Tight Optimization Bounds from Decision Diagrams

John Hooker

Tepper School of Business
Carnegie Mellon University

INFORMS 2020

Motivation

- Obtain **tight bounds** for hard problems solved by **heuristics**.
  - **Bounds** are needed to judge quality of solutions.
  - It's **really hard** to derive tight bounds for combinatorial problems, except in a branching framework.

Motivation

- Obtain **tight bounds** for hard problems solved by **heuristics**.
  - **Bounds** are needed to judge quality of solutions.
  - It's **really hard** to derive tight bounds for combinatorial problems, except in a branching framework.
- Focus on problems with **dynamic programming** formulations.
  - A general method for **bounding DPs**.
  - Very **different** from state space relaxation.

Motivation

- **Relaxed decision diagrams** can provide bounds.
  - But bounds are **weak** as the problem **scales up**.
- **Lagrangian duality** can provide bounds.
  - But they are usually **weak** because of **duality gap**.
- How about **DDs + Lagrangian**?
  - **When** can they be combined?

Objectives

- Test bed: **job sequencing problems.**
  - Which problems can be practically bounded with DP + Lagrangian?

Build on Recent Work

- Tight DD-based bounds for job sequencing with state-dependent processing times.
  - **Doesn't scale up.**

JH (2017)

- Bounds from DDs+Lagrangian for TSPTW within CP solver.

Bergman, Cire, van Hoeve (2015)

  - Not useful for **stand-alone** DD.
  - We need **general conditions** for combining DDs and Lagrangian relaxation.

Job Sequencing Example

- Problem: sequence jobs with given processing times
  - Minimize **tardiness** subject to **time windows**

