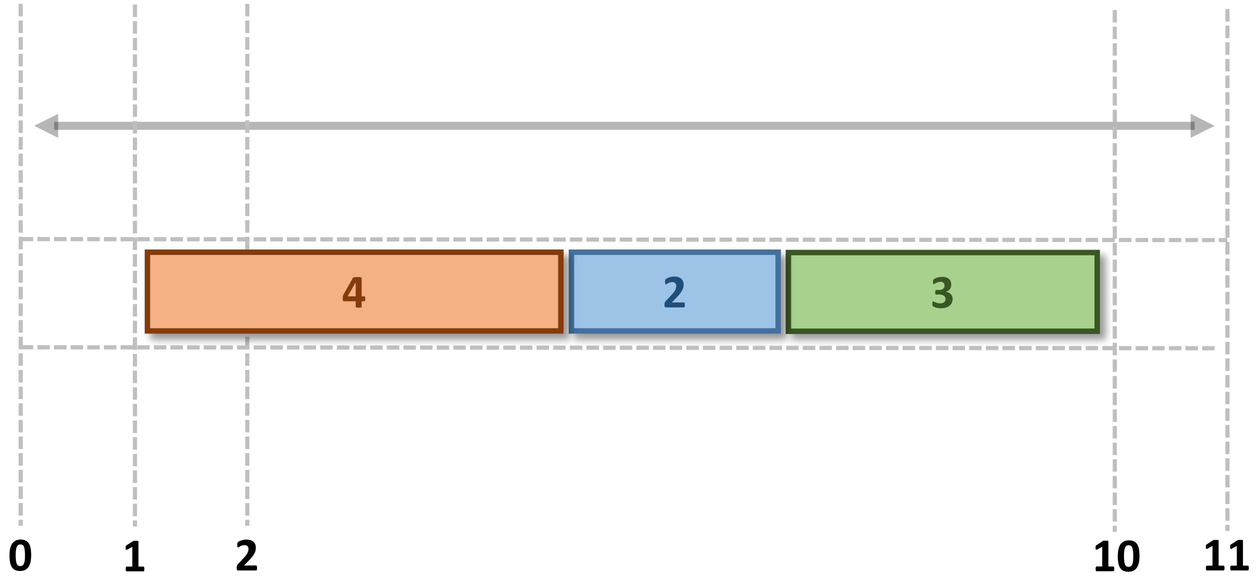
Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



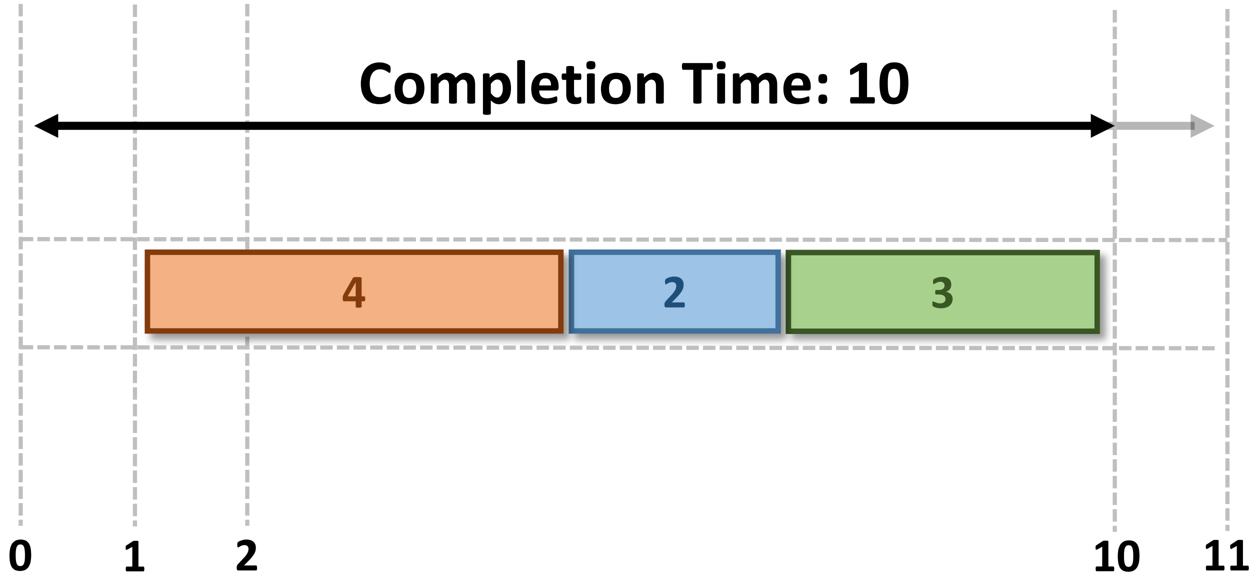
Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



Running Example

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1



CP Model

min $\quad C_{max}$

s.t. $\quad$ *NoOverlap*$(s_j \mid p_j : \text{all } j)$

$\quad C_{max} = \max\{s_1+p_1, \ldots, s_n+p_n\}$

$\quad s_i \in \{r_i, \ldots, T\}$, all i

Conventional propagation

- Use Cartesian product of variable domains as relaxation
- Propagate domains to strengthen relaxation

Relaxed Decision Diagram

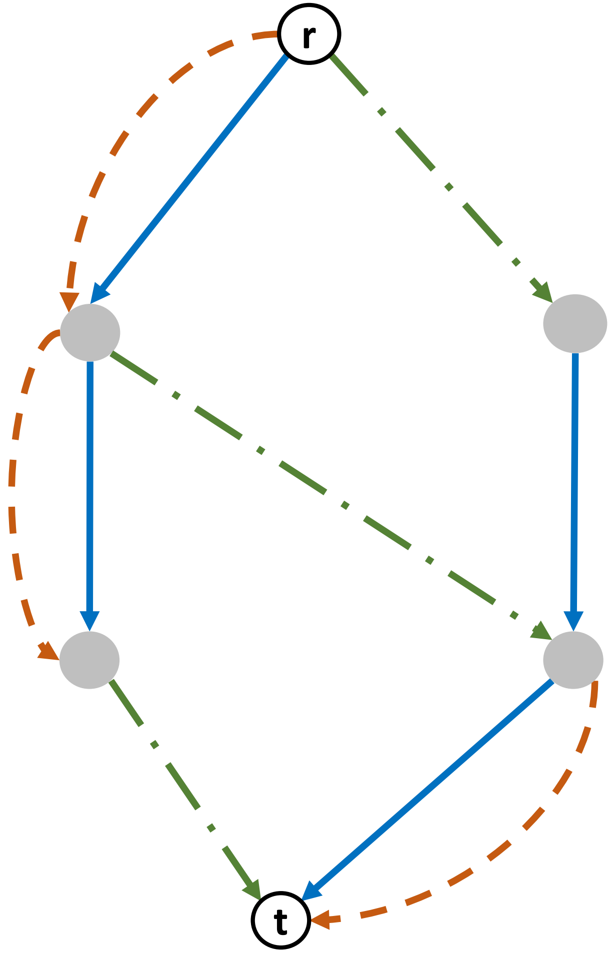
Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

Relaxed Decision Diagram

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

- **Layered Acyclic Graph**

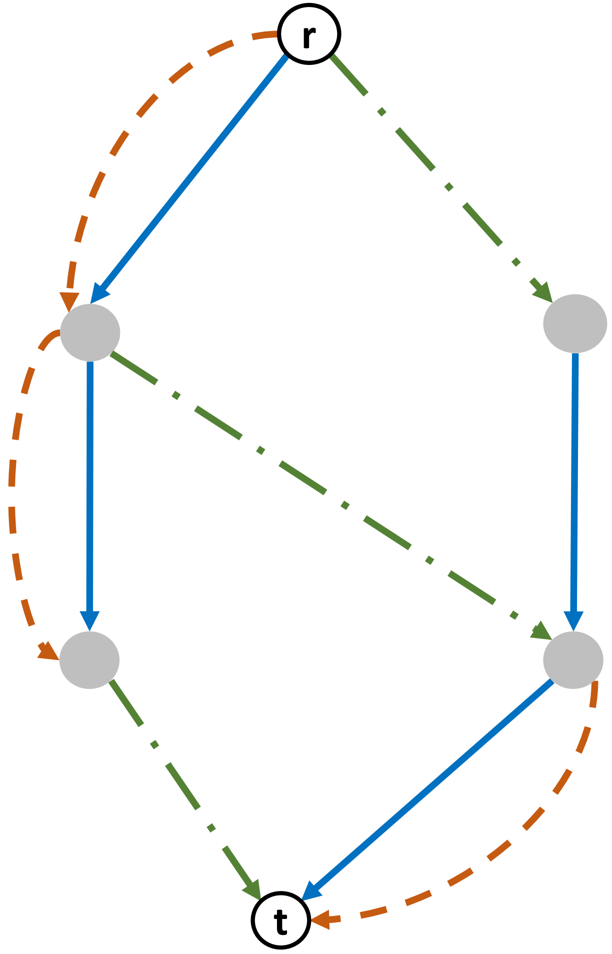
Relaxed Decision Diagram

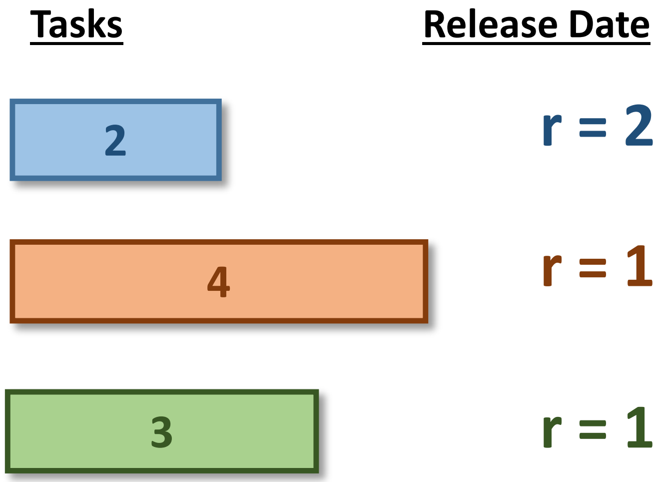


Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

- **Layered Acyclic Graph**

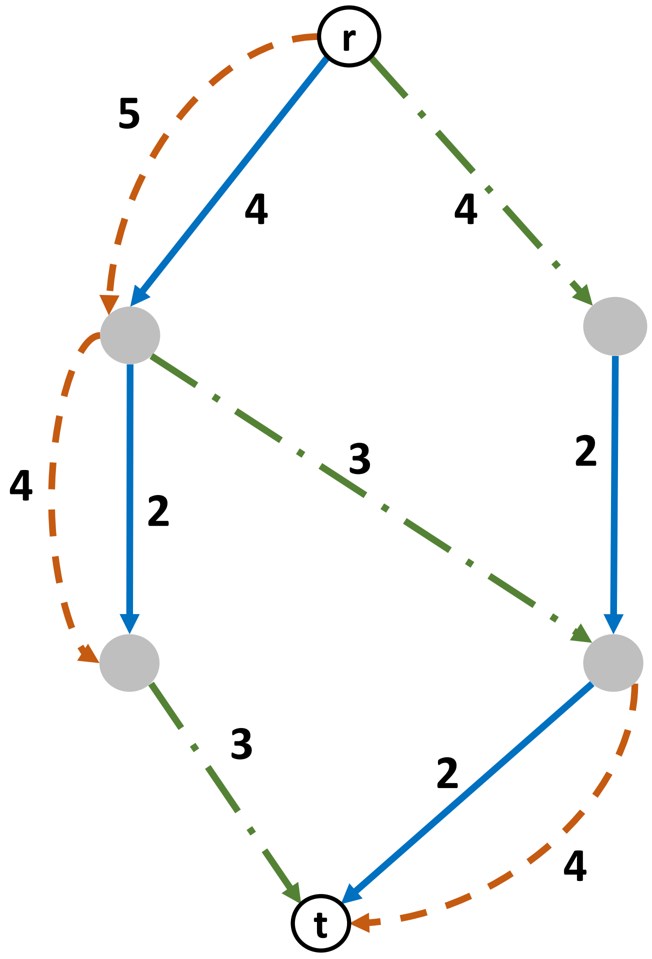
Relaxed Decision Diagram

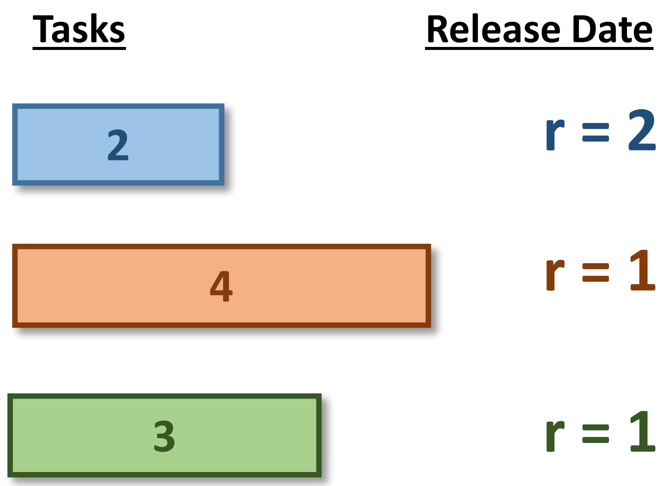




- **Layered Acyclic Graph**
- Arcs have **weights**

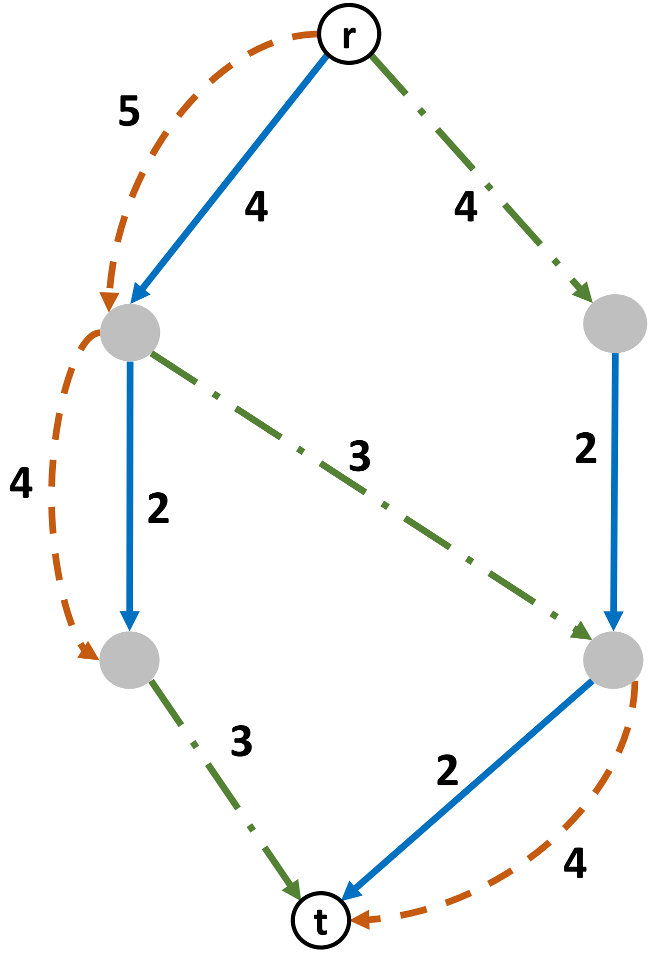
Relaxed Decision Diagram

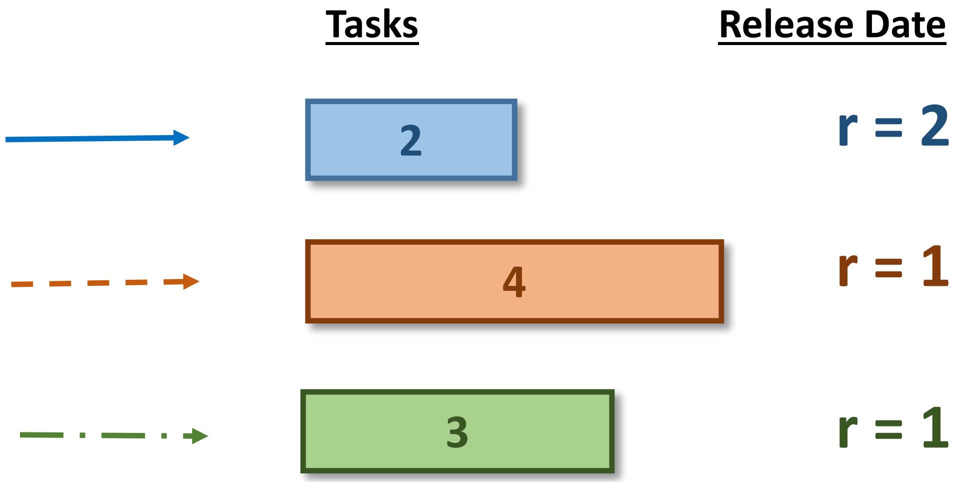




- **Layered Acyclic Graph**
- Arcs have **weights**

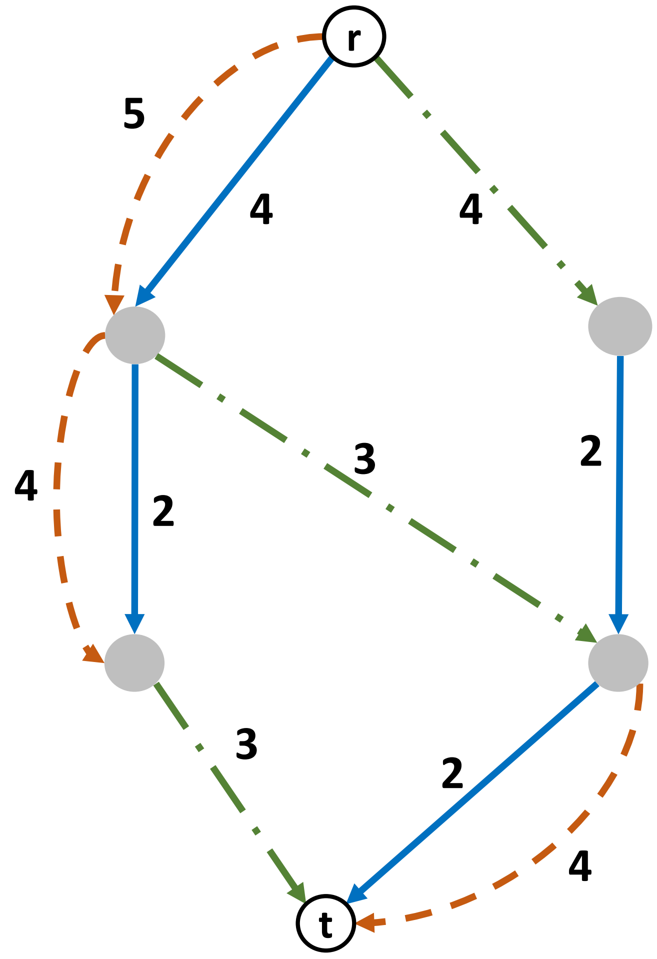
Relaxed Decision Diagram



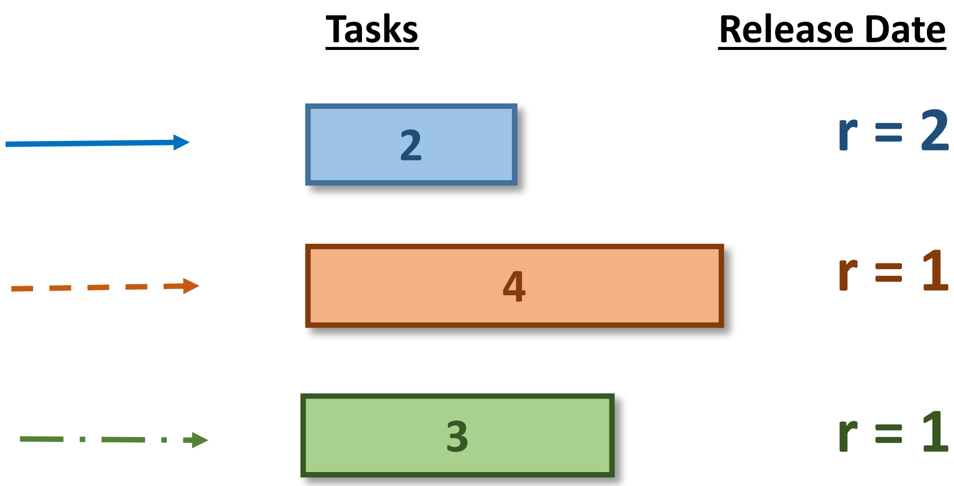


- **Layered Acyclic Graph**
- Arcs have **weights**
- **Paths** correspond to solutions, **path lengths** the solution costs