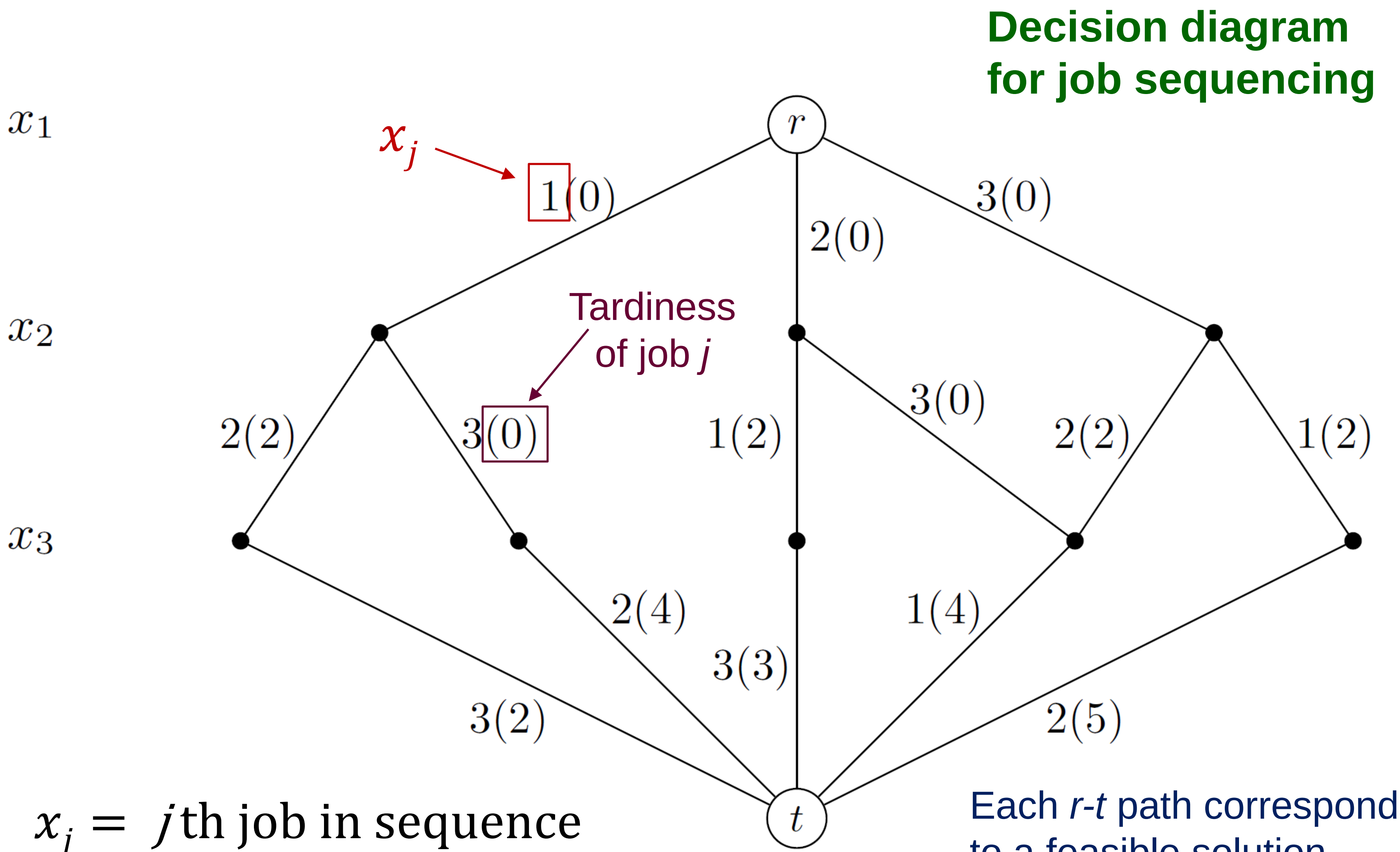
j	r_j	p_j	d_j
1	0	3	5
2	1	2	3
3	1	2	5

Release time

Processing time

Due date

Job Sequencing Example

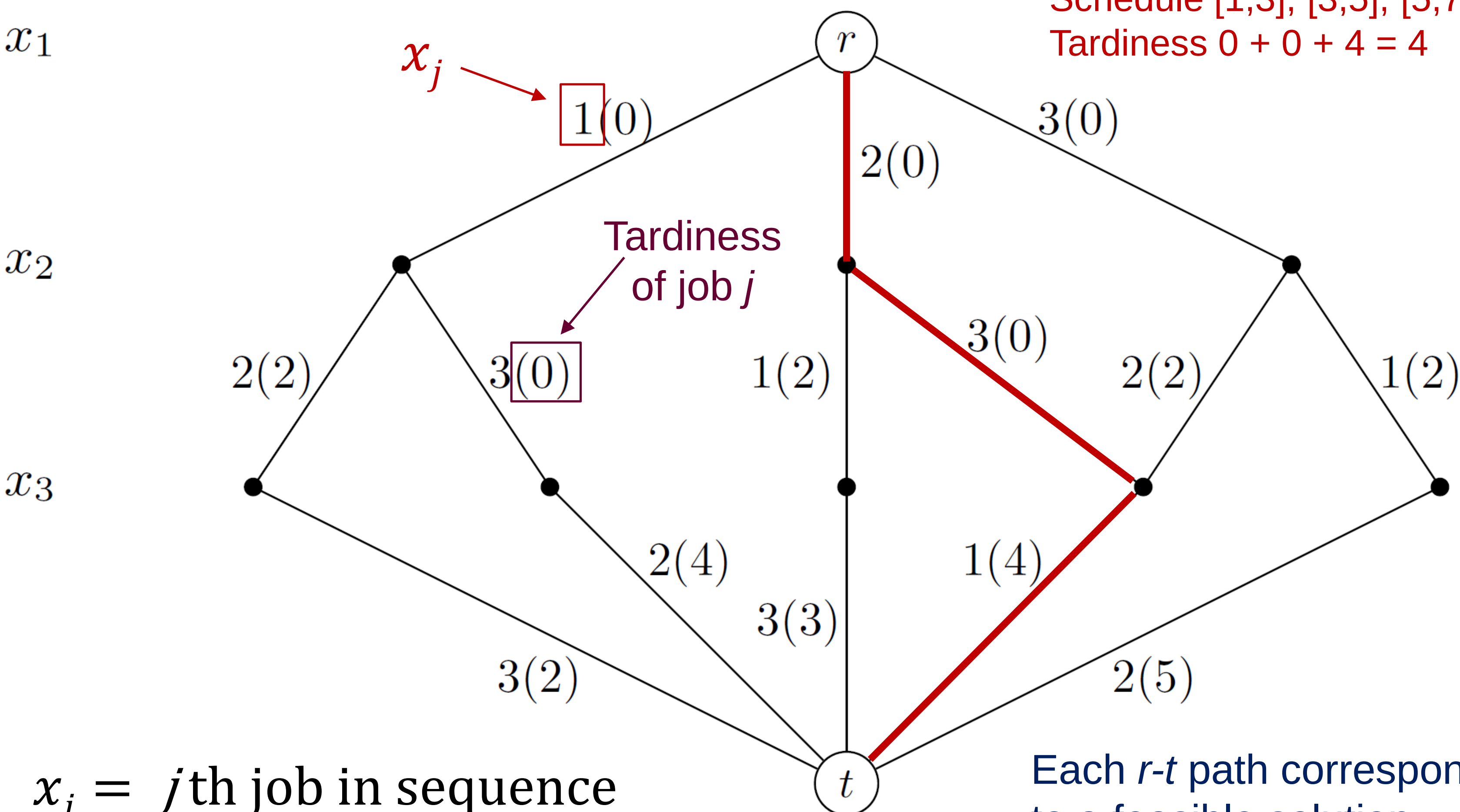


$x_j = j$th job in sequence

Each r-t path corresponds to a feasible solution

Job Sequencing

An optimal solution:
Sequence 2-3-1
Schedule [1,3], [3,5], [5,7]
Tardiness 0 + 0 + 4 = 4



$x_j = j$th job in sequence

Each r-t path corresponds to a feasible solution

Building a Decision Diagram

- Our approach:
  - Associate dynamic programming **states** with nodes..
  - …as in a state transition graph.

Dynamic Programming Model

- Our approach:
  - Associate dynamic programming **states** with nodes..
  - …as in a state transition graph.

General recursive model

State transition function

$$h_i(\boldsymbol{S}_i) = \min_{x_i \in X_i(\boldsymbol{S}_i)} \left\{ c_i(\boldsymbol{S}_i, x_i) + h_{i+1}\big(\phi_i(\boldsymbol{S}_i, x_i)\big) \right\}$$

State in stage *i*

Set of possible controls

Immediate cost

Cost to go

DP Model for Job Sequencing

State: $S_i = (V_i, t_i)$

- V_i: Set of jobs scheduled so far
- t_i: Finish time of last job scheduled

Initial state = $(\emptyset, 0)$

Controls: $X_i(V_i, t_i) = \{1, \ldots, n\} \setminus V_i$

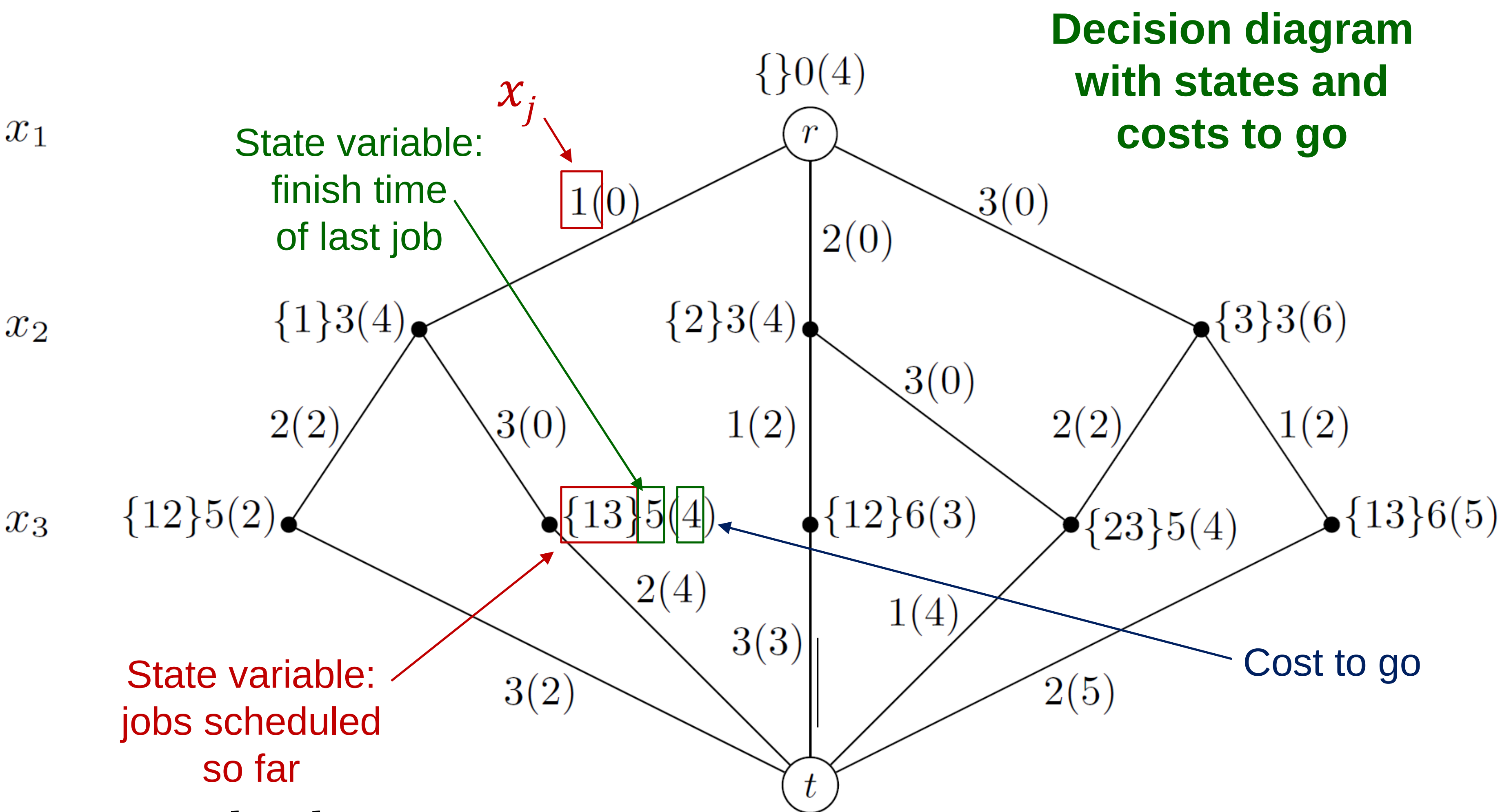
Immediate cost: $c_i\big((V_i, t_i), x_j\big) = \big(\max\{r_{x_i}, t_i\} + p_{x_i} - d_{x_i}\big)^+$

Transition: $\phi_i\big((V_i, t_i), x_i\big) = \big(V_i \cup \{x_i\}, \max\{r_{x_i}, t_i\} + p_{x_i}\big)$

$$h_i(\boldsymbol{S}_i) = \min_{x_i \in X_i(\boldsymbol{S}_i)} \Big\{ c_i(\boldsymbol{S}_i, x_i) + h_{i+1}\big((\phi_i(\boldsymbol{S}_i, x_i)\big) \Big\}$$

- $\boldsymbol{S}_i$: State in stage *i*
- $X_i(\boldsymbol{S}_i)$: Set of possible controls
- $c_i(\boldsymbol{S}_i, x_i)$: Immediate cost
- $h_{i+1}\big((\phi_i(\boldsymbol{S}_i, x_i)\big)$: Cost to go

Job Sequencing Diagram



$x_j =$ jth job in sequence

Relaxed Decision Diagram

- Motivation
  - **Shrink** diagram by allowing some **infeasible paths**.
  - **Shortest path** provides a **lower bound** on the optimal value.

Andersen, Hadžić, JH, Tiedemanmn (2007)

Building a Relaxed Diagram

- Node merger
  - **Merge** some nodes in the **exact** diagram.
  - …to make the diagram **smaller** while excluding no feasible solutions
  - …while introducing some infeasible low-cost solutions.

Andersen, Hadžić, JH, Tiedemanmn (2007)

Ciré and van Hoeve (2013)

Bergman, Ciré, van Hoeve, JH (2013)

Node Merger

- Merge nodes as the diagram is constructed
  - Combine states of the merged nodes in a way that yields a valid relaxation.
  - This may require **additional state variables**.

JH (2017)

Bergman, Ciré, van Hoeve, JH (2013, 2016)

Relaxed DP Model

- In the example, no new states needed
  - Transition function same as before.

Reflects node merger in layer *i* + 1

Recursion:

$$\bar{h}_i(\boldsymbol{S}_i) = \min_{x_i \in X_i(\boldsymbol{S}_i)} \left\{ c_i(\boldsymbol{S}_i, x_i) + \bar{h}_{i+1}\Big(\boxed{\rho_{i+1}} \big(\phi_i(\boldsymbol{S}_i, x_i)\big) \Big) \right\}$$

Relaxed DP Model

Set of jobs scheduled in **all** feasible solutions so far

$$S_i = (V_i, t_i)$$

Initial state = $(\emptyset, 0)$

Earliest possible finish time of immediately previous job

Transition:

$$\phi_i\big((V_i, t_i), x_j\big) = \big(V_i \cup \{x_i\}, \max\{r_{x_i}, t_i\} + p_{x_i}\big)$$

Recursion:

$$\bar{h}_i(\boldsymbol{S}_i) = \min_{x_i \in X_i(\boldsymbol{S}_i)} \Big\{ c_i(\boldsymbol{S}_i, x_i) + \bar{h}_{i+1}\Big(\rho_{i+1}\big(\phi_i(\boldsymbol{S}_i, x_i)\big)\Big)\Big\}$$