Relaxed Decision Diagram

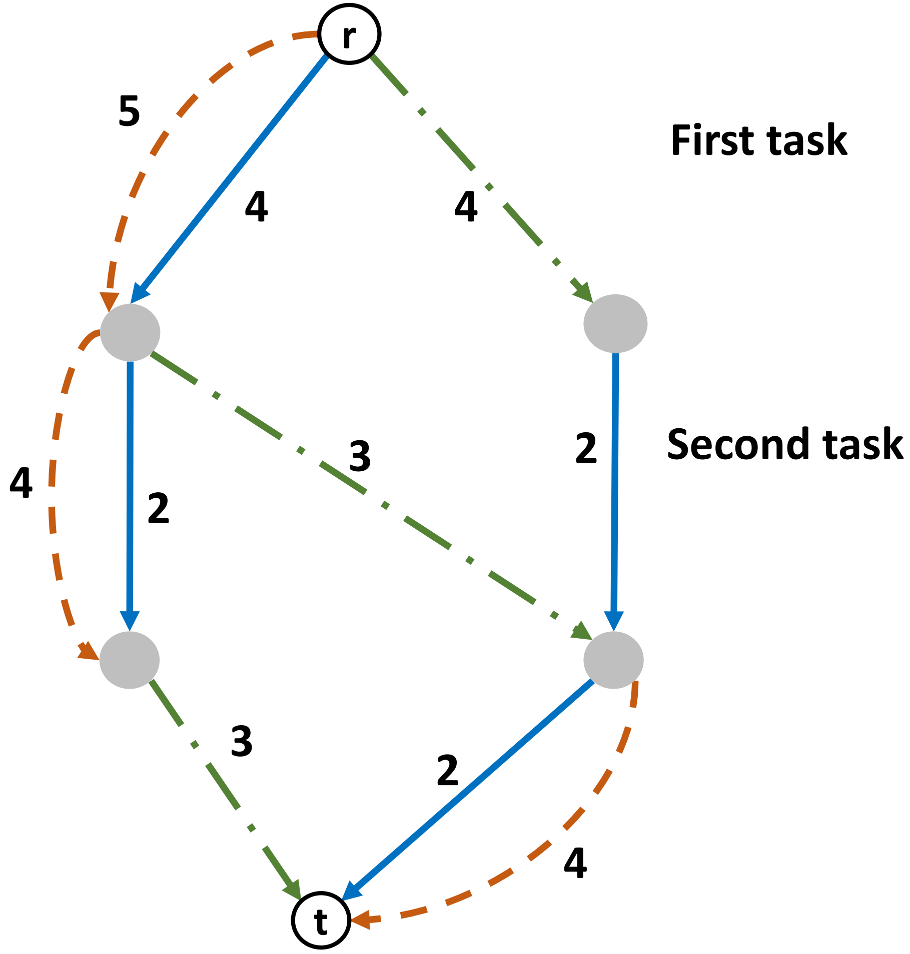


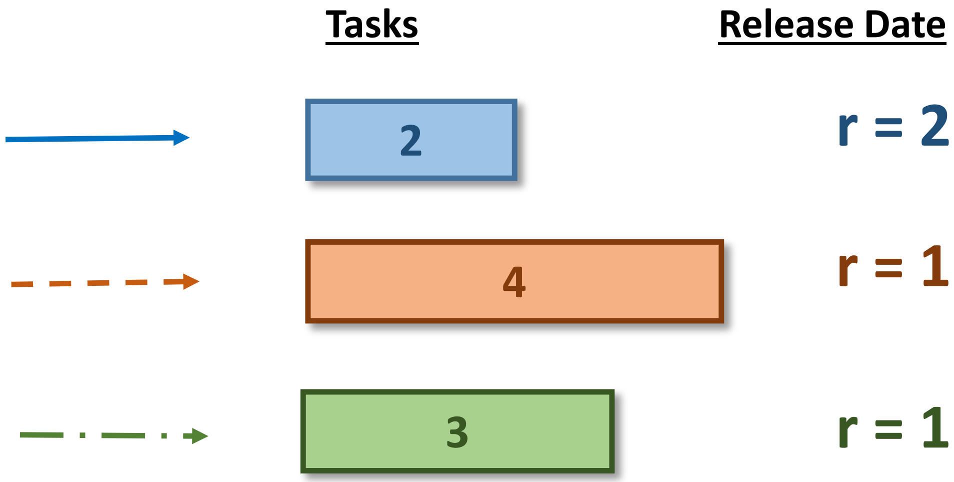
First task



- **Layered Acyclic Graph**
- Arcs have **weights**
- **Paths** correspond to solutions, **path lengths** the solution costs

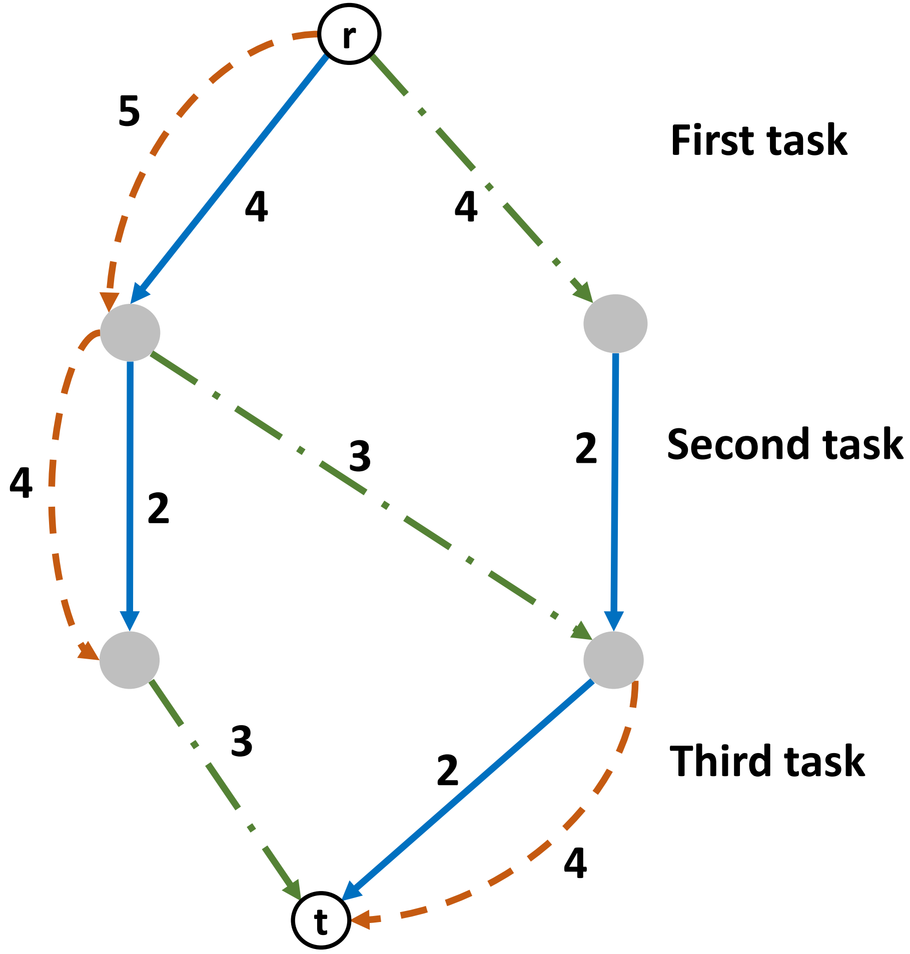
Relaxed Decision Diagram

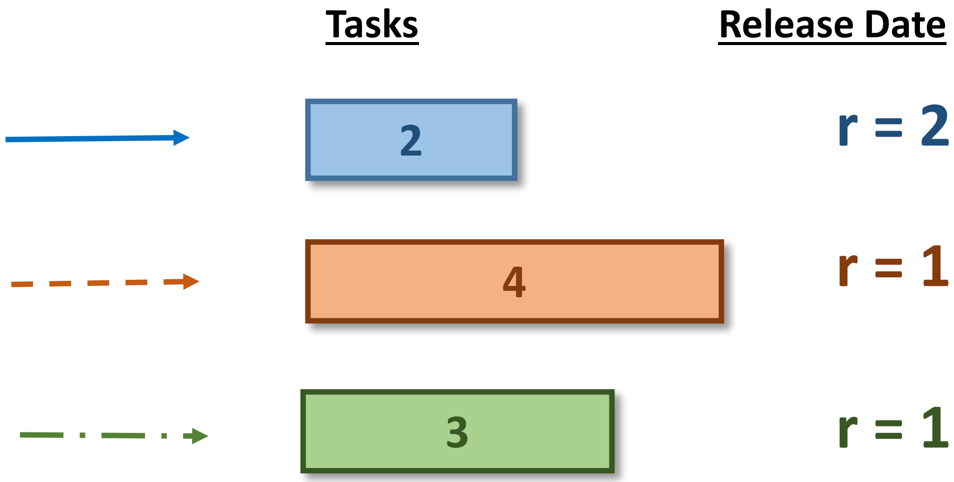




- **Layered Acyclic Graph**
- Arcs have **weights**
- **Paths** correspond to solutions, **path lengths** the solution costs

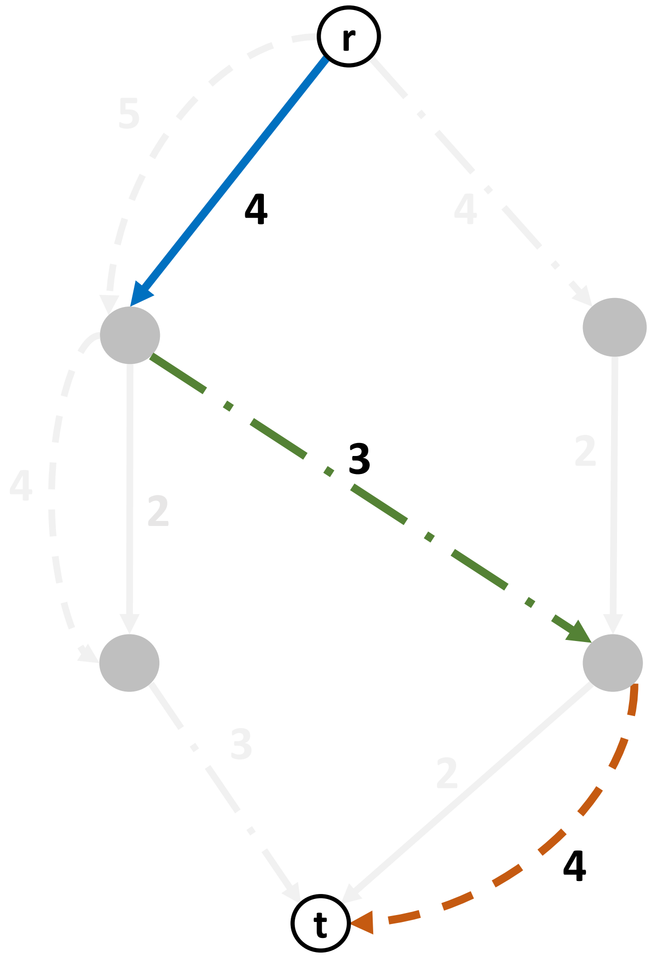
Relaxed Decision Diagram





- **Layered Acyclic Graph**
- Arcs have **weights**
- **Paths** correspond to solutions, **path lengths** the solution costs

Relaxed Decision Diagram



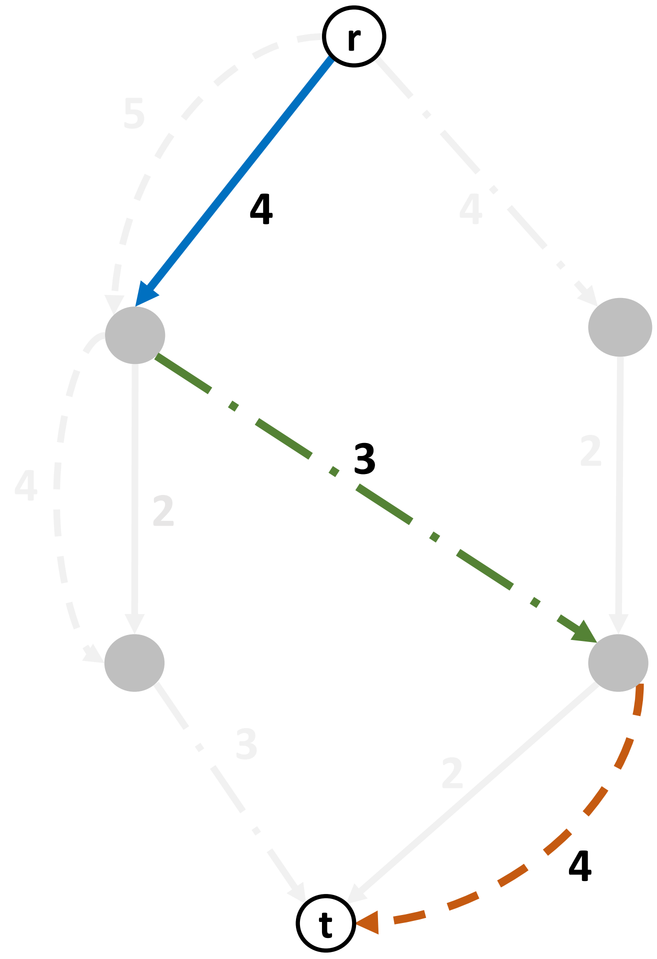
First task

Second task

Third task

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

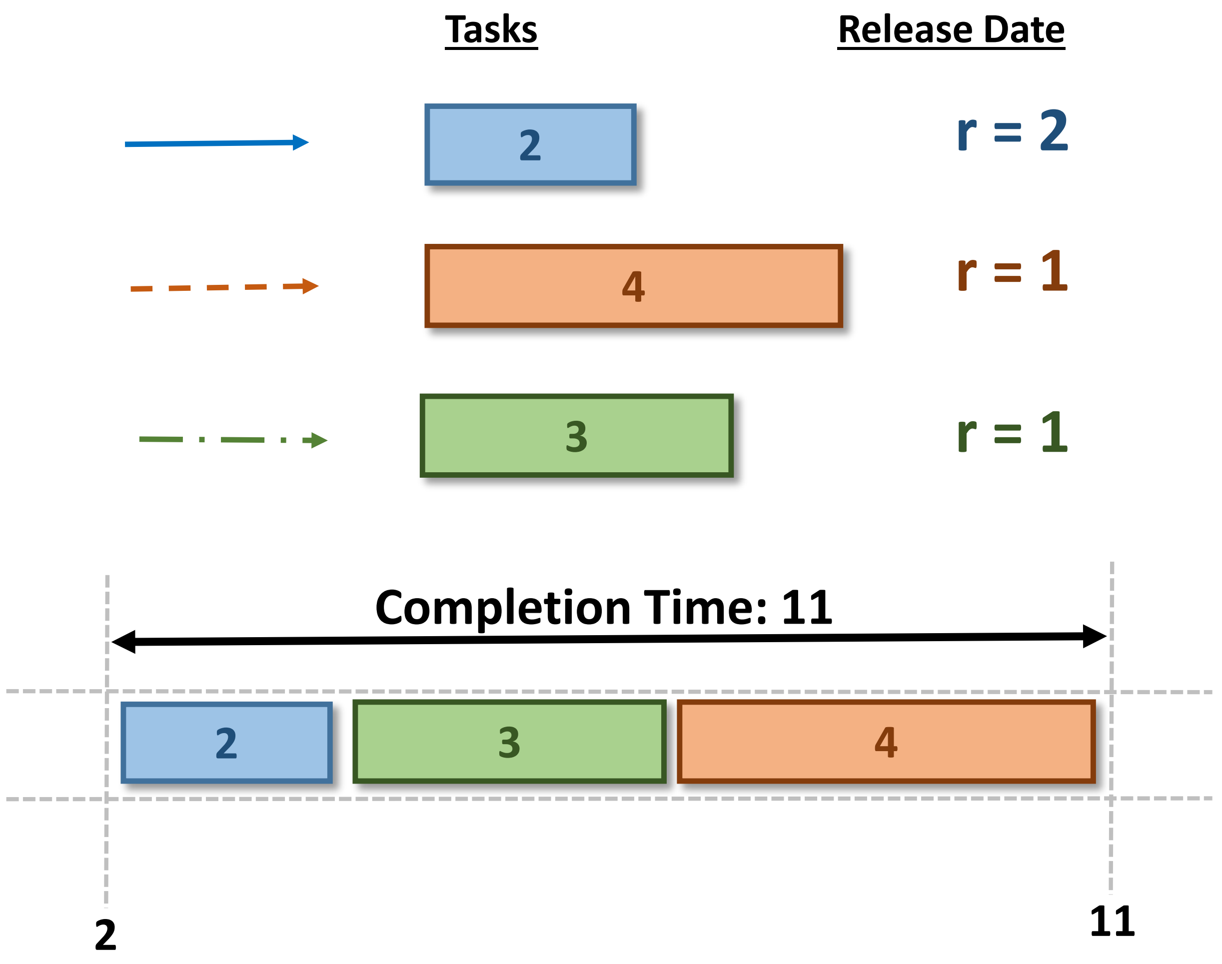
Relaxed Decision Diagram



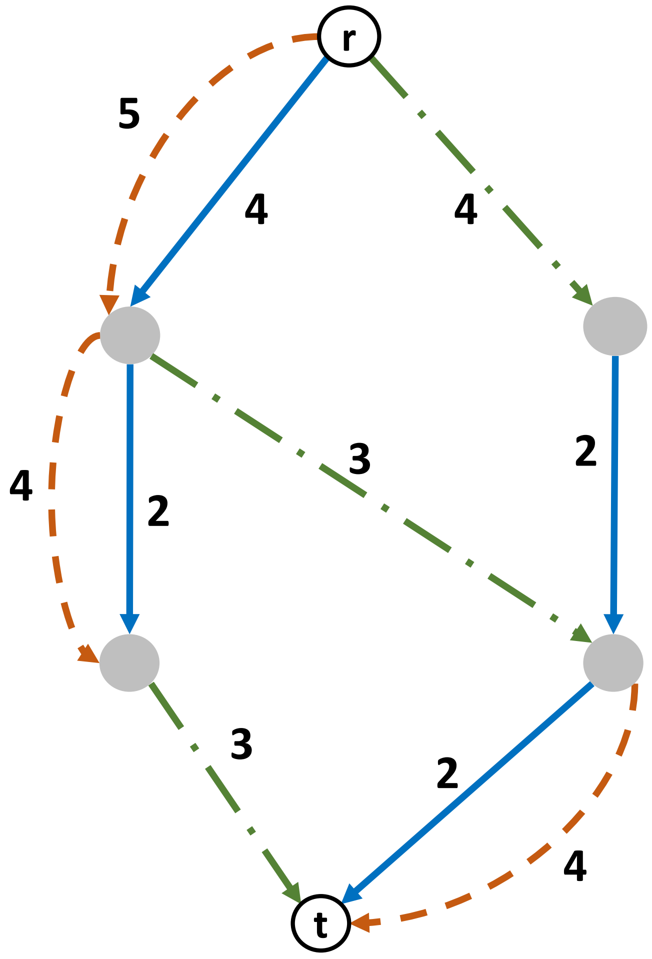
First task

Second task

Third task



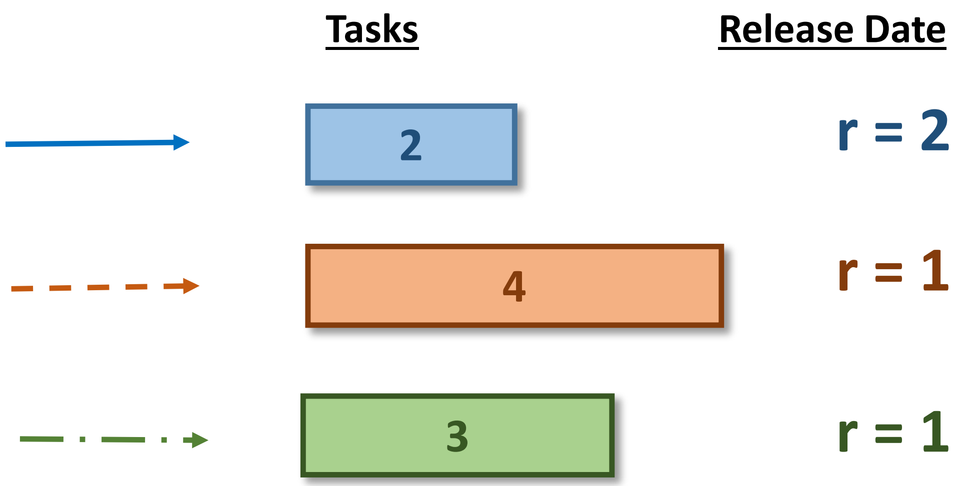
Relaxed Decision Diagram



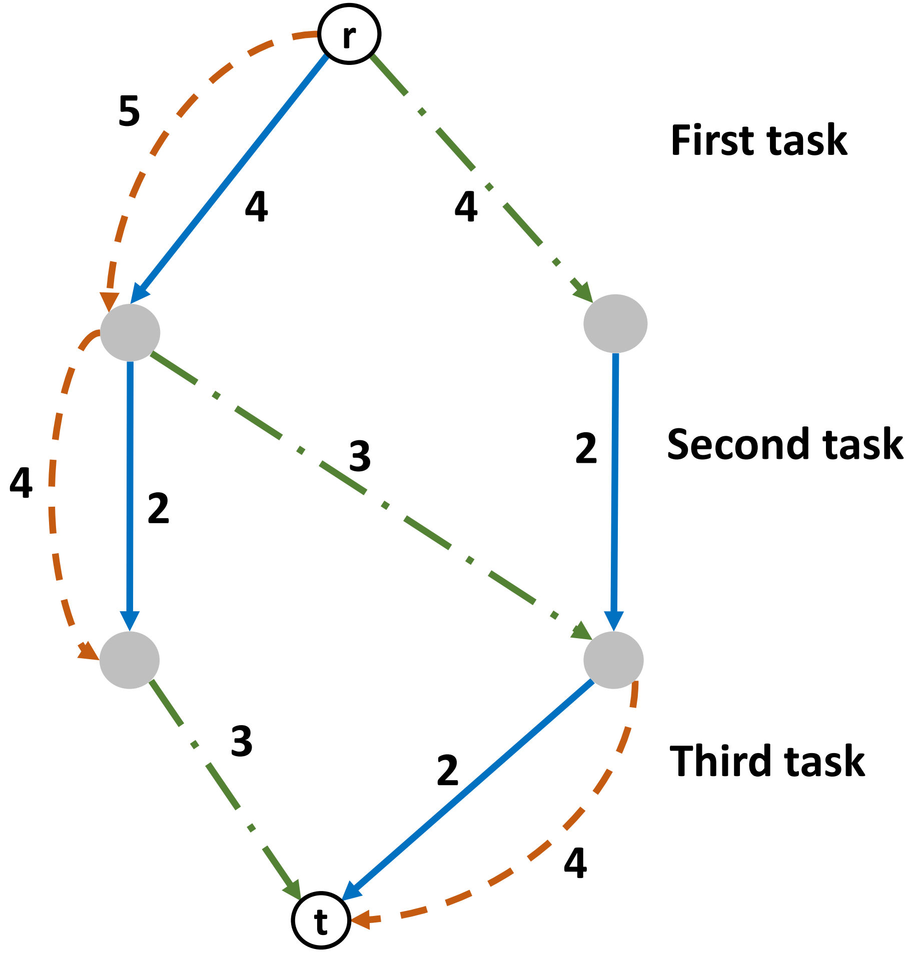
First task

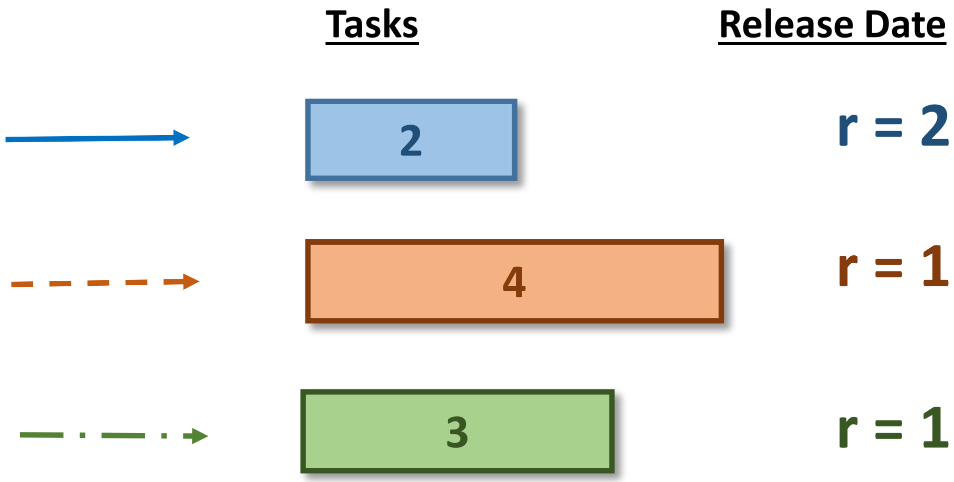
Second task

Third task



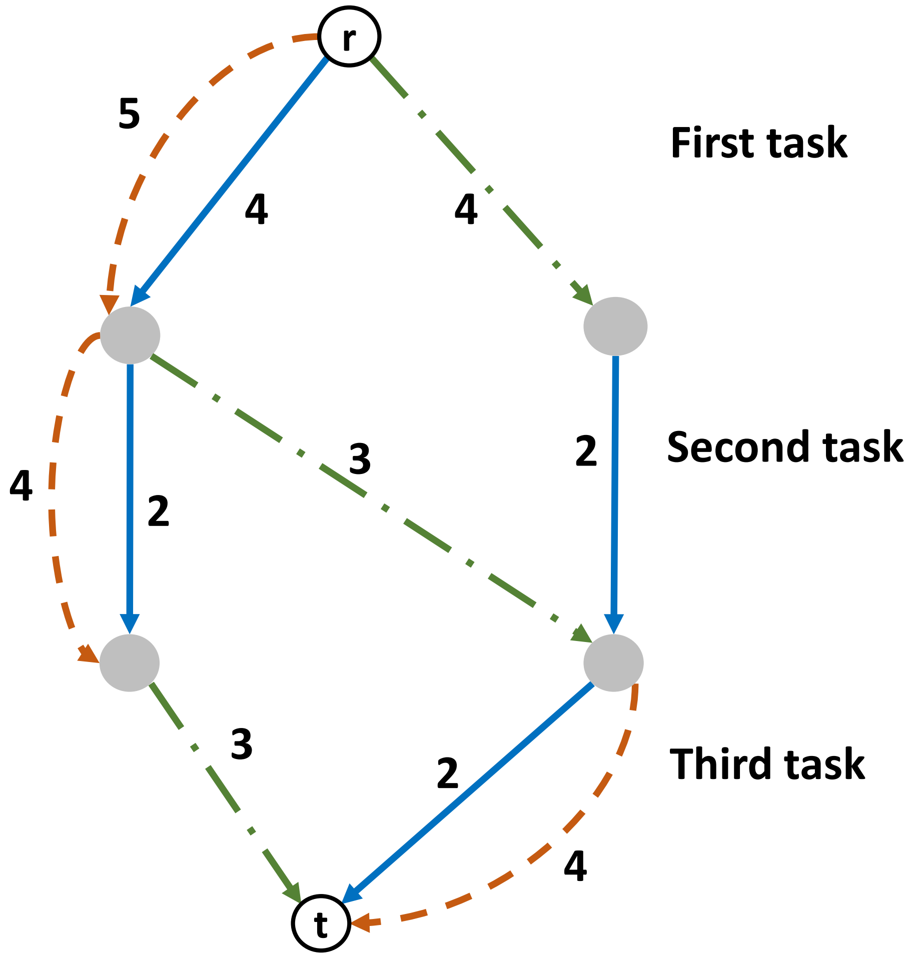
Relaxed Decision Diagram

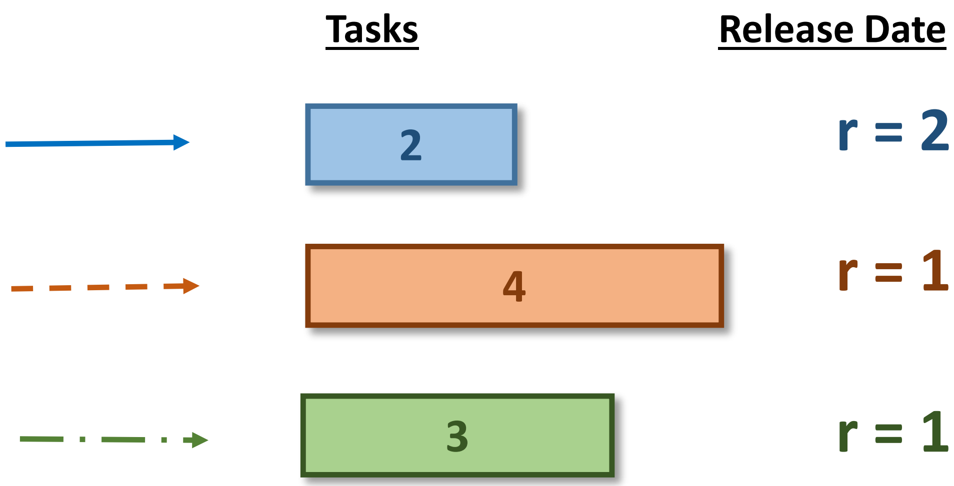




It is a **relaxation**

Relaxed Decision Diagram

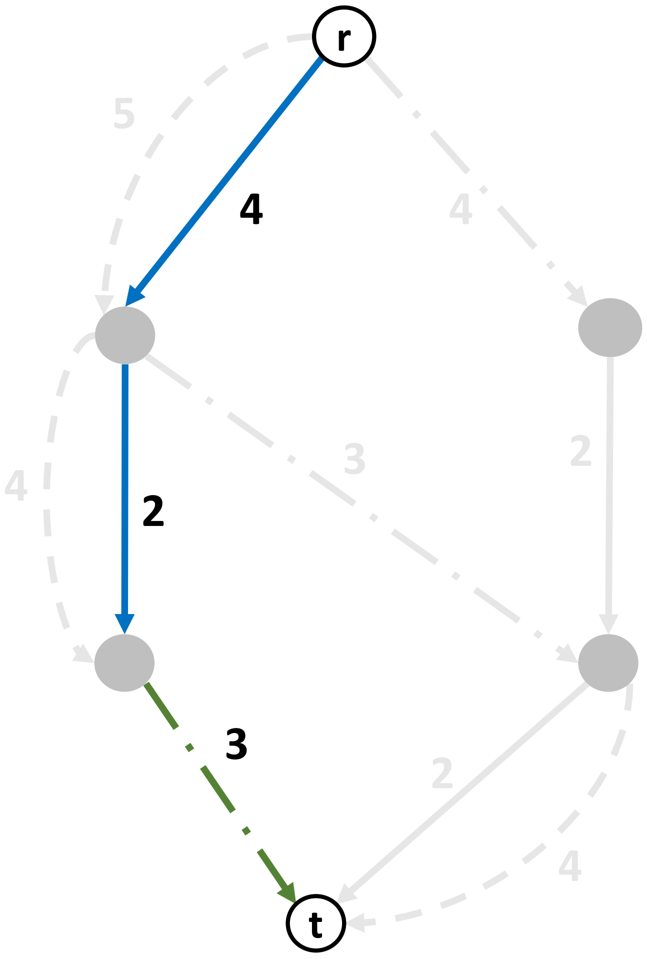




It is a **relaxation**

- **All feasible solutions** encoded by some path, but it may contain some **infeasible solutions** as well

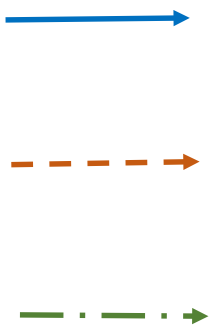
Relaxed Decision Diagram



First task

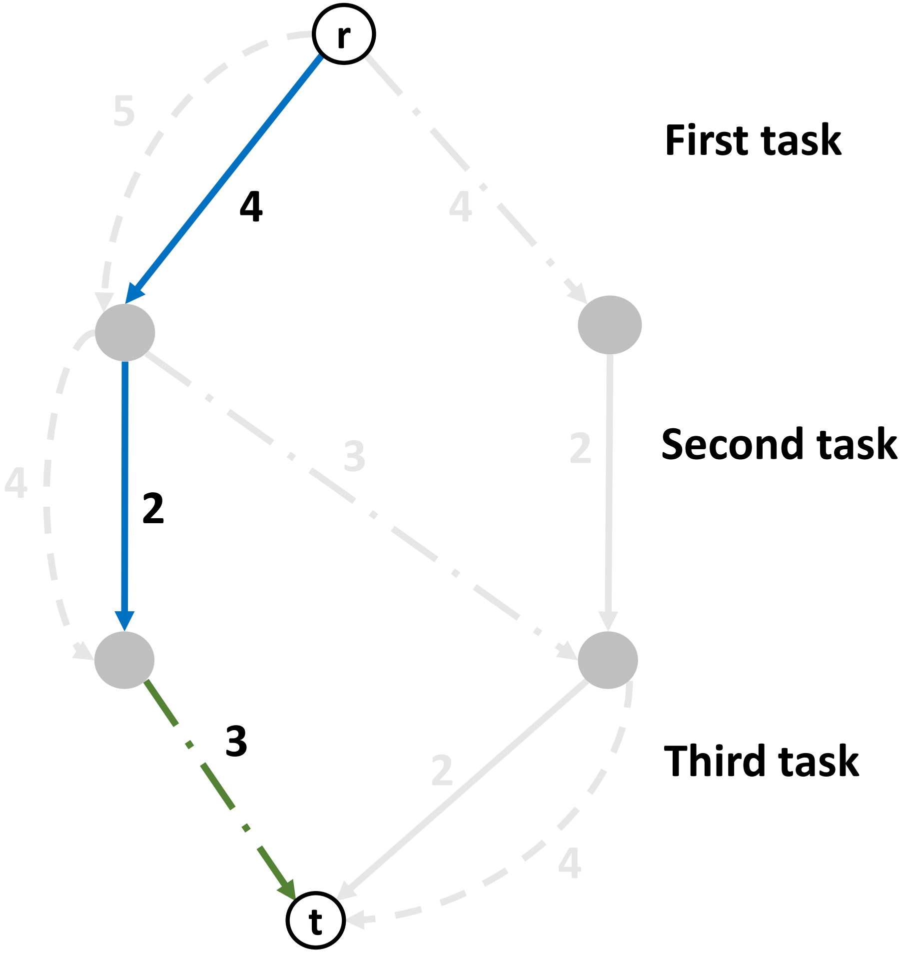
Second task

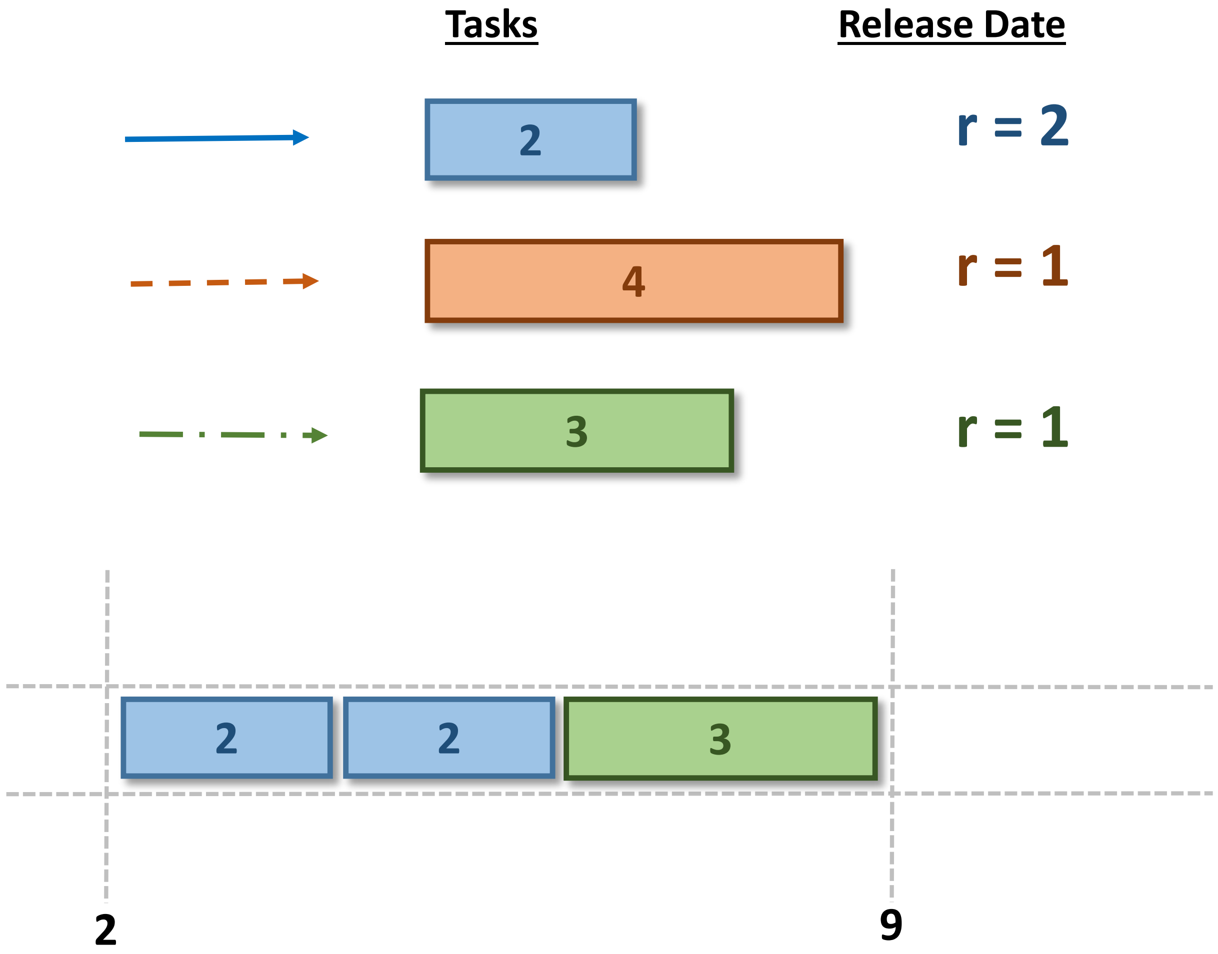
Third task



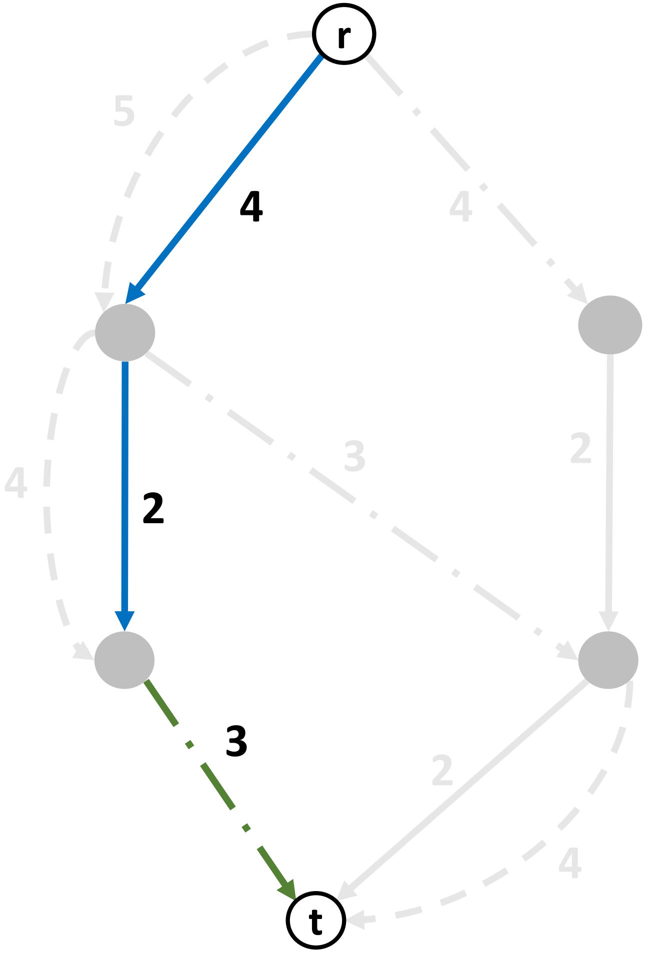
Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