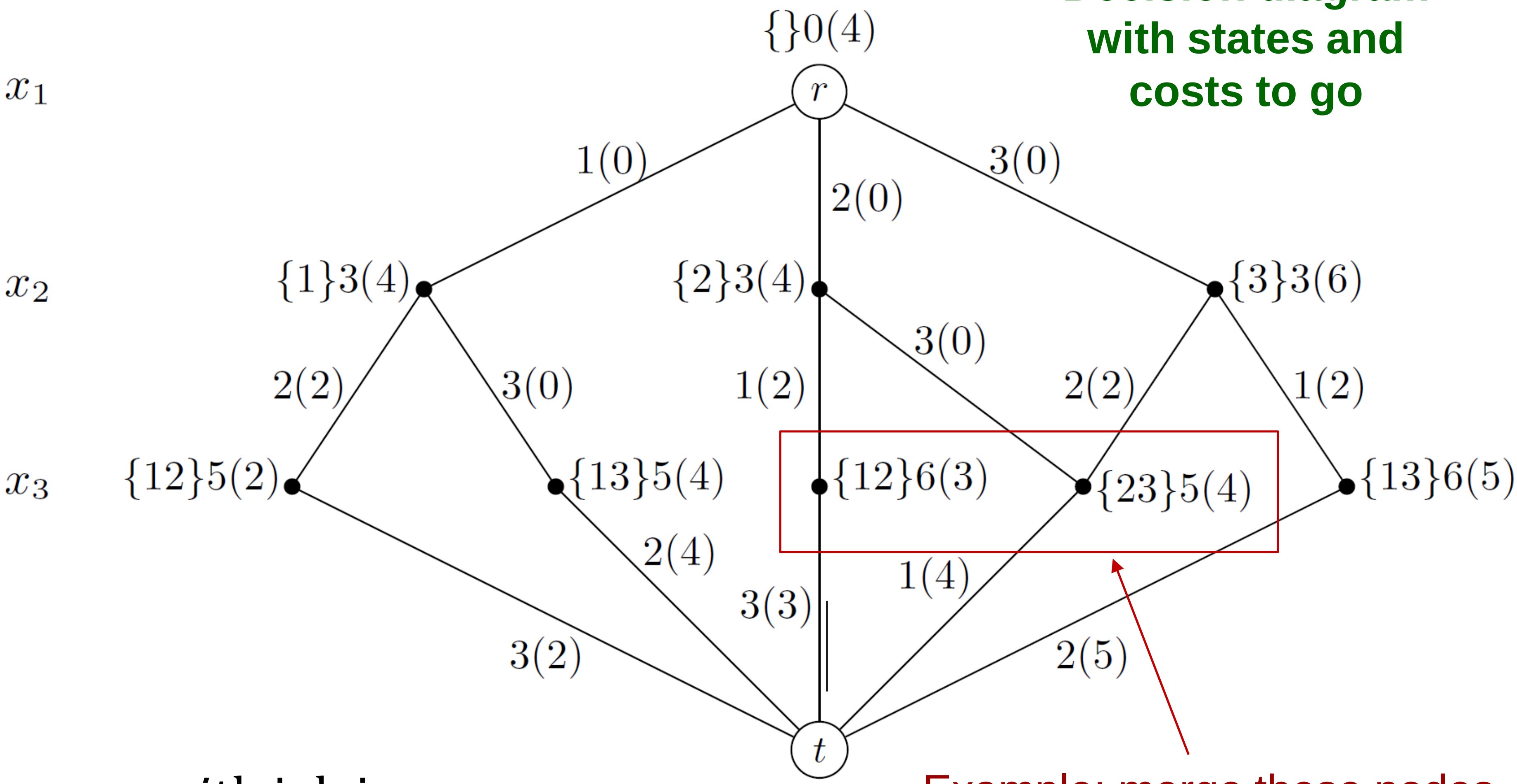
Node Merger in Relaxation

- Merge states as the diagram is constructed
  - States *S*, *T* merge to form state $S \oplus T$
- Merger operation must yield valid relaxation
  - There are **sufficient conditions** for this. JH (2017)
  - In state-dependent job sequencing,

$$(V, t) \oplus (V', t') = \big(V \cap V', \min\{t, t'\}\big)$$

Job Sequencing Diagram

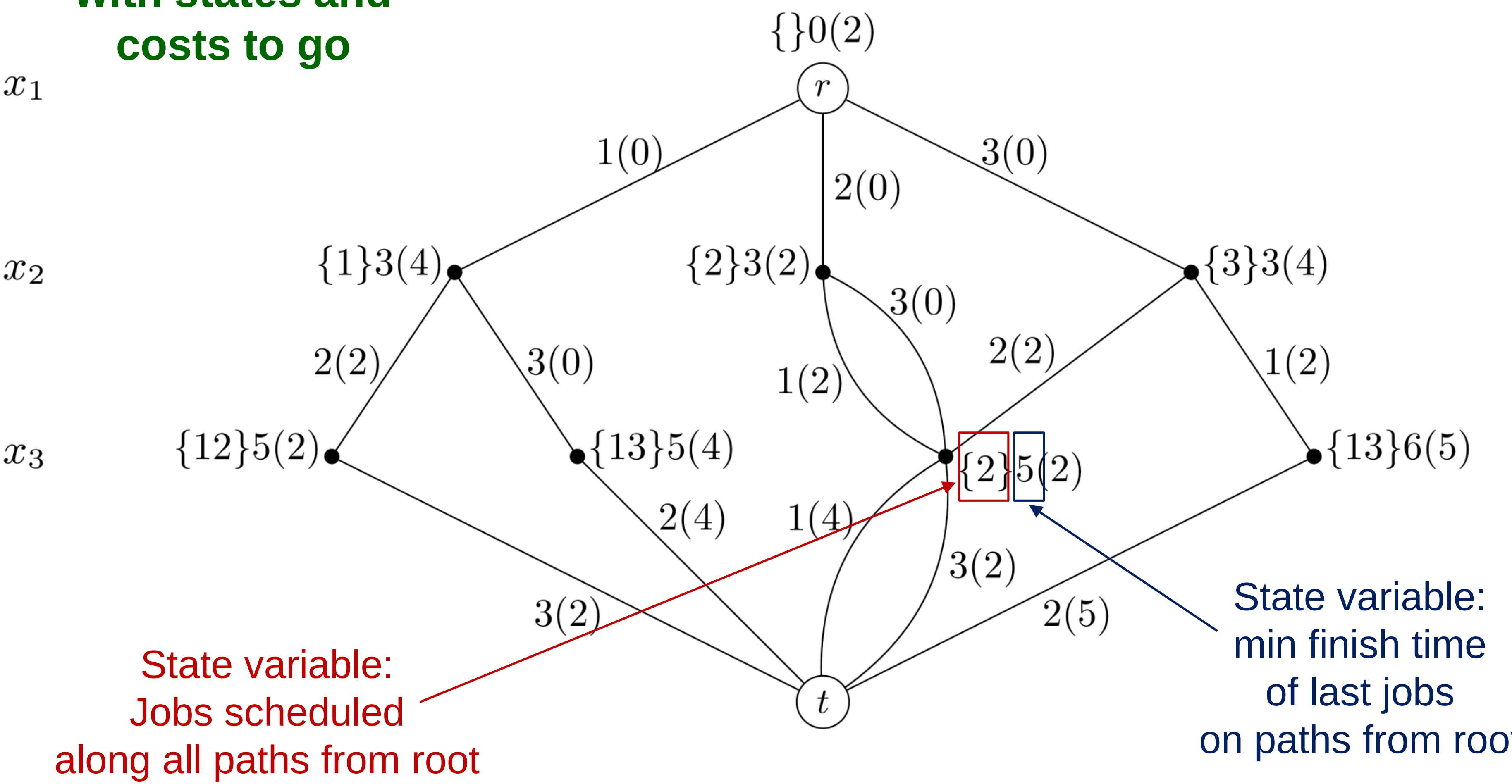
Decision diagram with states and costs to go