Relaxed Decision Diagram





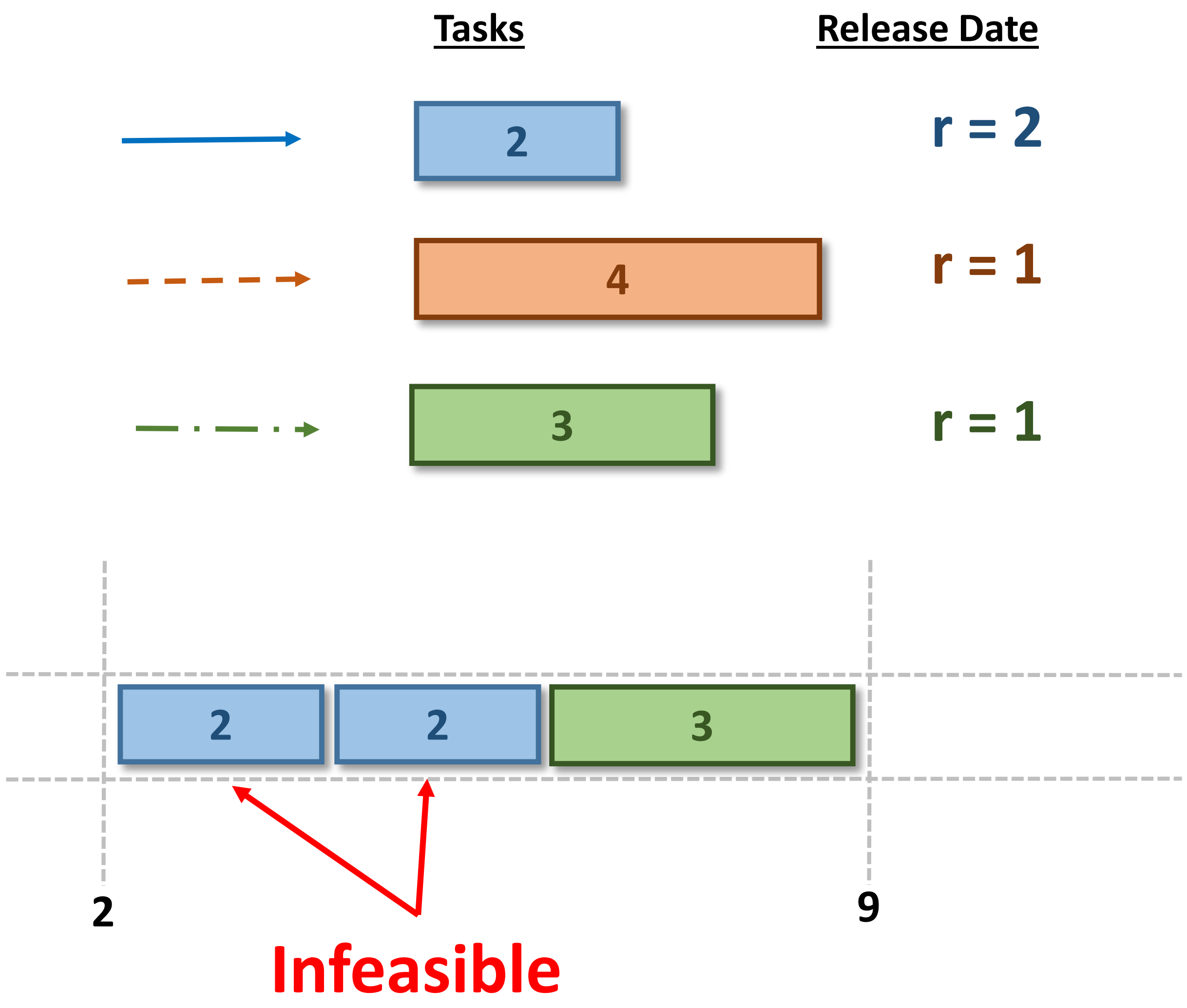
Relaxed Decision Diagram



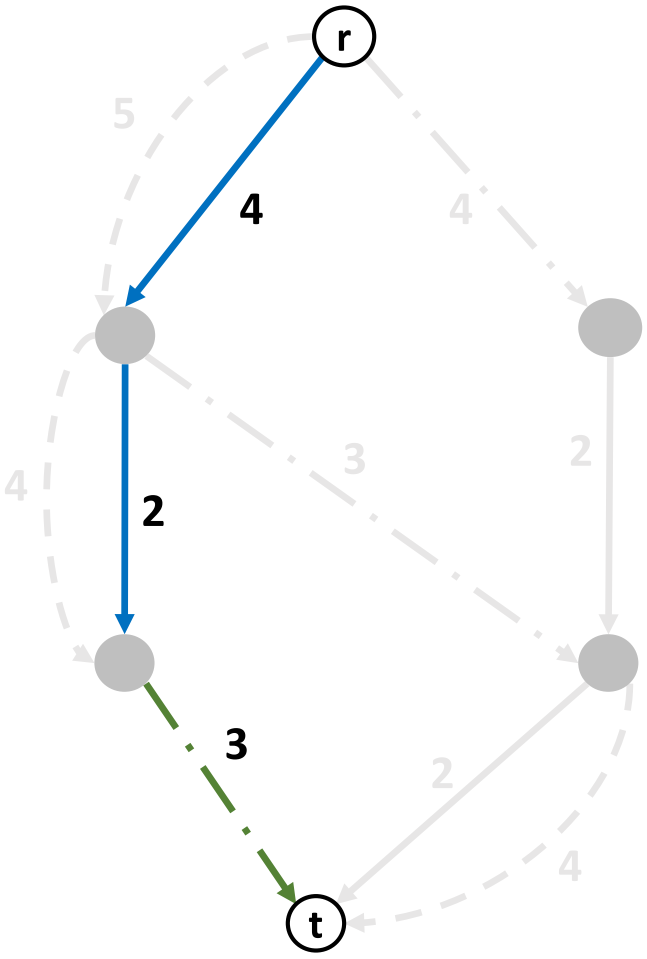
First task

Second task

Third task



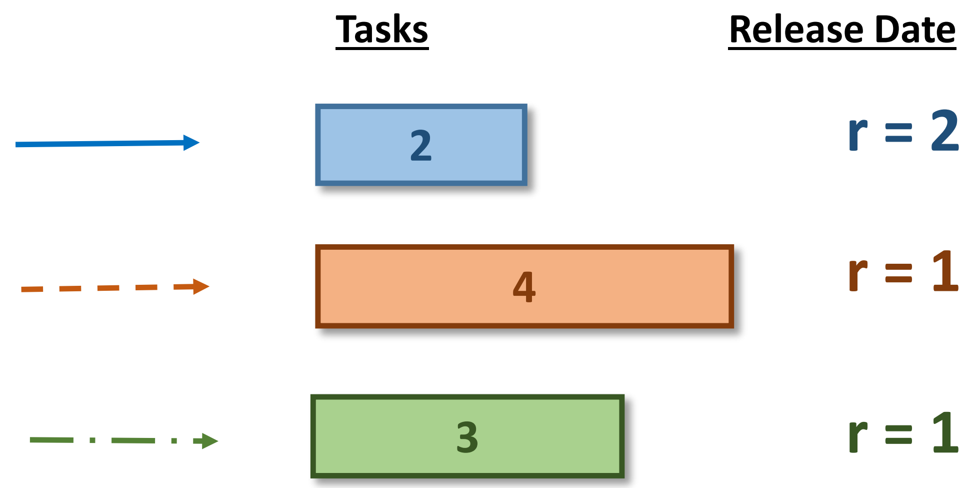
Relaxed Decision Diagram

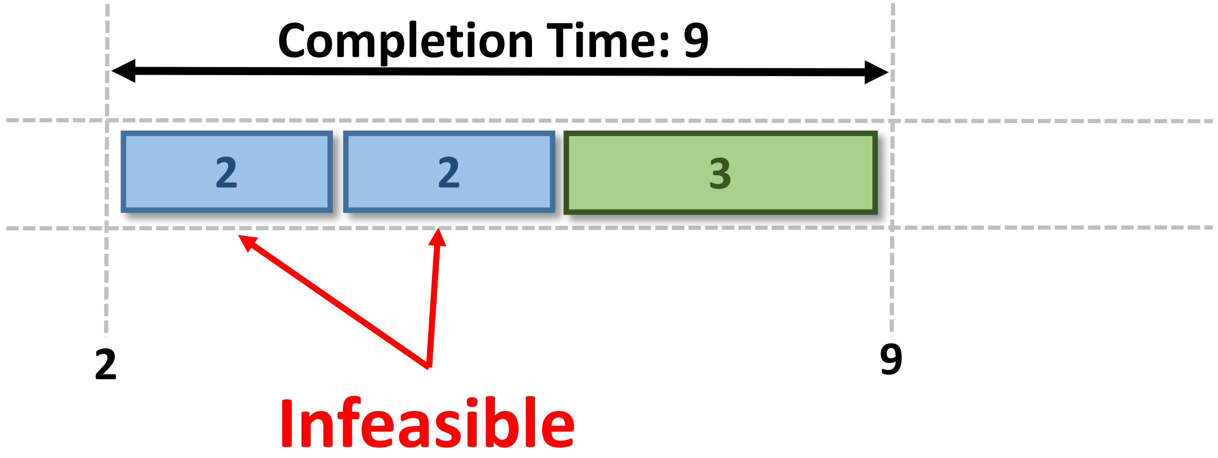


First task

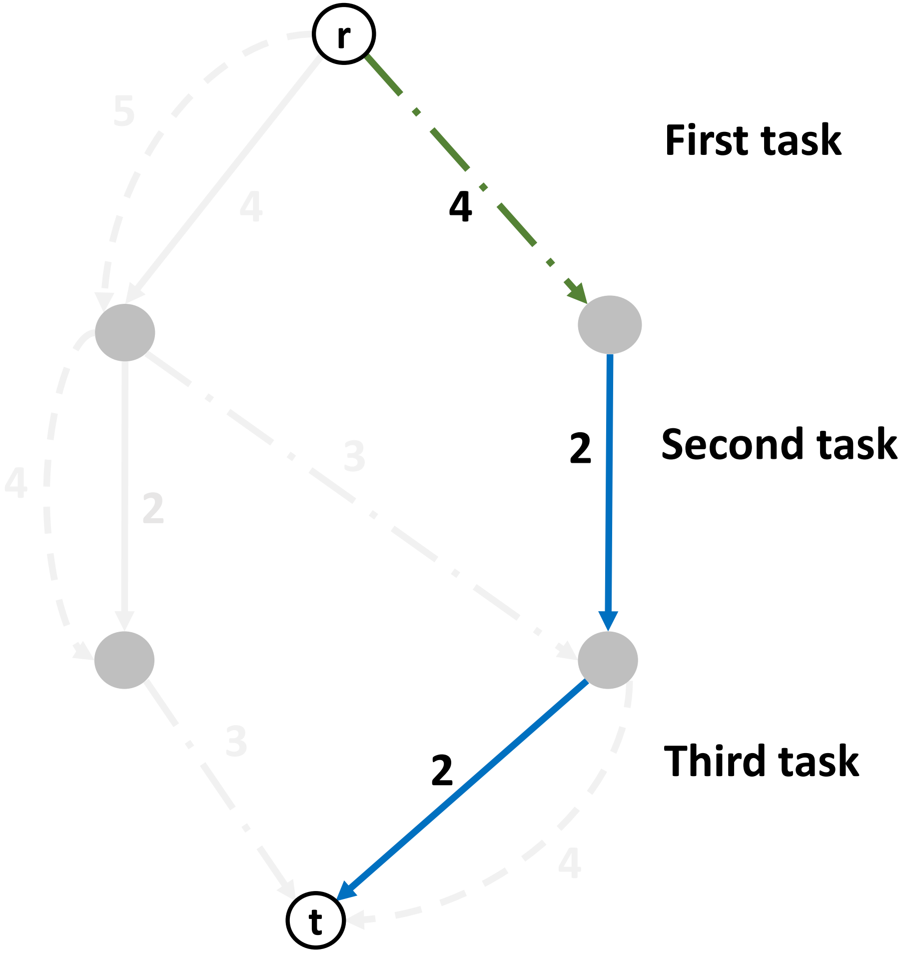
Second task

Third task



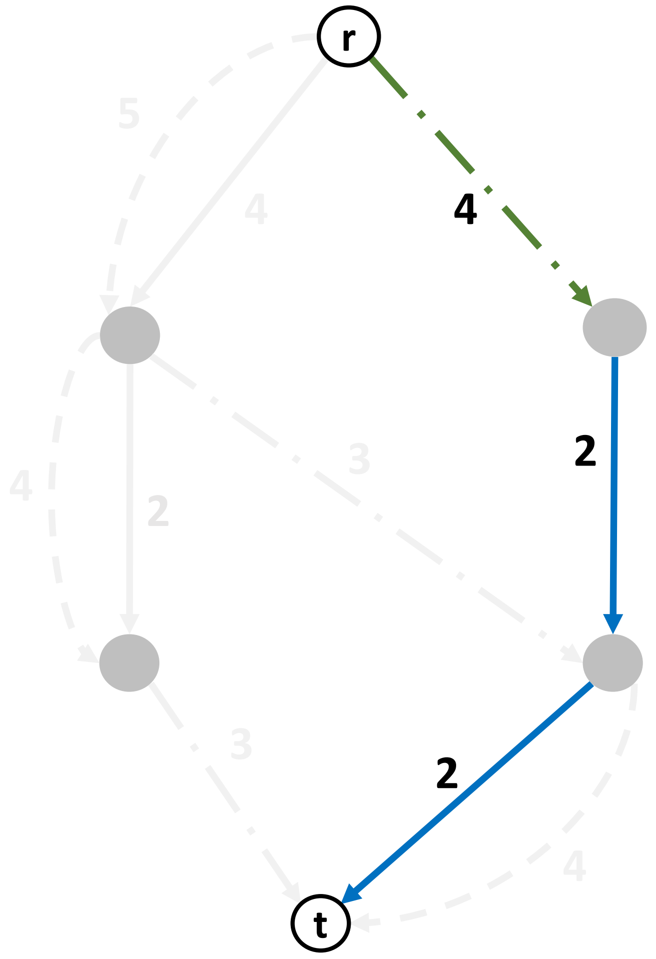


Relaxed Decision Diagram



Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

Relaxed Decision Diagram



First task

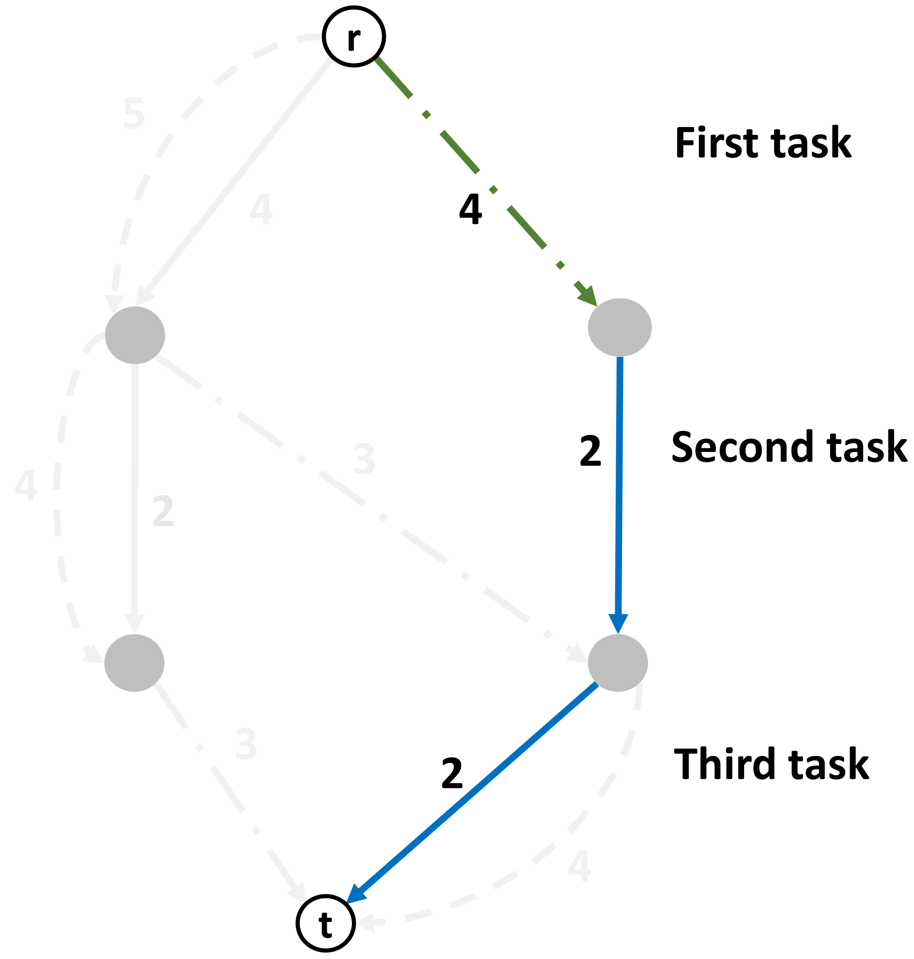
Second task

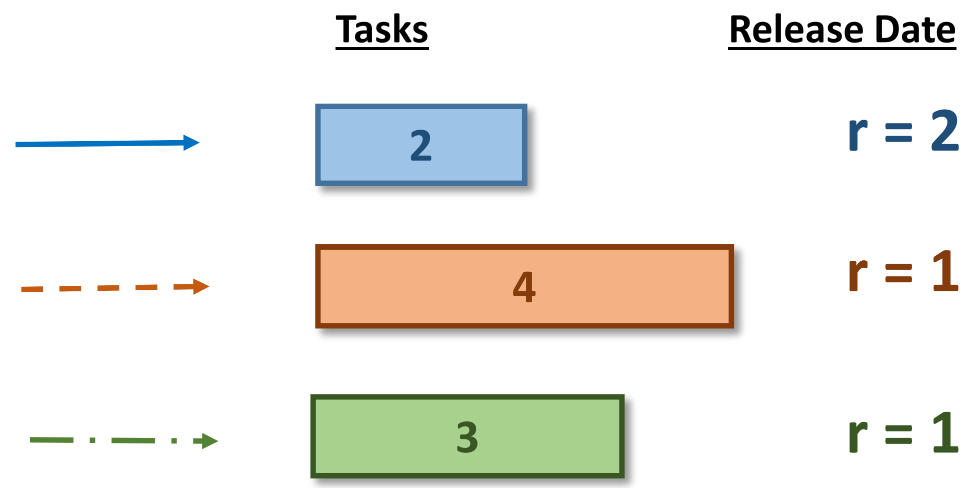
Third task

Tasks	Release Date
2	r = 2
4	r = 1
3	r = 1

- **Shortest path**

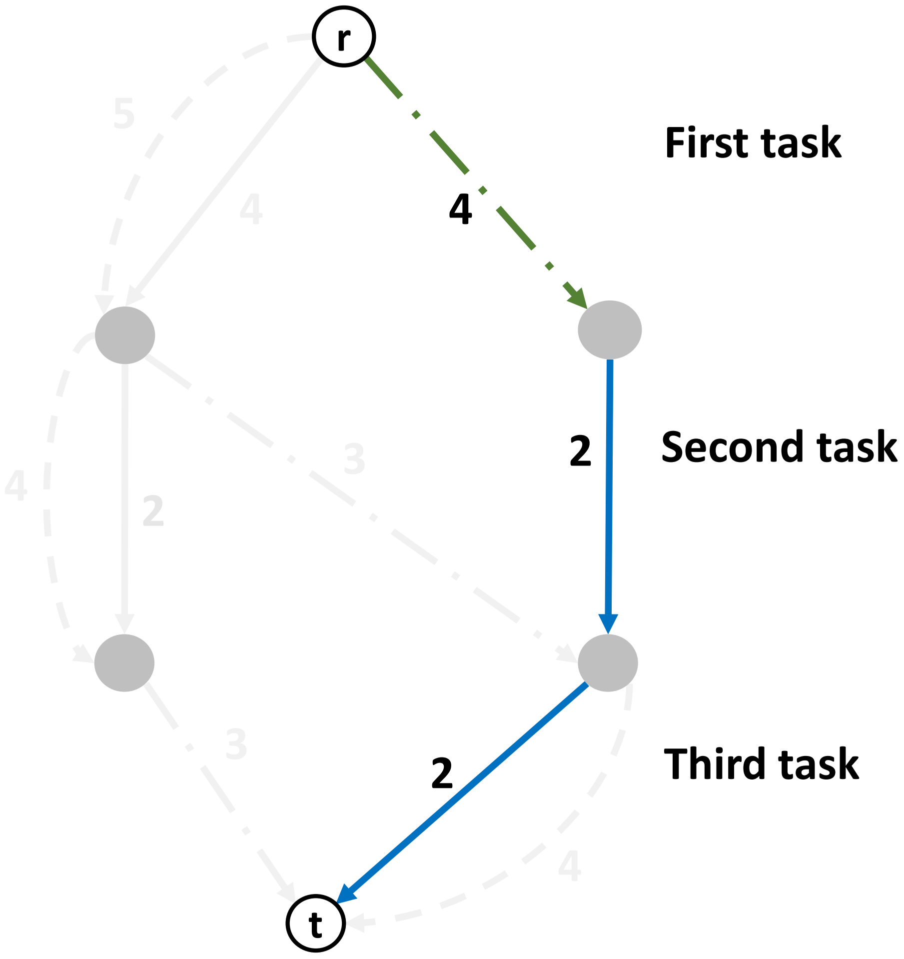
Relaxed Decision Diagram

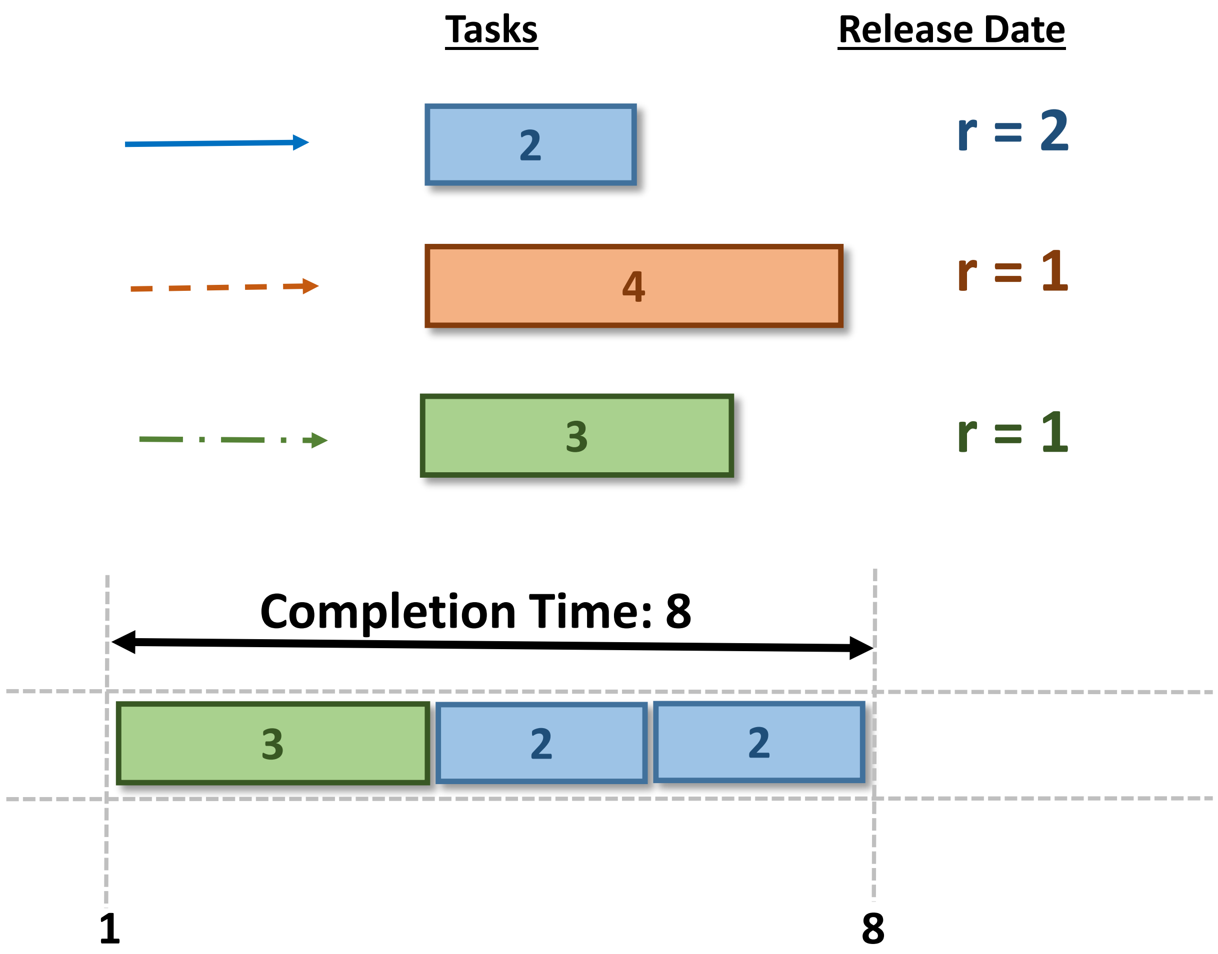




- **Shortest path**
  - Length: **Lower bound** on the optimal solution value

Relaxed Decision Diagram

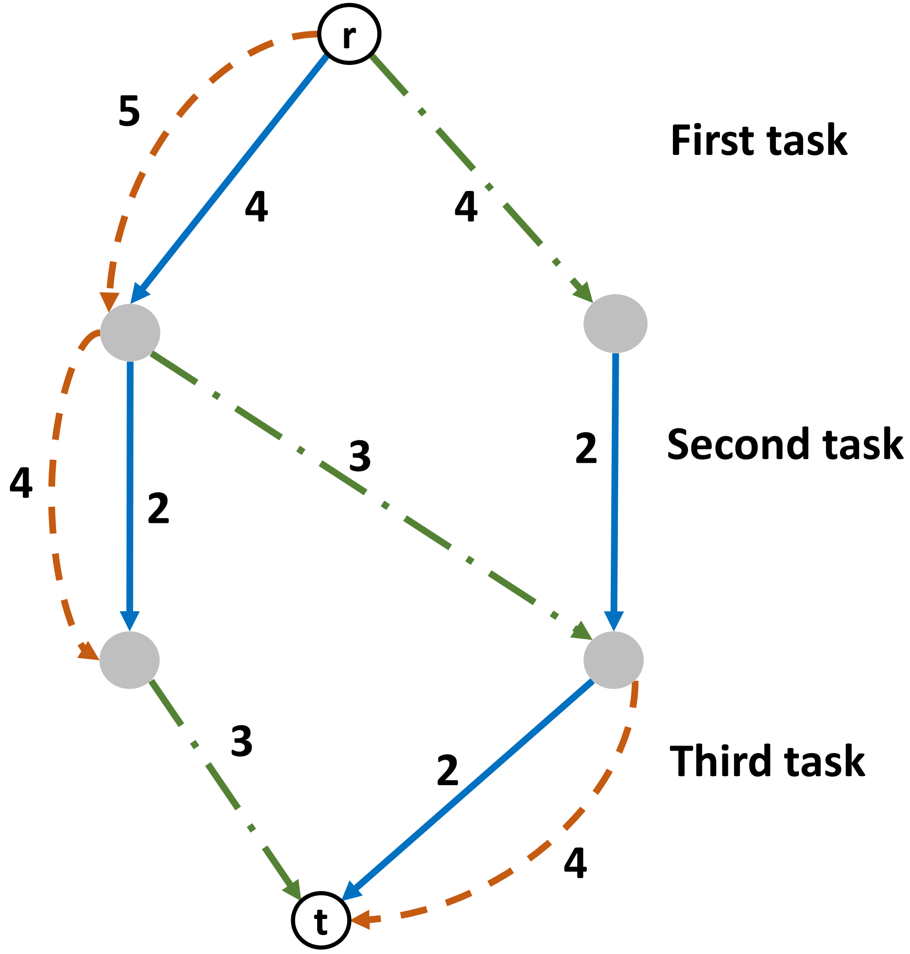




Uses of Relaxed Decision Diagrams

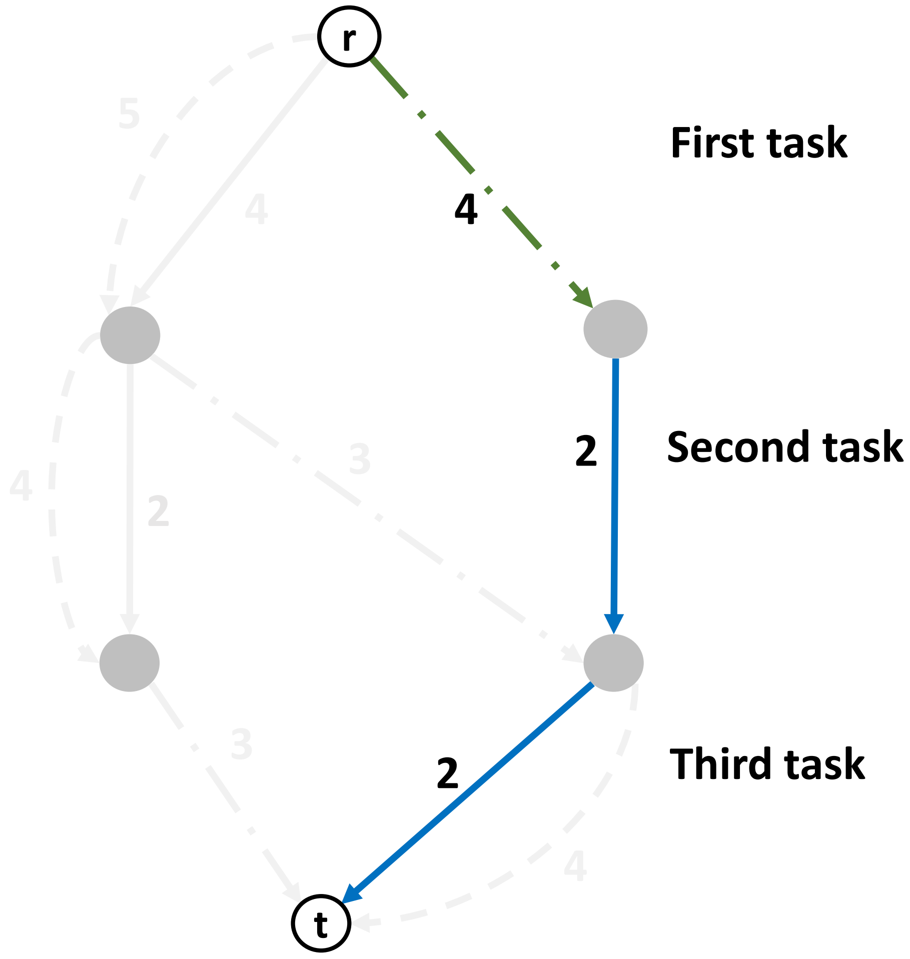
- Obtain **optimization bounds** (as LP relaxations do)
- As an **inference mechanism**
  - For example, as **global constraints** in CP *(MDD-based CP)*
- To guide **search** (in new branch-and-bound methods)

Issues



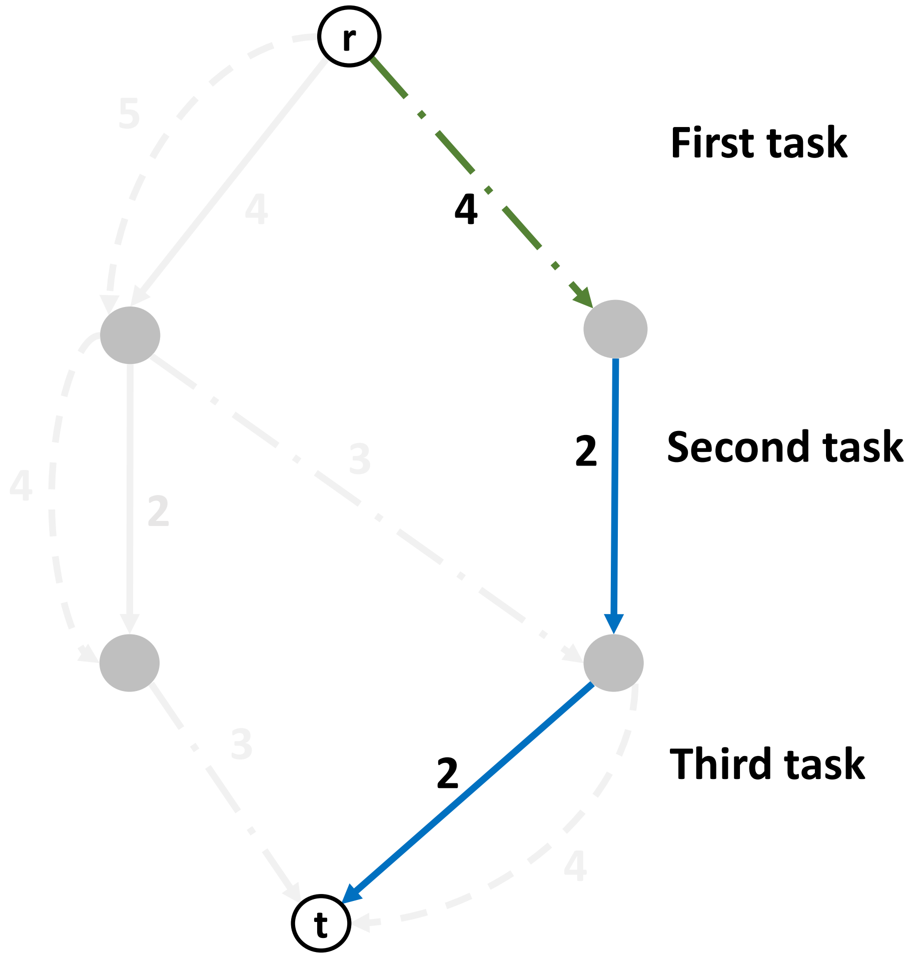
- Solutions of a relaxed DD may violate several constraints of the problem

Issues



- Solutions of a relaxed DD may violate several constraints of the problem

Issues



- Solutions of a relaxed DD may violate several constraints of the problem

- **Violation**: "All tasks performed once"

$$\sum_{e|v(e)=i} x_e = 1 \text{ for all tasks i}$$

Remedy: Lagrangian Relaxation

[Bergman et al., 2015]

min $\quad z = \text{shortest path}$

s.t. $\quad \sum_{e|v(e)=i} x_e = 1, \;\; \text{for all tasks } i$

(+other problem constraints)

Remedy: Lagrangian Relaxation

[Bergman et al., 2015]

min $\quad z$ = shortest path

s.t. $\quad \sum_{e|v(e)=i} x_e = 1$, for all tasks i $\longrightarrow$ Lagrangian multipliers λ_i

(+other problem constraints)

Remedy: Lagrangian Relaxation

[Bergman et al., 2015]

min $\quad z = \text{shortest path}$

s.t. $\quad \sum_{e|v(e)=i} x_e = 1,\ \text{for all tasks } i \longrightarrow$ Lagrangian multipliers λ_i

(+other problem constraints)

min $\quad z = \text{shortest path} + \sum_i \lambda_i \left(1 - \sum_{e|v(e)=i} x_e\right)$

s.t. $\quad$ (other problem constraints)

Remedy: Lagrangian Relaxation

[Bergman et al., 2015]

min $\quad z = \text{shortest path}$

s.t. $\quad \sum_{e|v(e)=i} x_e = 1$, for all tasks i → Lagrangian multipliers λ_i

(+other problem constraints)

min $\quad z = \text{shortest path} + \sum_i \lambda_i (1 - \sum_{e|v(e)=i} x_e)$

s.t. $\quad$ (other problem constraints)

This is done by updating shortest path weights!

General Approach

- We penalize infeasible solutions in a relaxed DD:

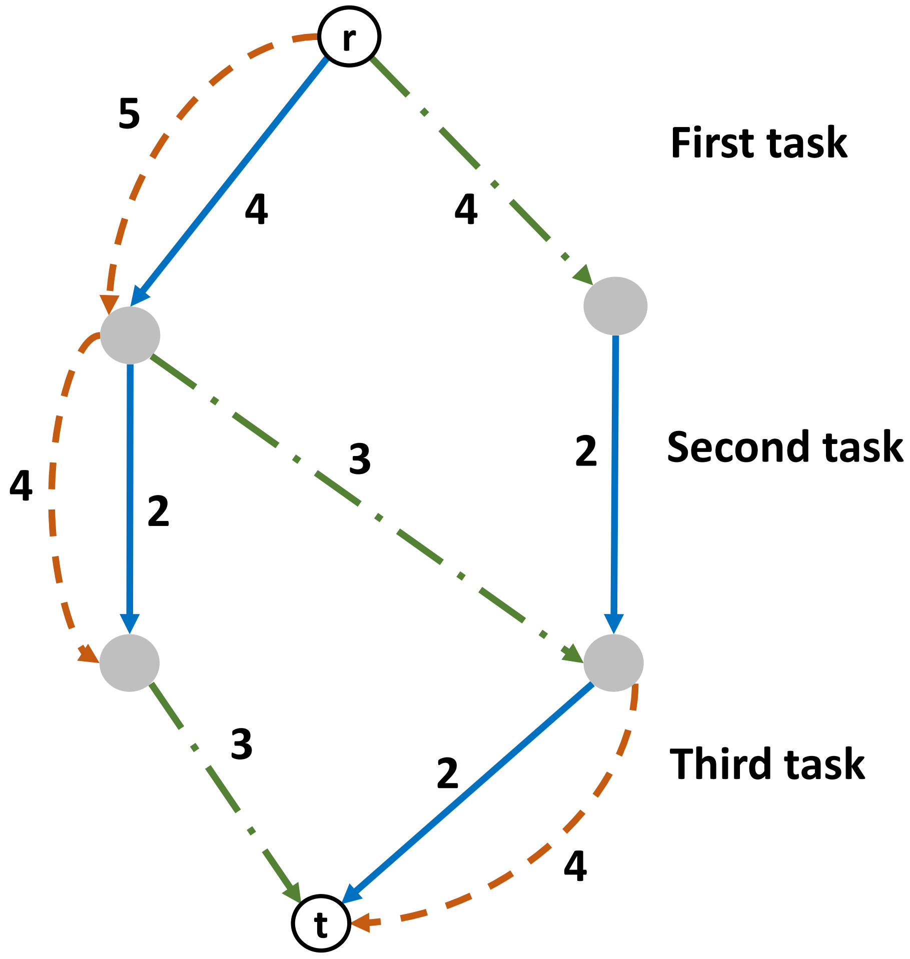
  Any separable constraint of the form

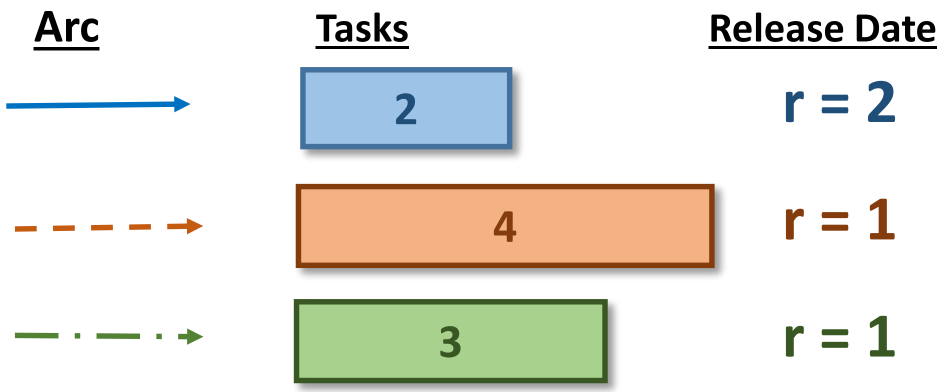
  $$f_1(x_1) + f_2(x_2) + \ldots + f_n(x_n) \leq c$$

  that **must** be satisfied by solutions of an MDD can be dualized

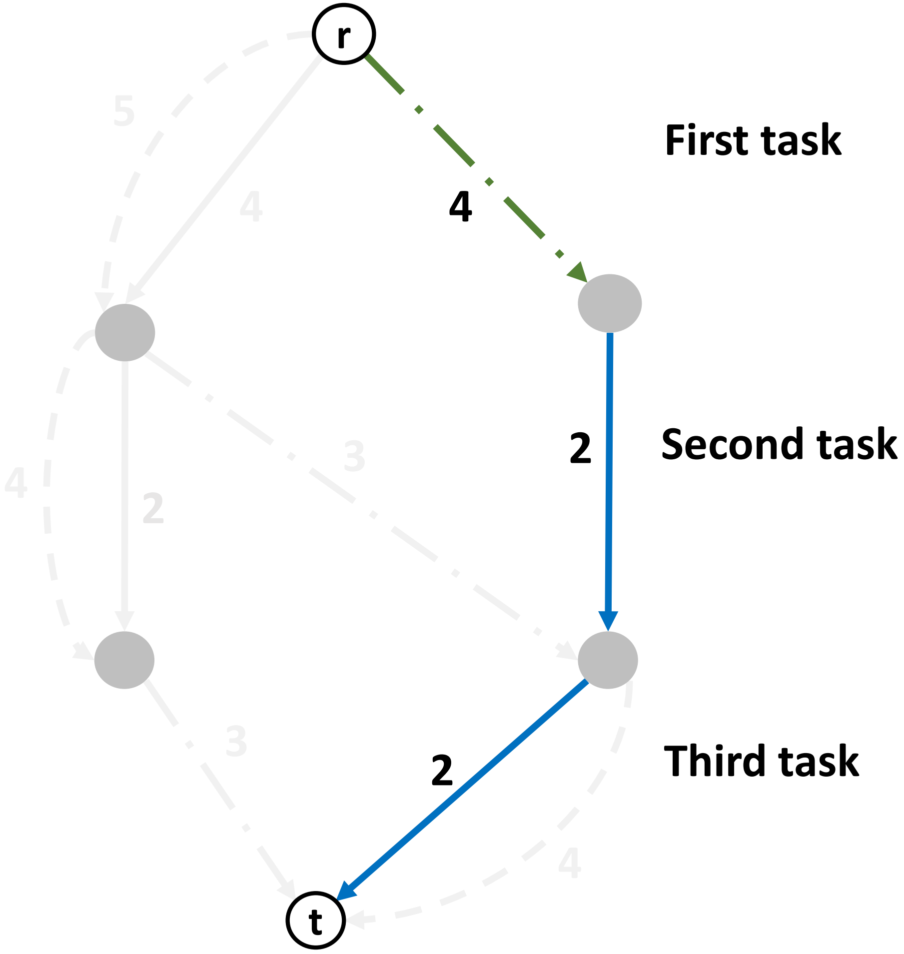
- We need only to focus on the **shortest path solution**
  - Identify a violated constraint and penalize
  - Systematic way directly adapted from LP
  - Shortest paths are **very fast** to compute

Improving Relaxed Decision Diagram



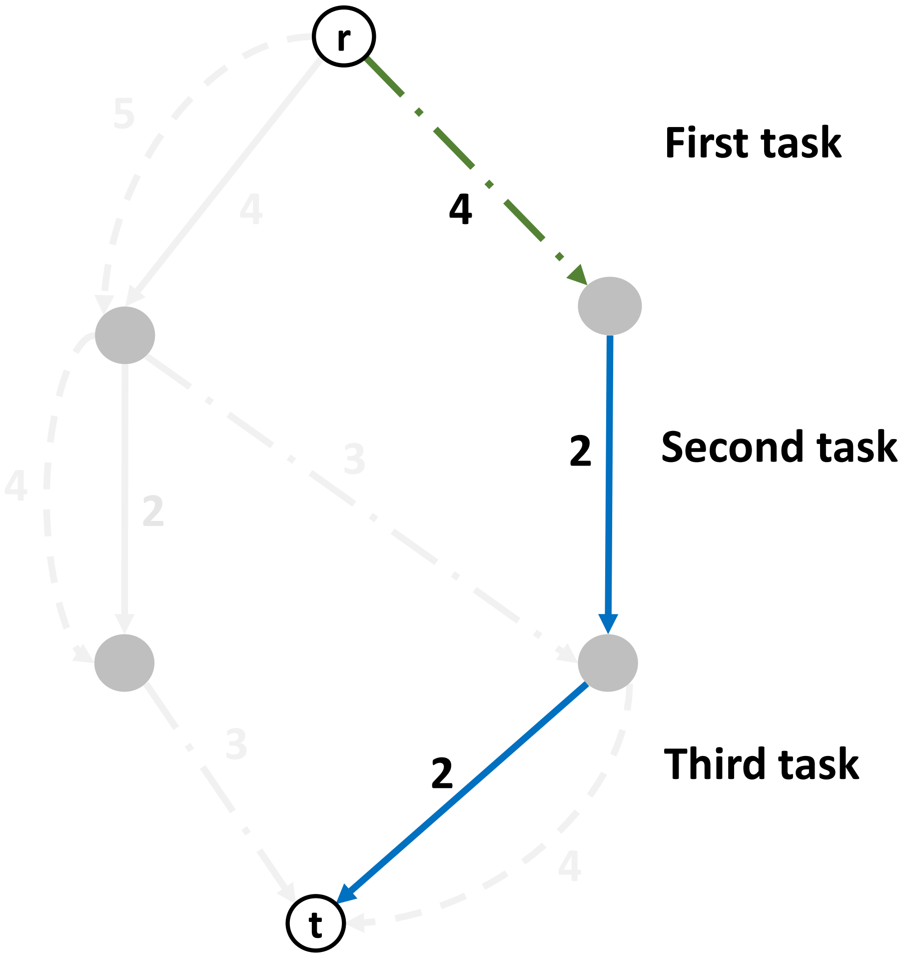


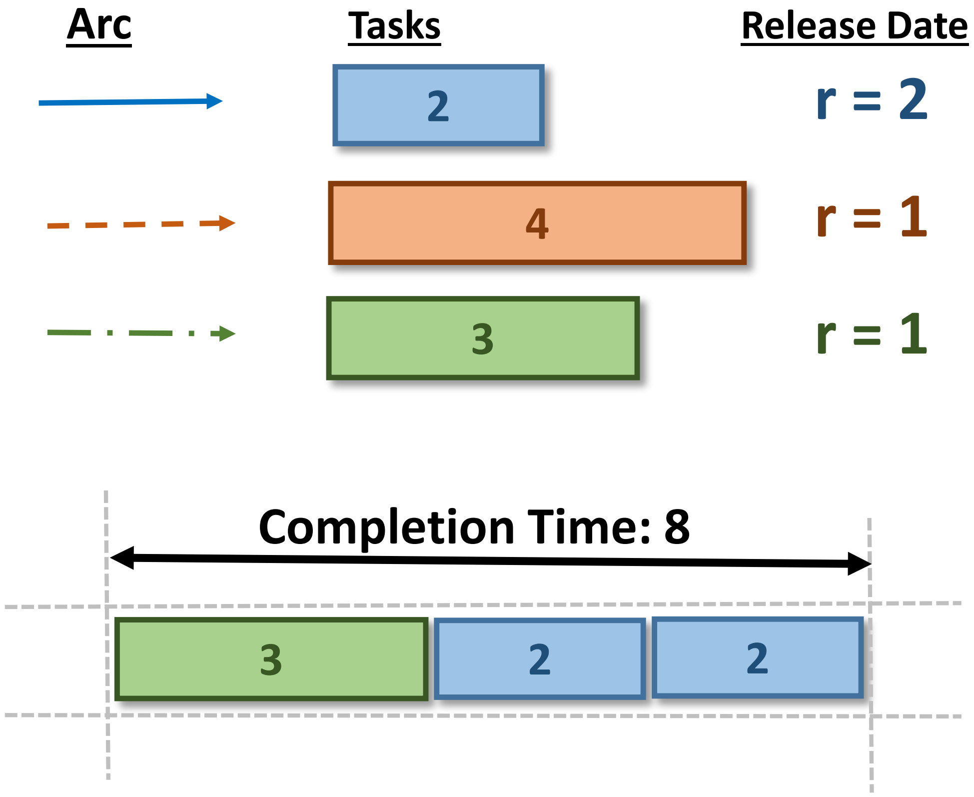
Improving Relaxed Decision Diagram



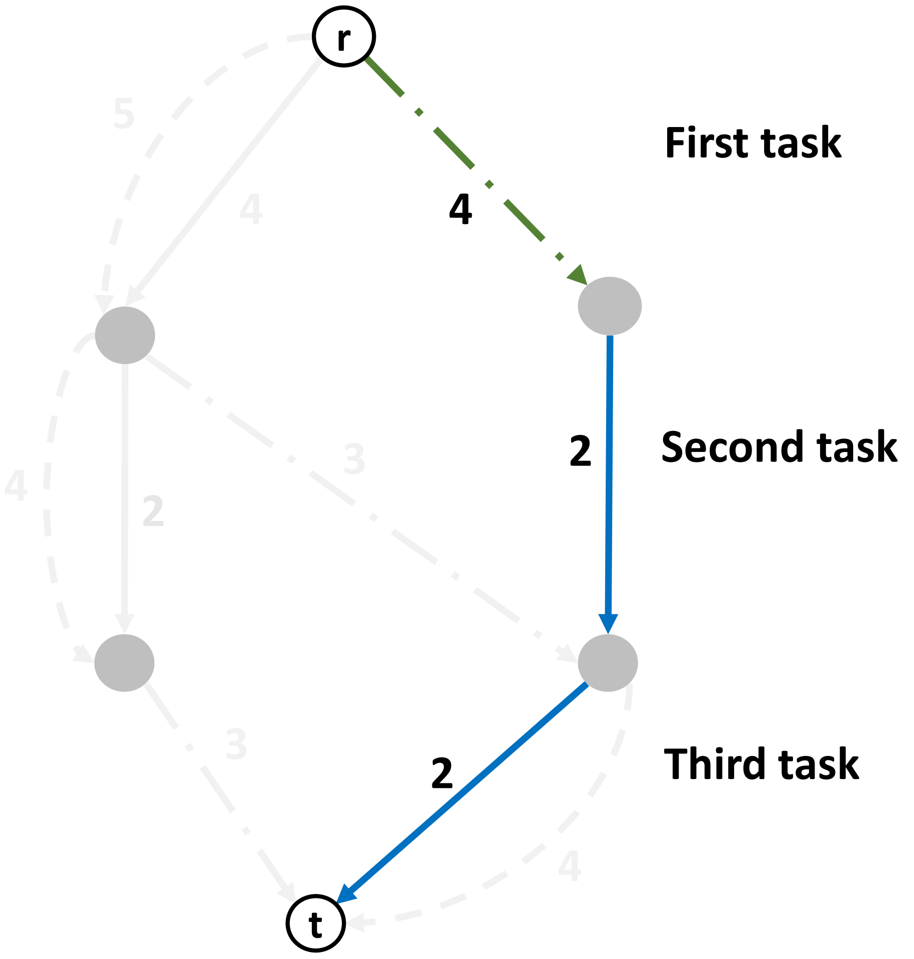
Arc	Tasks	Release Date
	2	r = 2
	4	r = 1
	3	r = 1