$x_i = i$ th job in sequence

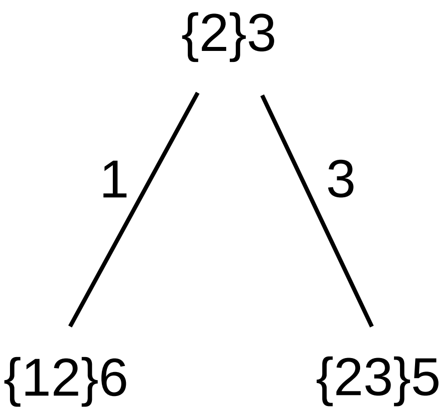
Job Sequencing Relaxed Diagram

Relaxed decision diagram with states and costs to go

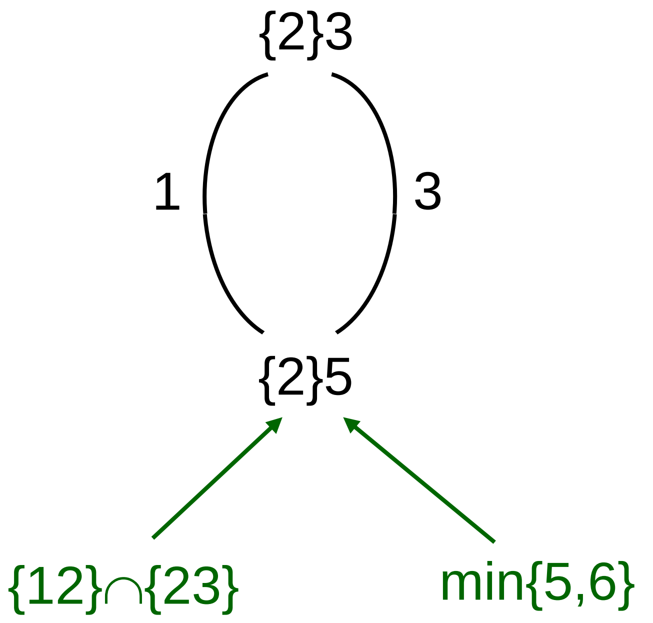


Job Sequencing Node Merger

Without merger



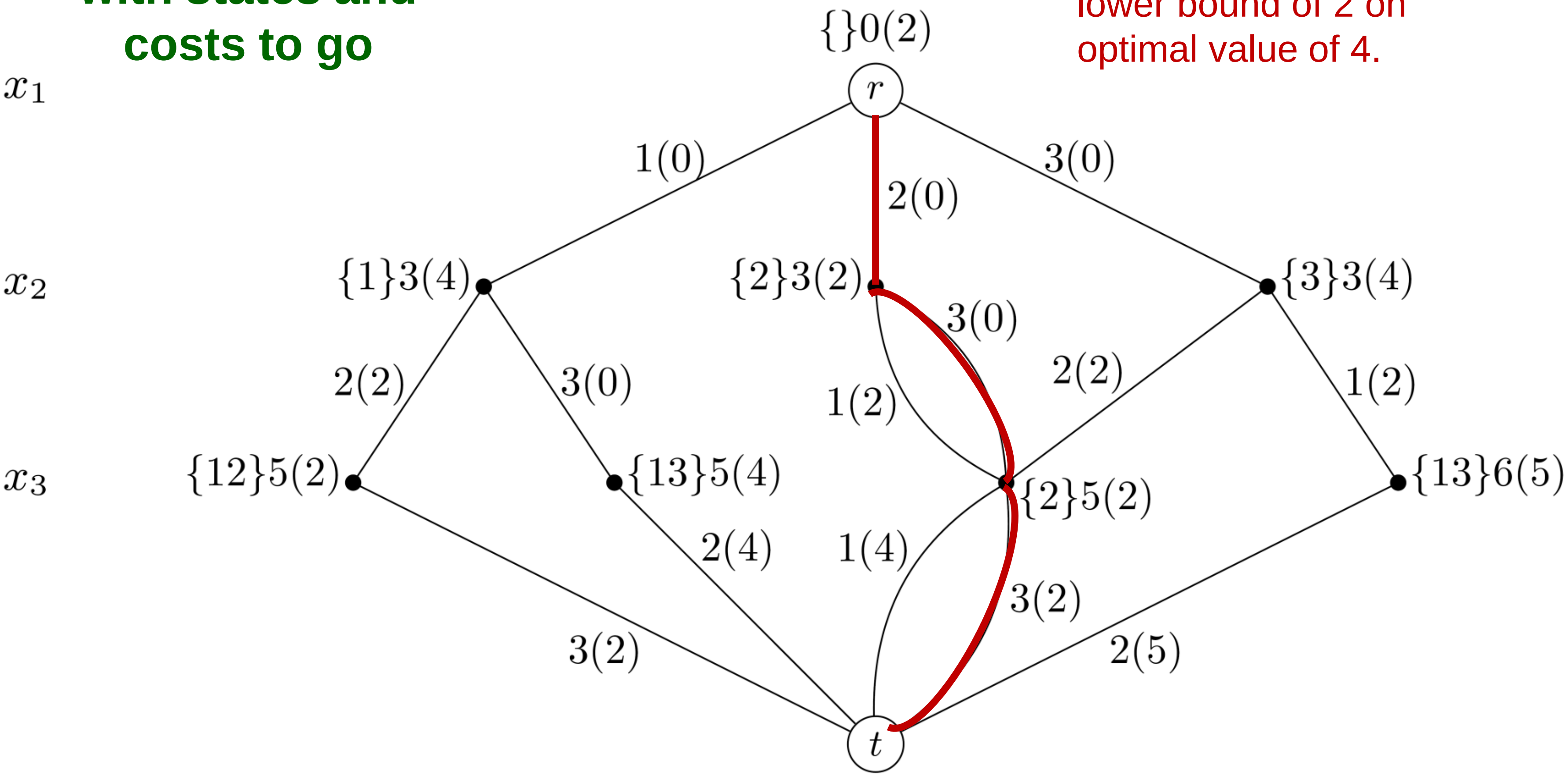
With merger



Job Sequencing Relaxed Diagram

Relaxed decision diagram with states and costs to go

Shortest path yields a lower bound of 2 on optimal value of 4.



Lagrangian Relaxation

- "Dualize" hard constraints.
  - By moving them into the objective functions

Consider a problem:

$$\min_{\boldsymbol{x} \in \boldsymbol{X}} \left\{ f(\boldsymbol{x}) \;\middle|\; \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \right\}$$

Lagrangian Relaxation

- "Dualize" hard constraints.
  - By moving them into the objective functions

Consider a problem:

$$\min_{\boldsymbol{x} \in \boldsymbol{X}} \left\{ f(\boldsymbol{x}) \;\middle|\; \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \right\}$$

Lagrangian relaxation:

$$\theta(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \boldsymbol{X}} \left\{ f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) \right\}$$