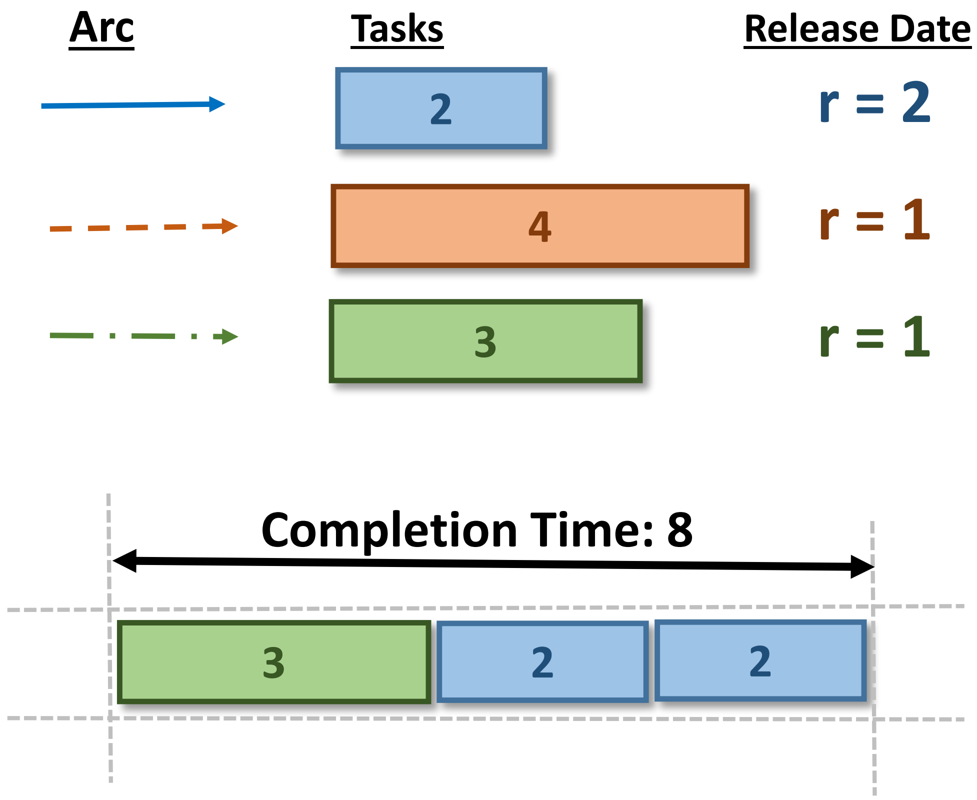
Improving Relaxed Decision Diagram





Improving Relaxed Decision Diagram

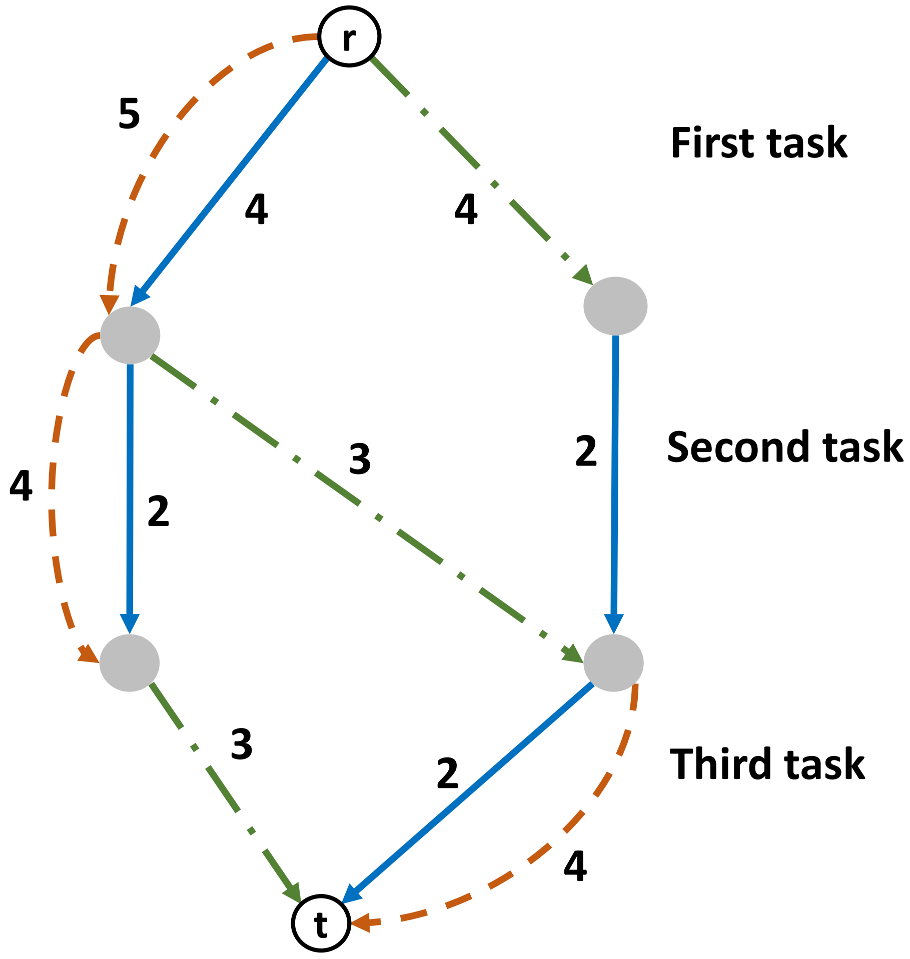


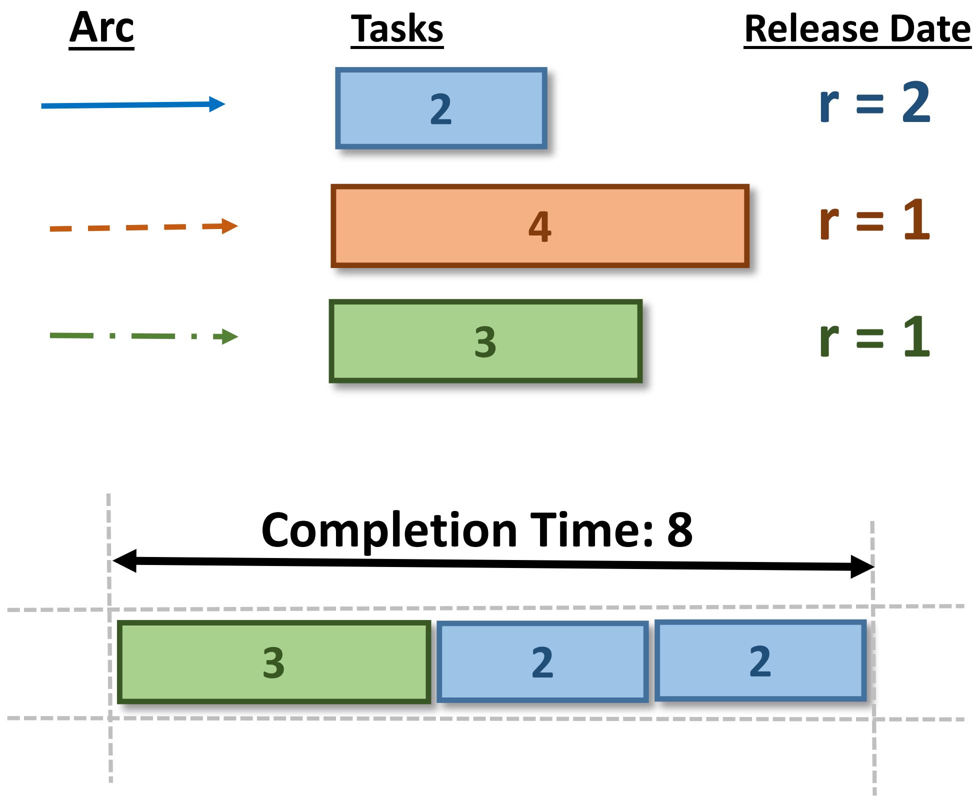


Penalization:

- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

Improving Relaxed Decision Diagram

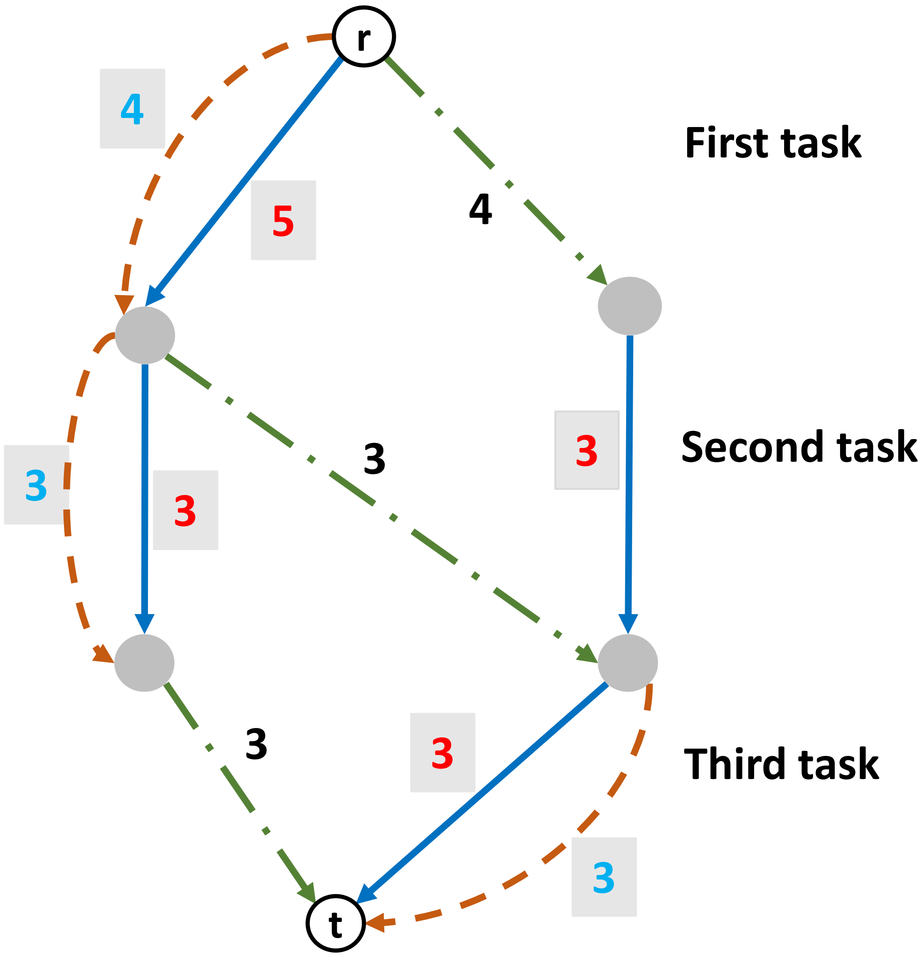


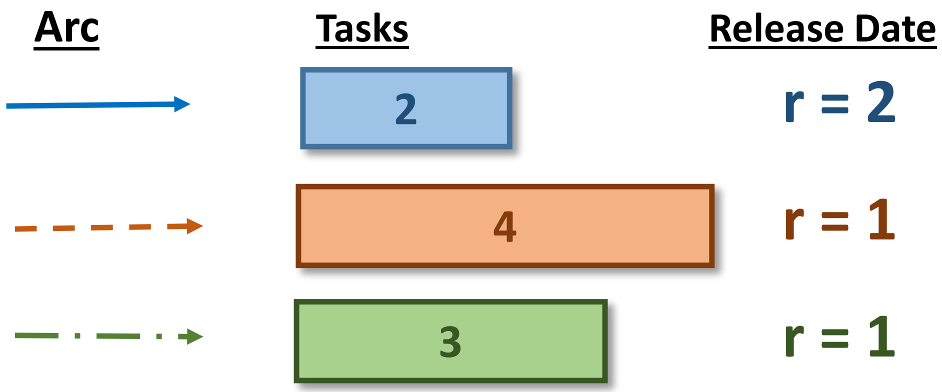


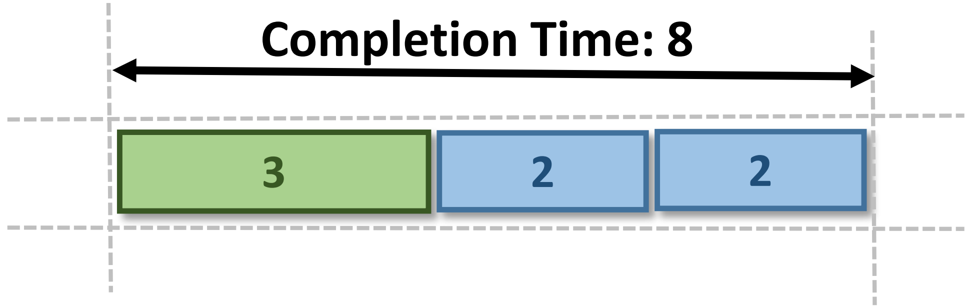
Penalization:

- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

Improving Relaxed Decision Diagram



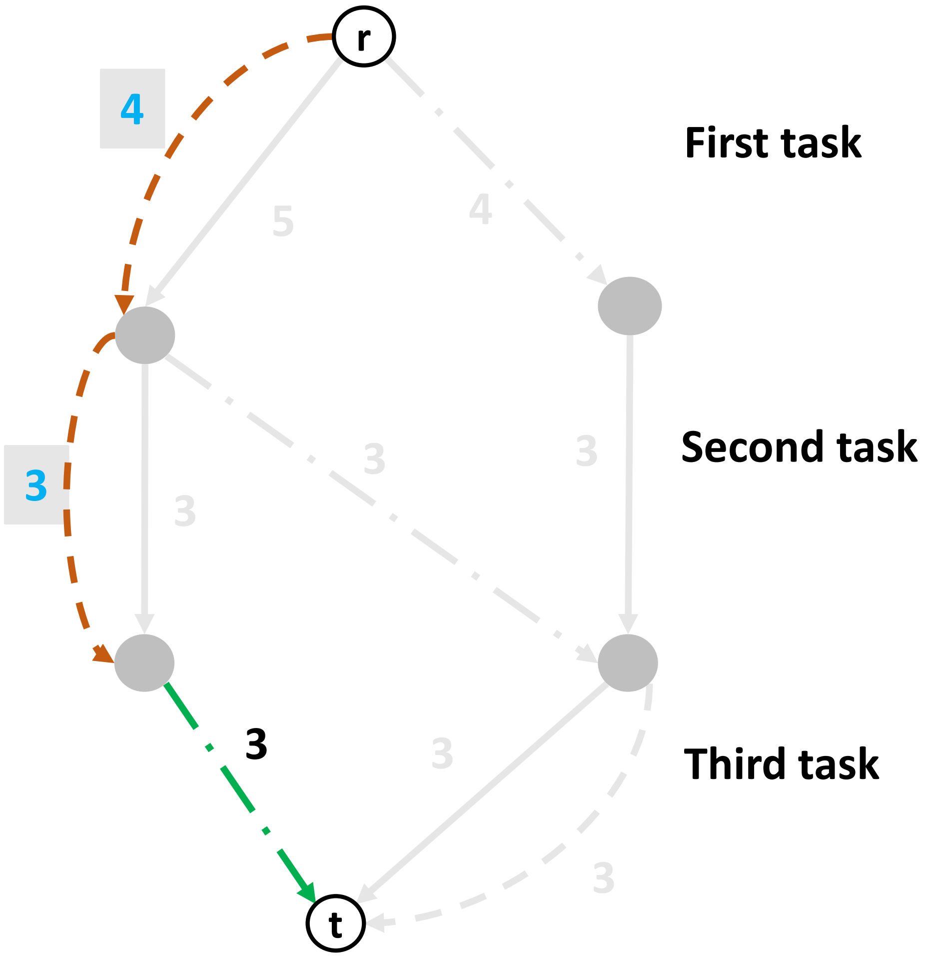


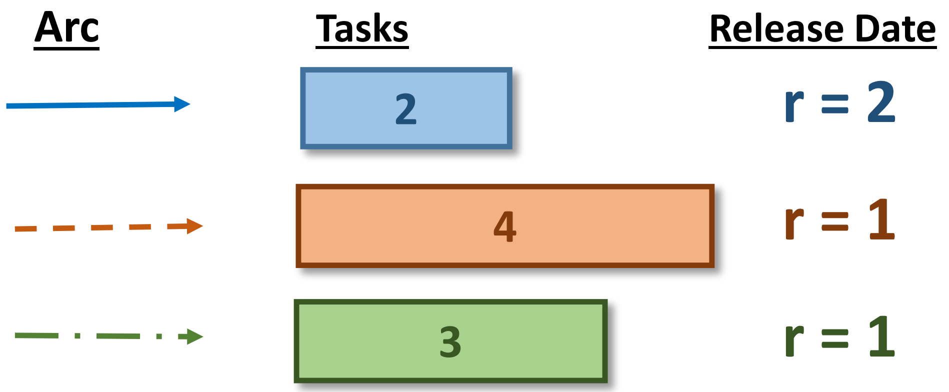


Penalization:

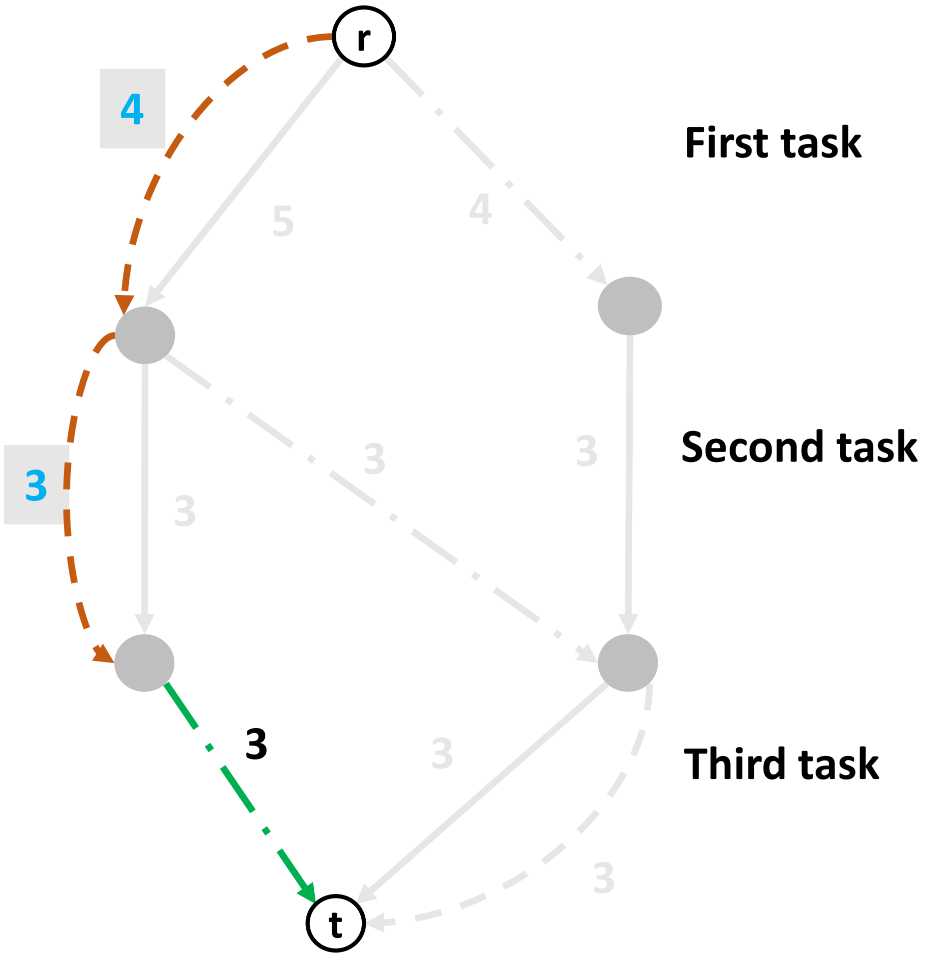
- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

Improving Relaxed Decision Diagram





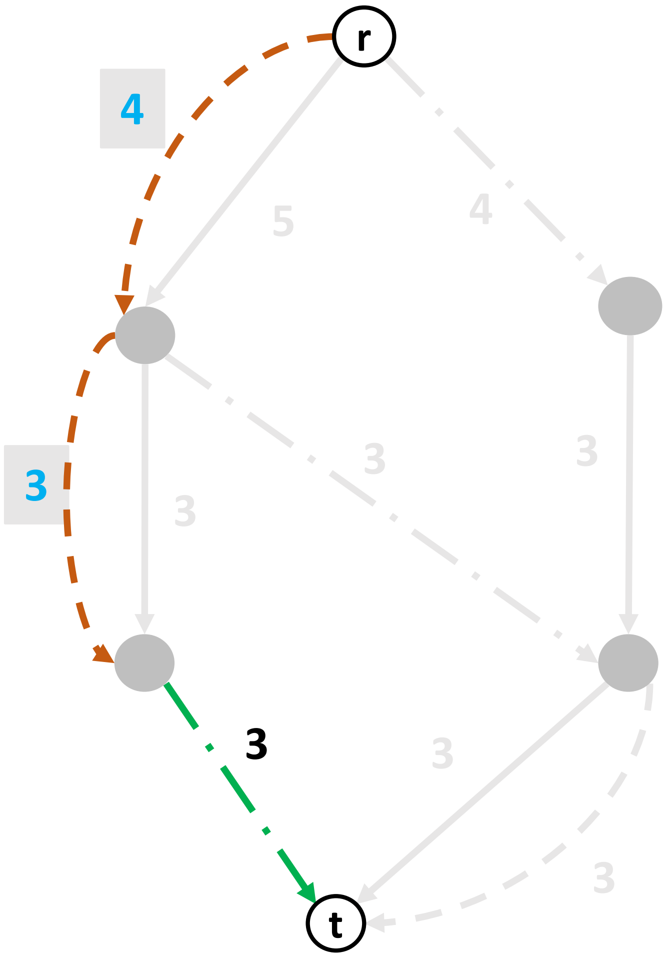
Improving Relaxed Decision Diagram



Arc	Tasks	Release Date
blue solid	2	r = 2
orange dashed	4	r = 1
green dash-dot	3	r = 1

- **New shortest path: 10**

Improving Relaxed Decision Diagram



First task

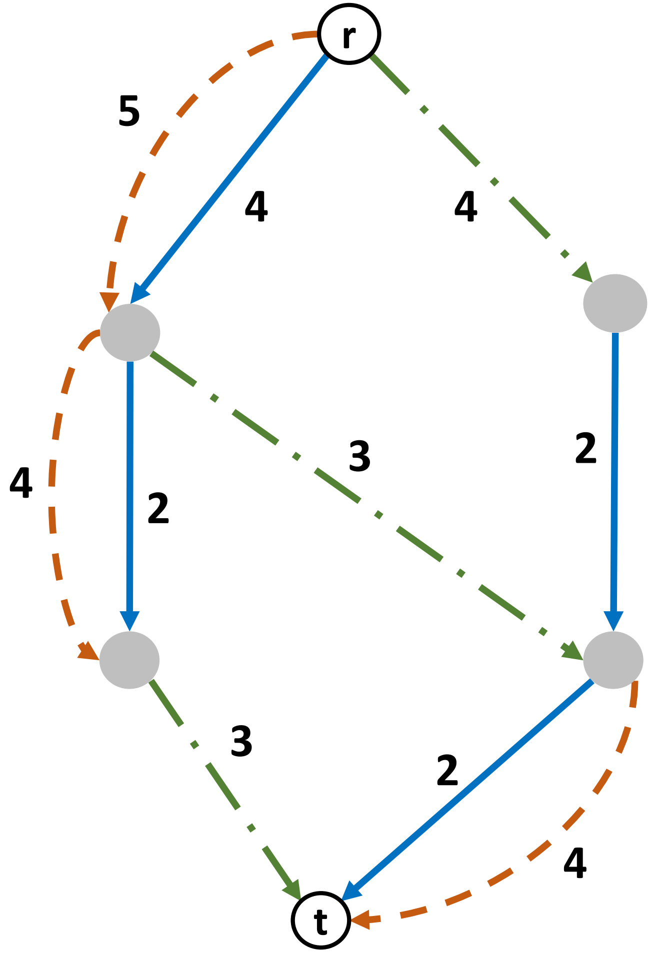
Second task

Third task

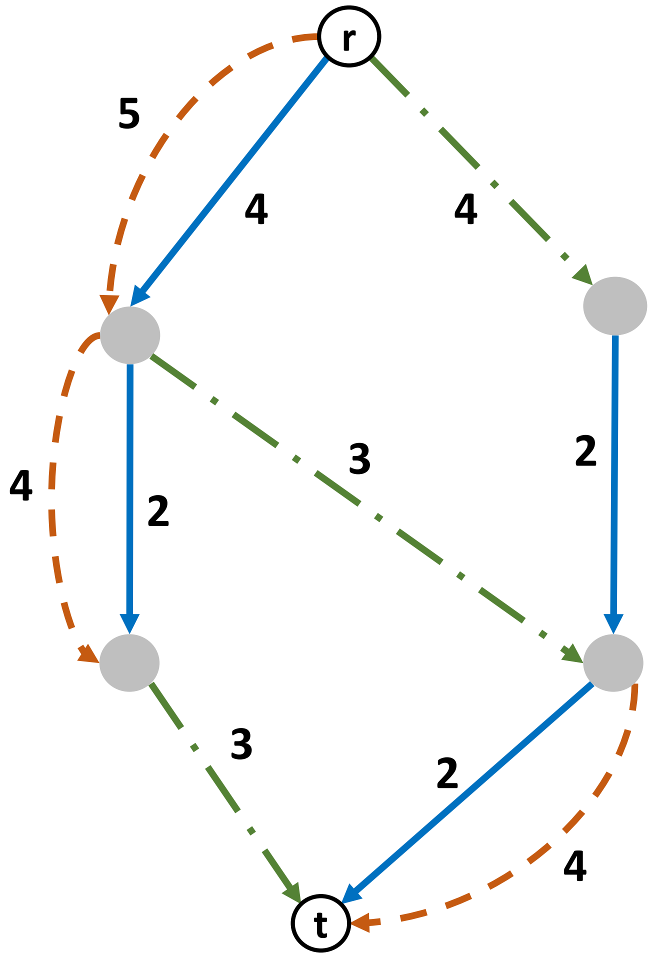
Arc	Tasks	Release Date
	2	r = 2
	4	r = 1
	3	r = 1

- **New shortest path: 10**
  - Guaranteed to be **a valid lower bound** for any penalties

Additional Filtering

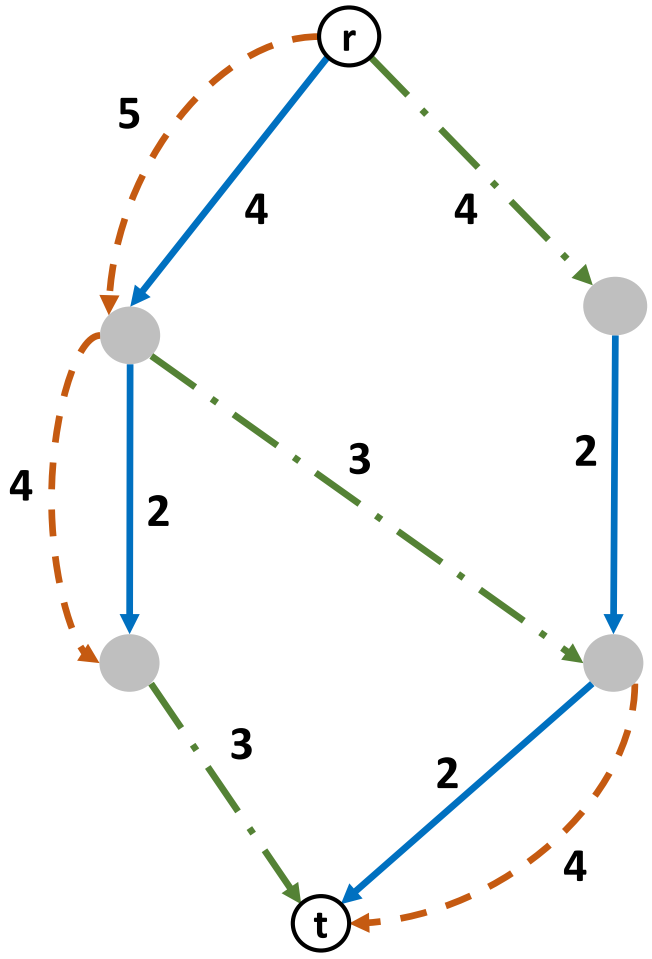


Additional Filtering



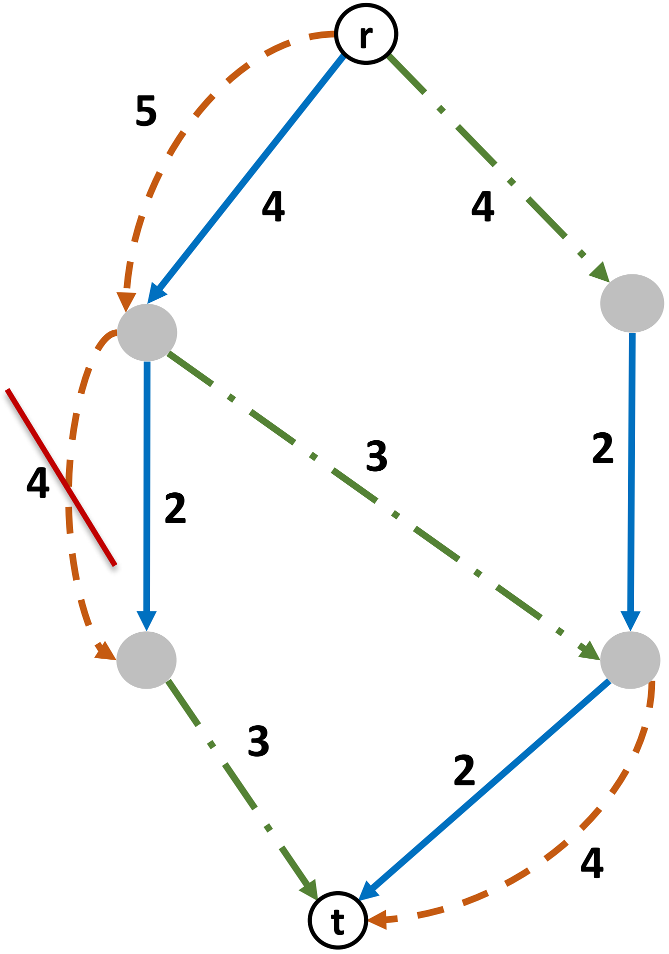
- If minimum solution value through an arc exceeds max(D(z)) then arc can be deleted

Additional Filtering



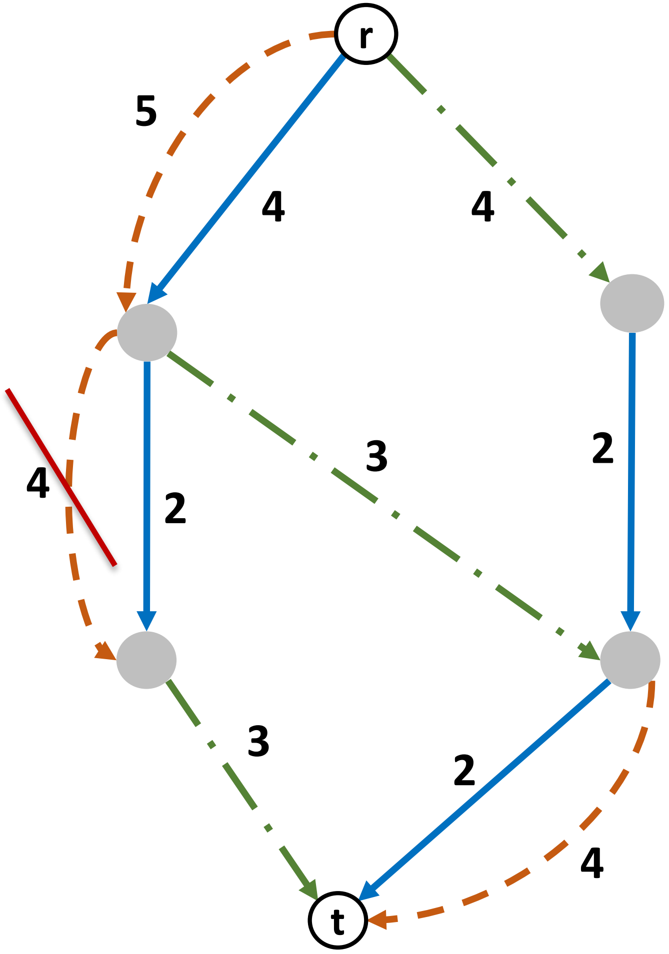
- If minimum solution value through an arc exceeds max(D(z)) then arc can be deleted
- Suppose a solution of value 10 is known

Additional Filtering



- If minimum solution value through an arc exceeds max(D(z)) then arc can be deleted
- Suppose a solution of value 10 is known

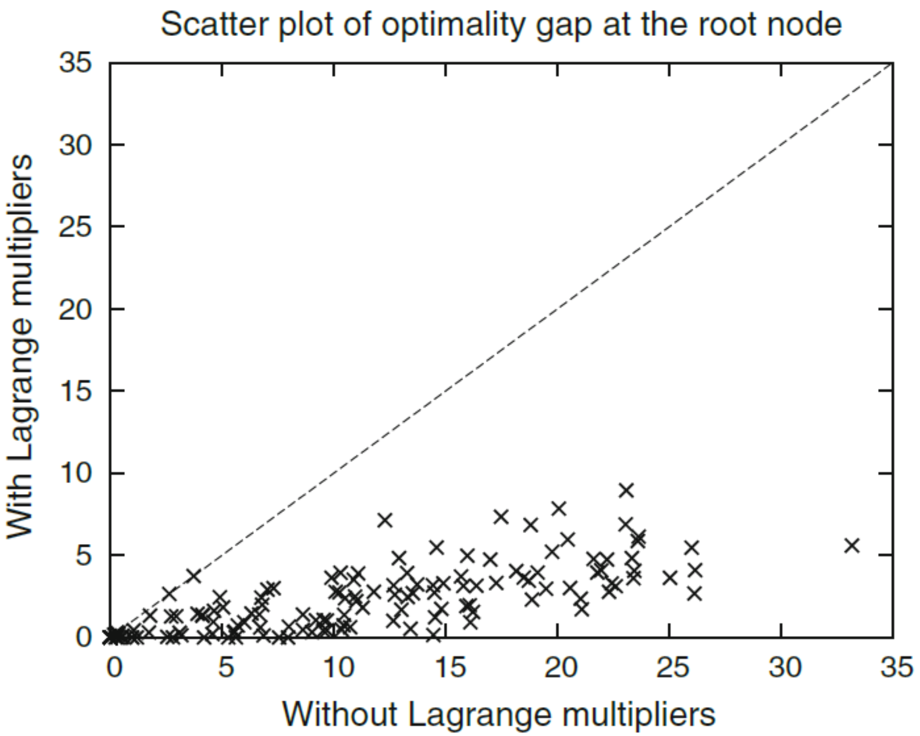
Additional Filtering

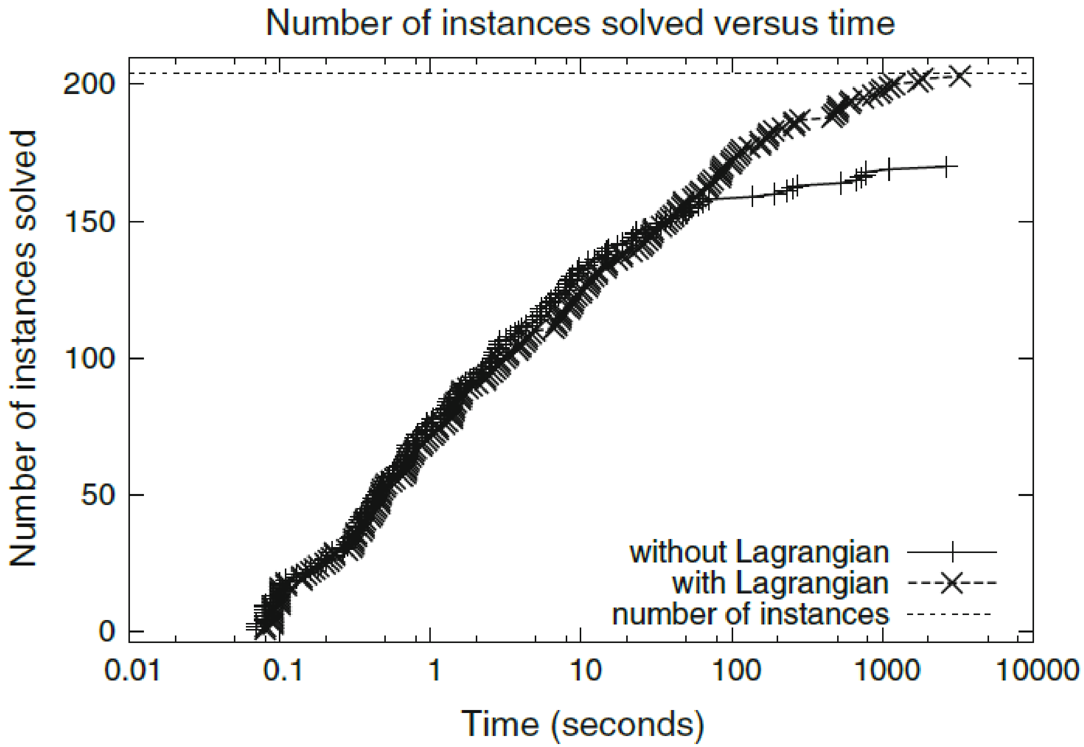


- If minimum solution value through an arc exceeds max(D(z)) then arc can be deleted
- Suppose a solution of value 10 is known
- MDD filtering extends to Lagrangian weights: More filtering possible

Impact on TSP with Time Windows

Impact on TSP with Time Windows





TSPTW instances

(Dumas and GendreauDumasExtended)

(*Constraints*, 2015)

Beyond Single Optimization Constraints

- Constraint propagation is CP's major strength
  - Rich modeling interface, fast domain filtering, "combinatorial programming"
- …but also its major weakness (when using domains)
  - Cartesian product of variable domains is weak relaxation,
  - Conventional domain propagation has limited communication power
  - Propagating relaxed MDDs can help, but not in all cases

Beyond Single Optimization Constraints

- Constraint propagation is CP's major strength
  - Rich modeling interface, fast domain filtering, "combinatorial programming"
- …but also its major weakness (when using domains)
  - Cartesian product of variable domains is weak relaxation,
  - Conventional domain propagation has limited communication power
  - Propagating relaxed MDDs can help, but not in all cases
- Up next: Lagrangian Propagation instead of domain propagation
  - Via Lagrangian decomposition [Bergman et al. CP2015], [Ha et al., CP2015]

Lagrangian Decomposition

$$(P) \qquad \max\{fx \mid Ax \le b, Cx \le d, x \in X\}$$

[Guignard & Kim, 1987]

Lagrangian Decomposition

$$(P) \quad \max\{fx \mid Ax \le b, Cx \le d, x \in X\}$$

$$= \max\{fx \mid Ay \le b, Cx \le d, x = y, x \in X, y \in Y\}$$

[Guignard & Kim, 1987]

Lagrangian Decomposition

$$(P) \qquad \max\{fx \mid Ax \le b, Cx \le d, x \in X\}$$

$$= \max\{fx \mid Ay \le b, Cx \le d, \boxed{x = y,} x \in X, y \in Y\}$$

[Guignard & Kim, 1987]

Lagrangian Decomposition

$$(P) \qquad \max\{fx \mid Ax \le b, Cx \le d, x \in X\}$$

$$= \max\{fx \mid Ay \le b, Cx \le d, \boxed{x = y,} x \in X, y \in Y\}$$

$$L_P(\lambda) := \max\{fx + \lambda(y - x) \mid Cx \le d, x \in X, Ay \le b, y \in Y\}$$

[Guignard & Kim, 1987]

Lagrangian Decomposition

$$(P) \qquad \max\{fx \mid Ax \le b, Cx \le d, x \in X\}$$

$$= \max\{fx \mid Ay \le b, Cx \le d, \boxed{x = y,} x \in X, y \in Y\}$$

$$L_P(\lambda) := \max\{fx + \lambda(y - x) \mid Cx \le d, x \in X, Ay \le b, y \in Y\}$$

$$= \max\{(f - \lambda)x \mid Cx \le d, x \in X\} + \max\{\lambda y \mid Ay \le by \in Y\}$$