Lagrangian Relaxation

- "Dualize" hard constraints.
  - By moving them into the objective functions

Consider a problem:

$$\min_{\boldsymbol{x} \in \boldsymbol{X}} \left\{ f(\boldsymbol{x}) \;\middle|\; \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \right\}$$

Lagrangian relaxation:

$$\theta(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \boldsymbol{X}} \left\{ f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x}) \right\}$$

Lagrangian dual:

$$\max_{\boldsymbol{\lambda}} \left\{ \theta(\boldsymbol{\lambda}) \right\}$$

Lagrangian Relaxation on DD

- "Dualize" hard constraints.
  - By moving them into the objective functions

In our example:

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \quad \Leftrightarrow \quad \text{alldiff}(x_1, \ldots, .x_n)$$

To formulate this, let

$$g(\boldsymbol{x}) = \left(g_1(\boldsymbol{x}), \ldots, g_n(\boldsymbol{x})\right)$$

$$g_j(\boldsymbol{x}) = -1 + \sum_{i=1}^{n} \boxed{[x_i = j]}$$

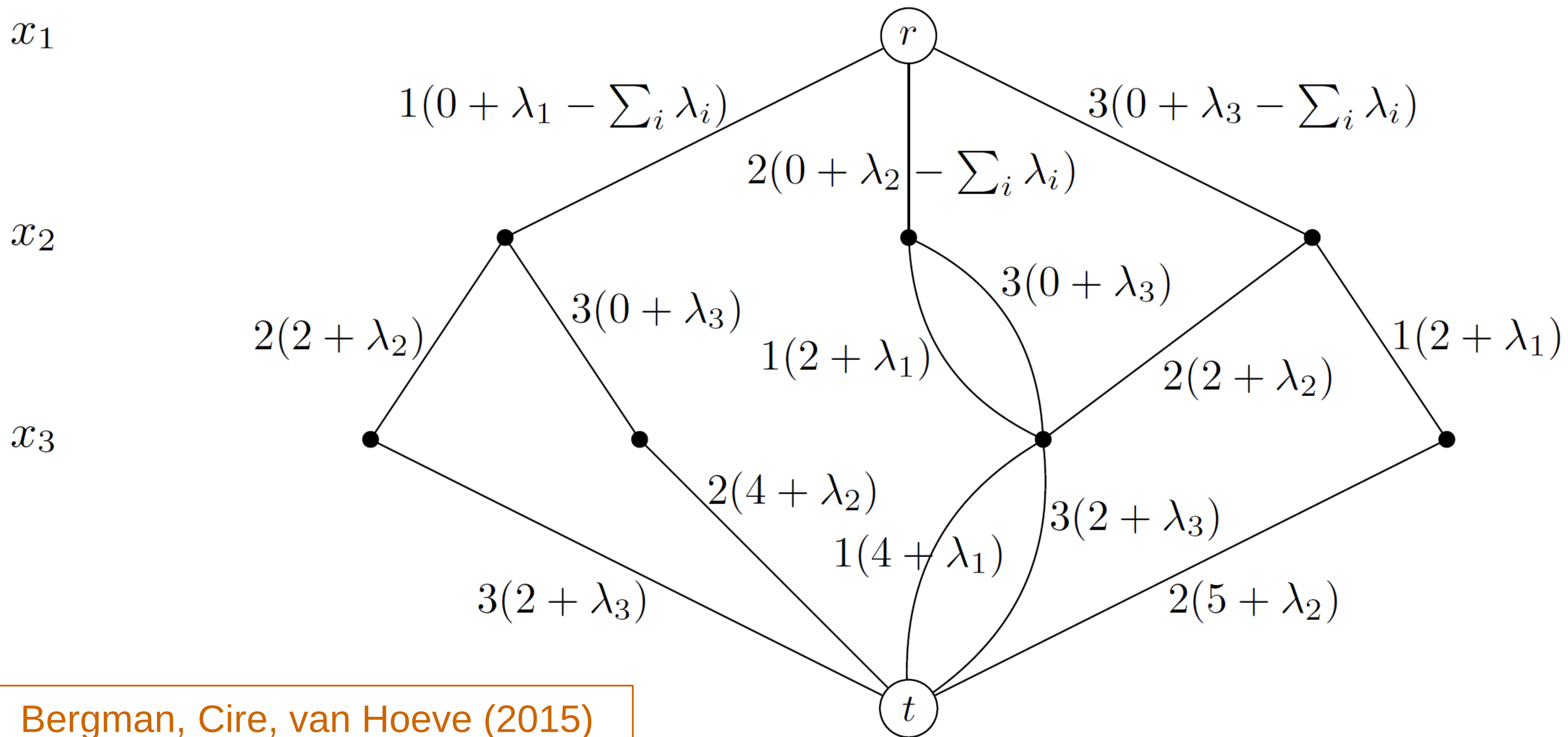
$$= \begin{cases} 1 & \text{if } x_i = j \\ 0 & \text{otherwise} \end{cases}$$