[Guignard & Kim, 1987]

Lagrangian Decomposition

$$(P) \quad \max\{fx \mid Ax \le b, Cx \le d, x \in X\}$$

$$= \max\{fx \mid Ay \le b, Cx \le d, \boxed{x = y,} x \in X, y \in Y\}$$

$$L_P(\lambda) := \max\{fx + \lambda(y - x) \mid Cx \le d, x \in X, Ay \le b, y \in Y\}$$

$$= \max\{(f - \lambda)x \mid Cx \le d, x \in X\} + \max\{\lambda y \mid Ay \le by \in Y\}$$

- Bound from Lagrangian Decomposition at least as strong as Lagrangian relaxation from either dualizing Ax ≤ b or Cx ≤ d

[Guignard & Kim, 1987]

Motivating CP Example

$$\texttt{alldiff}(x_1, x_2, x_3)$$

$$\texttt{alldiff}(x_2, x_4, x_5)$$

$$\texttt{alldiff}(x_3, x_5)$$

$$x_1 \in \{a, b\}$$

$$x_2 \in \{b, c\}$$

$$x_3 \in \{a, c\}$$

$$x_4 \in \{a, b\}$$

$$x_5 \in \{a, b, c\}$$

Motivating CP Example

$\texttt{alldiff}(x_1, x_2, x_3)$

$\texttt{alldiff}(x_2, x_4, x_5)$

$\texttt{alldiff}(x_3, x_5)$

$x_1 \in \{a, b\}$

$x_2 \in \{b, c\}$

$x_3 \in \{a, c\}$

$x_4 \in \{a, b\}$

$x_5 \in \{a, b, c\}$

Motivating CP Example

$$\texttt{alldiff}(x_1, x_2, x_3)$$
$$\texttt{alldiff}(x_2, x_4, x_5)$$
$$\texttt{alldiff}(x_3, x_5)$$
$$x_1 \in \{a, b\}$$
$$x_2 \in \{b, c\}$$
$$x_3 \in \{a, c\}$$
$$x_4 \in \{a, b\}$$
$$x_5 \in \{a, b, c\}$$

- Domain consistent; even pairwise consistent
- Constraint propagation has no effect here…

Motivating CP Example

$$\texttt{alldiff}(x_1, x_2, x_3)$$
$$\texttt{alldiff}(x_2, x_4, x_5)$$
$$\texttt{alldiff}(x_3, x_5)$$
$$x_1 \in \{a, b\}$$
$$x_2 \in \{b, c\}$$
$$x_3 \in \{a, c\}$$
$$x_4 \in \{a, b\}$$
$$x_5 \in \{a, b, c\}$$

$$\texttt{alldiff}(x_1, x_2, x_3)$$
$$\texttt{alldiff}(y_2, y_4, y_5)$$
$$\texttt{alldiff}(z_3, z_5)$$
$$x_1 \in \{a, b\}$$
$$x_2, y_2 \in \{b, c\} \qquad x_2 = y_2$$
$$x_3, z_3 \in \{a, c\} \qquad x_3 = z_3$$
$$y_4 \in \{a, b\}$$
$$y_5, z_5 \in \{a, b, c\} \qquad y_5 = z_5$$

- Domain consistent; even pairwise consistent
- Constraint propagation has no effect here…

Motivating CP Example

$$\texttt{alldiff}(x_1, x_2, x_3)$$
$$\texttt{alldiff}(x_2, x_4, x_5)$$
$$\texttt{alldiff}(x_3, x_5)$$
$$x_1 \in \{a, b\}$$
$$x_2 \in \{b, c\}$$
$$x_3 \in \{a, c\}$$
$$x_4 \in \{a, b\}$$
$$x_5 \in \{a, b, c\}$$

$$\texttt{alldiff}(x_1, x_2, x_3)$$
$$\texttt{alldiff}(y_2, y_4, y_5)$$
$$\texttt{alldiff}(z_3, z_5)$$
$$x_1 \in \{a, b\}$$
$$x_2, y_2 \in \{b, c\} \qquad x_2 = y_2$$
$$x_3, z_3 \in \{a, c\} \qquad x_3 = z_3$$
$$y_4 \in \{a, b\}$$
$$y_5, z_5 \in \{a, b, c\} \qquad y_5 = z_5$$

- Domain consistent; even pairwise consistent
- Constraint propagation has no effect here…

Lagrangian Decomposition for CP

$$
\begin{array}{ll}
\max & \overline{\lambda}_2(x_2 \neq y_2) + \overline{\lambda}_3(x_3 \neq z_3) + \overline{\lambda}_5(y_5 \neq z_5) \\
\text{s.t.} & \texttt{alldiff}(x_1, x_2, x_3) \\
& \texttt{alldiff}(y_2, y_4, y_5) \\
& \texttt{alldiff}(z_3, z_5)
\end{array}
$$

Lagrangian Decomposition for CP

$$
\begin{array}{ll}
\max & \overline{\lambda}_2(x_2 \neq y_2) + \overline{\lambda}_3(x_3 \neq z_3) + \overline{\lambda}_5(y_5 \neq z_5) \\
\text{s.t.} & \texttt{alldiff}(x_1, x_2, x_3) \\
& \texttt{alldiff}(y_2, y_4, y_5) \\
& \texttt{alldiff}(z_3, z_5)
\end{array}
$$

Represent $x_i \neq y_i$ as

$$((x_i = v) - (y_i = v)) \quad \text{for all } v \in D(x_i)$$

So $\overline{\lambda}_i := \lambda_i[v]$

Lagrangian Decomposition for CP

$$\begin{array}{ll} \max & \overline{\lambda}_2(x_2 \neq y_2) + \overline{\lambda}_3(x_3 \neq z_3) + \overline{\lambda}_5(y_5 \neq z_5) \\ \text{s.t.} & \texttt{alldiff}(x_1, x_2, x_3) \\ & \texttt{alldiff}(y_2, y_4, y_5) \\ & \texttt{alldiff}(z_3, z_5) \end{array}$$

Represent $x_i \neq y_i$ as

$((x_i = v) - (y_i = v))$ for all $v \in D(x_i)$

So $\overline{\lambda}_i := \lambda_i[v]$

$$\max \sum_{v \in D(x_2)} \lambda_2[v]((x_2 = v) - (y_2 = v)) + \ldots$$

Lagrangian Decomposition for CP

$$\begin{array}{ll} \max & \bar{\lambda}_2(x_2 \neq y_2) + \bar{\lambda}_3(x_3 \neq z_3) + \bar{\lambda}_5(y_5 \neq z_5) \\ \text{s.t.} & \texttt{alldiff}(x_1, x_2, x_3) \\ & \texttt{alldiff}(y_2, y_4, y_5) \\ & \texttt{alldiff}(z_3, z_5) \end{array}$$

Represent $x_i \neq y_i$ as

$((x_i = v) - (y_i = v))$ for all $v \in D(x_i)$

So $\bar{\lambda}_i := \lambda_i[v]$

$$\max \sum_{v \in D(x_2)} \lambda_2[v]((x_2 = v) - (y_2 = v)) + \ldots$$

$$= \lambda_2[x_2] - \lambda_2[y_2] + \lambda_3[x_3] - \lambda_3[z_3] + \lambda_5[y_5] - \lambda_5[z_5]$$

Final Decomposition

$$\text{obj}_1 = \max \left\{ \; \overline{\lambda}_2[x_2] + \overline{\lambda}_3[x_3] \mid \texttt{alldiff}(x_1, x_2, x_3) \right\}$$

$$\text{obj}_2 = \max \left\{ -\overline{\lambda}_2[y_2] + \overline{\lambda}_5[y_5] \mid \texttt{alldiff}(y_2, y_4, y_5) \right\}$$

$$\text{obj}_3 = \max \left\{ -\overline{\lambda}_3[z_3] - \overline{\lambda}_5[z_5] \mid \texttt{alldiff}(z_3, z_5) \right\}$$

Final Decomposition

$$\left.\begin{aligned}
\mathrm{obj}_1 &= \max\left\{\ \overline{\lambda}_2[x_2] + \overline{\lambda}_3[x_3] \mid \texttt{alldiff}(x_1, x_2, x_3)\right\} \\
\mathrm{obj}_2 &= \max\left\{-\overline{\lambda}_2[y_2] + \overline{\lambda}_5[y_5] \mid \texttt{alldiff}(y_2, y_4, y_5)\right\} \\
\mathrm{obj}_3 &= \max\left\{-\overline{\lambda}_3[z_3] - \overline{\lambda}_5[z_5] \mid \texttt{alldiff}(z_3, z_5)\right\}
\end{aligned}\right\}$$

sum gives *upper bound* on satisfiability

Final Decomposition

$$\left.\begin{aligned}
\mathrm{obj}_1 &= \max\left\{\ \overline{\lambda}_2[x_2] + \overline{\lambda}_3[x_3] \mid \texttt{alldiff}(x_1, x_2, x_3)\right\} \\
\mathrm{obj}_2 &= \max\left\{-\overline{\lambda}_2[y_2] + \overline{\lambda}_5[y_5] \mid \texttt{alldiff}(y_2, y_4, y_5)\right\} \\
\mathrm{obj}_3 &= \max\left\{-\overline{\lambda}_3[z_3] - \overline{\lambda}_5[z_5] \mid \texttt{alldiff}(z_3, z_5)\right\}
\end{aligned}\right\}$$

sum gives *upper bound* on satisfiability

- Can be used for feasibility problems and optimization problems
  - Synchronize the ‘support’ solutions within the constraints
  - Systematic method to improve bounding in CP
  - ‘Generic relaxation’

Final Decomposition

$$\left.\begin{aligned}
\mathrm{obj}_1 &= \max\left\{\ \overline{\lambda}_2[x_2] + \overline{\lambda}_3[x_3] \mid \texttt{alldiff}(x_1, x_2, x_3)\right\} \\
\mathrm{obj}_2 &= \max\left\{-\overline{\lambda}_2[y_2] + \overline{\lambda}_5[y_5] \mid \texttt{alldiff}(y_2, y_4, y_5)\right\} \\
\mathrm{obj}_3 &= \max\left\{-\overline{\lambda}_3[z_3] - \overline{\lambda}_5[z_5] \mid \texttt{alldiff}(z_3, z_5)\right\}
\end{aligned}\right\} \text{sum gives \textit{upper bound} on satisfiability}$$

- Can be used for feasibility problems and optimization problems
  - Synchronize the ‘support’ solutions within the constraints
  - Systematic method to improve bounding in CP
  - ‘Generic relaxation’
- Extended cost-based domain filtering!

‘Global’ Lagrangian Domain Filtering

- Consider Lagrangian Decomposition with subproblems j=1..m
  - Let $z_j|_{x_i=v}$ be the objective value of j-th subproblem, subject to x_i=v

‘Global’ Lagrangian Domain Filtering

- Consider Lagrangian Decomposition with subproblems j=1..m
  - Let $z_j|_{x_i=v}$ be the objective value of j-th subproblem, subject to x_i=v
- We have, given lower bound B:
  - If $\sum_j z_j|_{x_i=v} < B$ then v can be removed from $D(x_i)$

'Global' Lagrangian Domain Filtering

- Consider Lagrangian Decomposition with subproblems j=1..m
  - Let $z_j|_{x_i=v}$ be the objective value of j-th subproblem, subject to x_i=v
- We have, given lower bound B:
  - If $\sum_j z_j|_{x_i=v} < B$ then v can be removed from D(x_i)

E.g., for our example, we can deduce
$x_1 \neq a$, $x_2 \neq b$, $x_3 \neq c$, $x_5 \neq c$

$$\texttt{alldiff}(x_1, x_2, x_3)$$
$$\texttt{alldiff}(x_2, x_4, x_5)$$
$$\texttt{alldiff}(x_3, x_5)$$

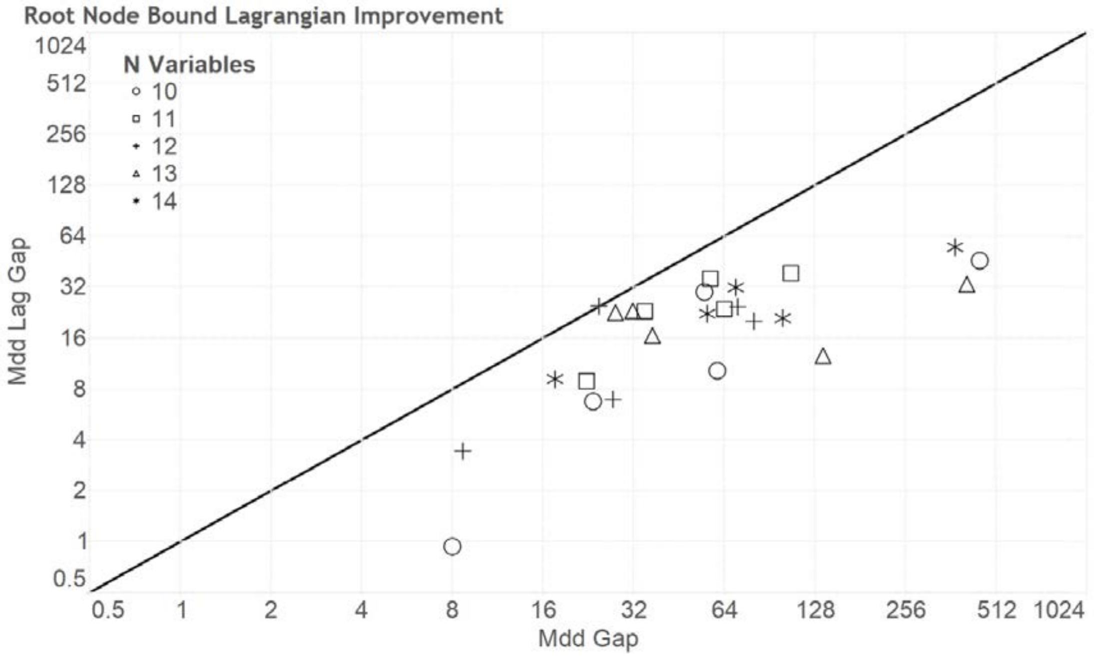
$$x_1 \in \{a, b\}$$
$$x_2 \in \{b, c\}$$
$$x_3 \in \{a, c\}$$
$$x_4 \in \{a, b\}$$
$$x_5 \in \{a, b, c\}$$

Discussion

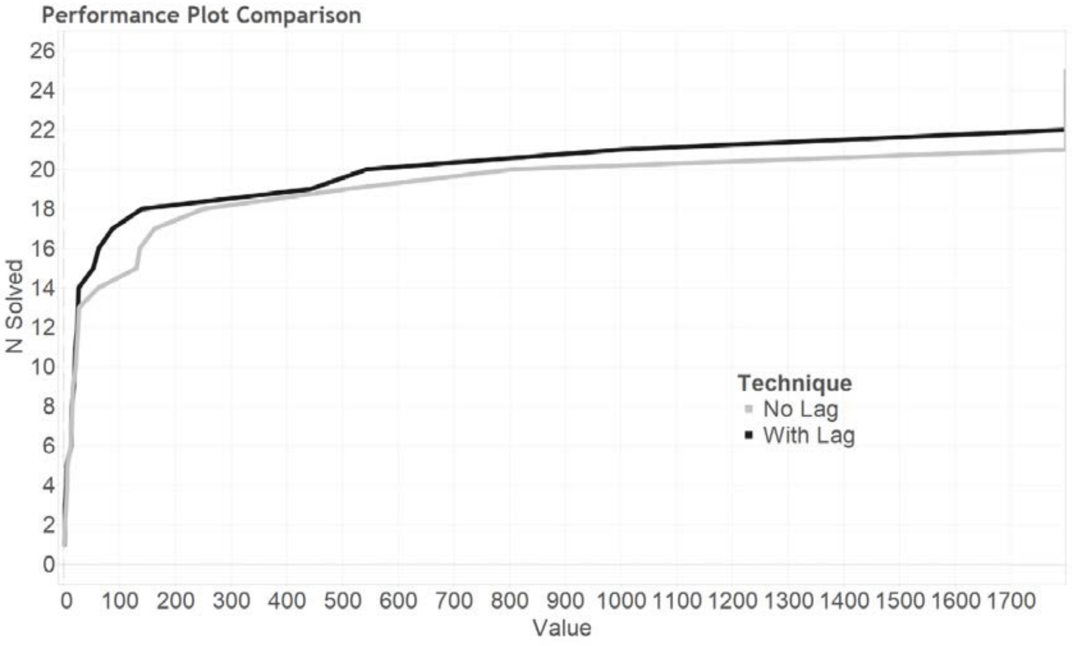
- Keep Lagrangian multipliers under control
  - Apply to a select subset of constraints
  - Can solve at root and keep multipliers fixed during search
  - Optimality not required for Lagrangian relaxation
- Computing of $z_j \,|_{x_i=v}$ may be challenging
  - Depends on constraint structure
  - Can use relaxation of the constraint instead (any bound holds)
  - Can use Relaxed Decision Diagram: Automatic extension

Experimental Results: Alldifferent

root node bound improvement



performance plot comparison

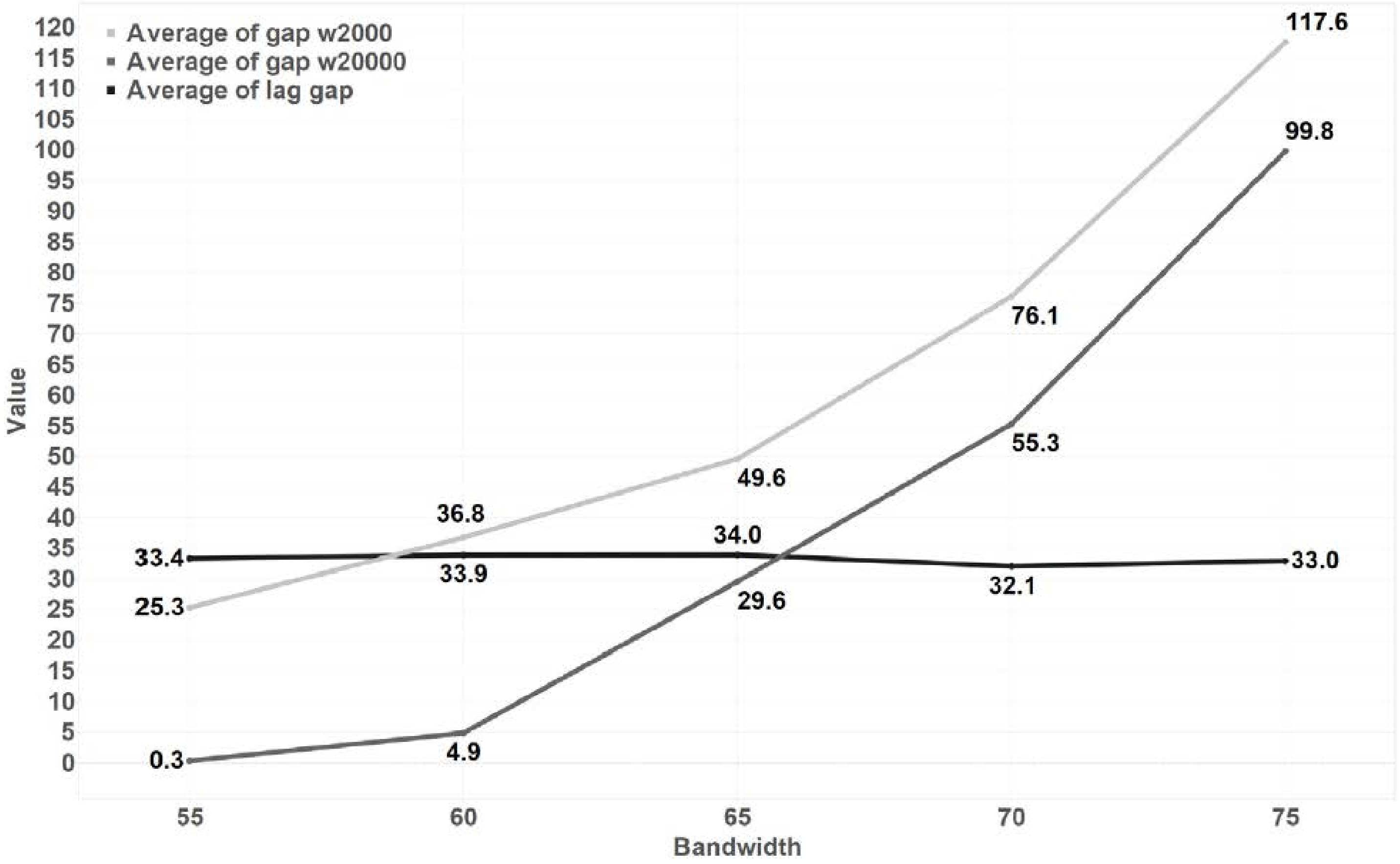


Systems of overlapping alldifferent constraints, with weighted sum as objective.

Each MDD represents a subset of constraints

Experimental Results: Set Covering

optimality gap



- Single MDD relaxation
  - widths 2,000 and 2,0000
- Lagrangian Decomposition
  - split constraints into multiple MDDs
- Problem instances with increasing bandwidth

- Lagrangian Decomposition much more stable!

More References

- Fontaine, Michel, Van Hentenryck [CP 2014]
- Ha, Quimper, Rousseau [CP 2015]
- Bergman, Cire, v.H. [CP 2015]

- Chu, Gange, Stuckey, “Lagrangian Decomposition via Sub-problem Search” CPAIOR 2016
  - Monday May 30, 10:45-12:00 session

Summary

- Lagrangian Relaxation and Decomposition provide systematic and efficient approach to
  - improve optimization bounds in CP
  - improve constraint propagation
- CP's is ideal environment for automated Lagrangian relaxations
  - problem is represented with building blocks (constraints) that communicate
  - "combinatorial programming"