Bergman, Cire, van Hoeve (2015)

Lagrangian Relaxation on DD

Lagrange penalties included in arc costs

Path length now includes total Lagrange penalty



Bergman, Cire, van Hoeve (2015)

Solvilng the Lagrangian Dual

- Solve by subgradient optimization
  - Use **Polyak's method** to determine stepsize

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \sigma_k \, \boldsymbol{g}(\boldsymbol{x}^k)$$

Stepsize, given by

Subgradient, where $\boldsymbol{x}^k$ is value of $\boldsymbol{x}$ obtained when computing $\theta(\boldsymbol{\lambda}^k)$

$$\sigma_k = \frac{\theta^* - \theta(\boldsymbol{\lambda}^k)}{||\boldsymbol{g}(\boldsymbol{x}^k)||_2^2}$$

where θ^* = known upper bound on optimal value. Let θ^* be value of best known heuristic solution

Combining DD & Lagrangian Duality

- Express $\boldsymbol{g}(\boldsymbol{x})$ in terms of *immediate penalty functions*

$$\boldsymbol{g}(\boldsymbol{x}) = \sum_{i=1}^{n} \gamma_i(\bar{\boldsymbol{S}}', x_i)$$

Subset of state variables

  – In our example,

$$\boldsymbol{g}(\boldsymbol{x}) = \sum_{i=1}^{n} \big(-[i=1] + [x_i = 1],\ \ldots,\ -[i=1] + [x_i = n] \big)$$

Here, $\bar{\boldsymbol{S}}'_i = \emptyset$

Combining DD & Lagrangian Duality

- Identify state variables on which immediate cost depends.
  - In our example, cost depends on x_i and state variable t_i

$$c_i\big((V_i, t_i), x_j\big) = \big(\max\{r_{x_i}, t_i\} + p_{x_i} - d_{x_i}\big)^+$$

- Identify state variables on which immediate penalty functions depend
  - In our example, they depend only on x_i and no state variables

$$\boldsymbol{\gamma}_i = \big(-[i=1] + [x_i = 1],\ \ldots,\ -[i=1] + [x_i = n]\big)$$

Combining DD & Lagrangian Duality

Theorem. Lagrangian relaxation can be implemented in a relaxed DD if nodes are merged **only when** their states **agree** on the values of the state variables on which the immediate cost functions and the immediate penalty functions depend.

- Applies to **dynamic programming in general**.
- Useful when immediate cost and penalty functions depend on **only a few state variables**.

Survey of Job Sequencing Problems

- Minimizing **tardiness** subject to **time windows**
  - In our example, **cost depends** on x_i and **state variable t_i**

$$c_i\big((V_i, t_i), x_j\big) = \big(\max\{r_{x_i}, t_i\} + p_{x_i} - d_{x_i}\big)^+$$

  - **Immediate penalty** depends only on control (true of all sequencing problems)
  - We can merge states that agree on t_i.
  - Relaxed DD has **reasonable size**.
  - We will experiment with **Crauwells-Potts-Wassenhove (**CPW) instances.

Survey of Job Sequencing Problems

- Minimizing **earliness + tardiness** wrt time windows
  - Measure lateness by due date d_j and earliness by desired release date e_j.
  - **Cost now depends on** x_i and **2 state variables** s_i, t_i

$$(V_i, t_i), x_j) = \alpha_{x_i}\left(s_i - p_{x_i} + e_{x_i}\right)^+ + \beta_{x_i}\left(t_i + p_{x_i} - d_{x_i}\right)^+$$

  - Merge states that agree on s_i and t_i, which **remain equal** throughout the relaxed DD.
  - **In effect, cost depends on only one state variable**.
  - We will experiment with **Biskup-Feldman** instances.

Survey of Job Sequencing Problems

- Minimizing tardiness with **time-dependent** costs or processing times
  - Two senses:
    - Dependent on **position** of each job in the sequence.
    - Dependent on **clock time** when job is processed.
  - Easy to check that in either case, costs depends only on current stage (not a state variable) and state variable t_i
  - **This is practical**, and similar to previous problems.

Survey of Job Sequencing Problems

- **Traveling salesman** problem
  - …**without** time windows.
  - **Cost depends only on a state variable y_i** representing previous job.

$$c_i\big((V_i, y_i), x_j\big) = p_{y_i x_i}$$

  - **This is practical** and used in Bergman, Cire, van Hoeve (2015)

Survey of Job Sequencing Problems

- **Traveling salesman** problem with **time windows**
  - **Cost depends on state variables t_i and y_i.**

$$c_i\big((V_i, y_i, t_i), x_j\big) = (r_{x_i} - t_i)^+ + p_{y_i x_i}$$

  - Mergers must agree on **two state variables** and can result in **huge** relaxed DD.
  - This is confirmed by experiments on Dumas instances.
  - **Not practical** for stand-alone DD.

Survey of Job Sequencing Problems

- Minimizing tardiness with state-dependent processing times.
  - Cost depends on state variables t_i and U_i.

$$c_i\big((V_i, U_i, t_i), x_j\big) = \Big(\max\{r_{x_i}, t_i\} + p_{x_i}(U_i) - d_{x_i}\Big)^+$$

  - Mergers must agree on **two state variables** and can result in huge relaxed DD.
  - **Not practical.**

Computational Results

- To test quality of bound…
  - We need instances with **known optimal solutions** or **very good heuristic solutions**.

Computational Results

- 50 Crauwels-Potts-Wassenhove (CPW) instances
  - Only a handful solved to optimality in 1998
  - **Most have been solved** to proven optimality since then.

Computational Results

- 50 Crauwels-Potts-Wassenhove (CPW) instances
  - Only a handful solved to optimality in 1998
  - **Most have been solved** to proven optimality since then.
- 60 Biskup-Feldman instances
  - **None solved** to proven optimality
  - **No useful bounds** known
  - Compare with best known solutions (Ying, Lin, Lu 2017)

Computational Results

- 50 Crauwels-Potts-Wassenhove (CPW) instances
  - Only a handful solved to optimality in 1998
  - **Most have been solved** to proven optimality since then.
- 60 Biskup-Feldman instances
  - **None solved** to proven optimality
  - **No useful bounds** known
  - Compare with best known solutions (Ying, Lin, Lu 2017)

– We need a gap < 1% or 2% to be really useful

Implementation

- Code written in C++
  - Run on my laptop.
- Solving the Lagrangean dual
  - Convergence typically slow for Lagrangian duality.
    - Let it run for 50,000 iterations
    - Bound almost as good if truncated earlier.
    - **Almost all reported computation time is due to solving Lagrangian dual.**

Computational Results

CPW instances, 40 jobs

Instance	Target	Bound	Gap	Percent gap
1	913	883	30	3.29%
2	1225	1179	46	3.76%
3	537	483	54	10.06%
4	2094	2047	47	2.24%
5	990	980	10	1.01%
6	6955	6939	16	0.23%
7	6324	6299	25	0.40%
8	6865	6743	122	1.78%
9	16225	16049	176	1.08%
10	9737	9591	146	1.50%
11	17465	17417	48	0.27%
12	19312	19245	67	0.35%
13	29256	29003	253	0.86%

*Best known solution

Instance	Target	Bound	Gap	Percent gap
14	*14377	14100	277	1.93%
15	26914	26755	159	0.59%
16	72317	72120	197	0.27%
17	78623	78501	122	0.16%
18	74310	74131	179	0.24%
19	77122	77083	39	0.05%
20	63229	63217	12	0.02%
21	77774	77754	20	0.03%
22	100484	100456	28	0.03%
23	135618	135617	1	0.001%
24	119947	119914	33	0.03%
25	128747	128705	42	0.03%

*Best known solution

Time = about 20 minutes per instance

Computational Results

CPW instances, 50 jobs

50 jobs				
Instance	Target	Bound	Gap	Percent gap
1	2134	2100	34	1.59%
2	1996	1864	132	6.61%
3	2583	2552	31	1.20%
4	2691	2673	18	0.67%
5	1518	1342	176	11.59%
6	26276	26054	222	0.84%
7	11403	11128	275	2.41%
8	8499	8490	9	0.11%
9	9884	9507	377	3.81%
10	10655	10594	61	0.57%
11	*43504	43472	32	0.07%
12	*36378	36303	75	0.21%
13	45383	45310	73	0.16%

*Best known solution

50 jobs				
Instance	Target	Bound	Gap	Percent gap
14	*51785	51702	83	0.16%
15	38934	38910	47	0.12%
16	87902	87512	390	0.44%
17	84260	84066	194	0.23%
18	104795	104633	162	0.15%
19	*89299	89163	136	0.15%
20	72316	72222	94	0.13%
21	214546	214476	70	0.03%
22	150800	150800	0	0%
23	224025	223922	103	0.05%
24	116015	115990	25	0.02%
25	240179	240172	7	0.003%

*Best known solution

Time = about 40 minutes per instance

Computational Results

- **CPW results**
  - Bounds are reasonably tight.
  - 42 of 50 bounds < 2%
  - 35 of 50 bounds < 1%.
  - 13 of 50 bounds < 0.1%
  - 3 bounds really bad
  - Optimality proved for 1 instance.

Computational Results

Biskup-Feldman instances, 20 jobs

Instance	$(h_1, h_2) = (0.1, 0.2)$			
	Target	Bound	Gap	Percent gap
20 jobs				
1	4089	4089	0	0%
2	8251	8244	7	0.08%
3	5881	5877	4	0.07%
4	8977	8971	6	0.07%
5	4028	4024	4	0.10%
6	6306	6288	18	0.29%
7	10204	10204	0	0%
8	3742	3739	3	0.08%
9	3317	3310	7	0.21%
10	4673	4669	4	0.09%

Instance	$(h_1, h_2) = (0.2, 0.5)$			
	Target	Bound	Gap	Percent gap
20 jobs				
1	1162	1162	0	0%
2	2770	2766	4	0.14%
3	1675	1669	6	0.36%
4	3113	3108	5	0.16%
5	1192	1187	5	0.42%
6	1557	1557	0	0%
7	13573	3569	4	0.11%
8	990	979	11	1.11%
9	1056	1055	1	0.09%
10	1355	1349	6	0.44%

Time = about 30 seconds per instance

Computational Results

Biskup-Feldman instances, 50 jobs

Instance	$(h_1, h_2) = (0.1, 0.2)$ Target	Bound	Gap	Percent gap
50 jobs				
1	39250	39250	0	0%
2	29043	29043	0	0%
3	33180	33180	0	0%
4	25856	25847	9	0.03%
5	31456	31439	17	0.05%
6	33452	33444	8	0.02%
7	42234	42228	6	0.01%
8	42218	42203	15	0.04%
9	33222	33218	4	0.01%
10	31492	31481	11	0.03%

Instance	$(h_1, h_2) = (0.2, 0.5)$ Target	Bound	Gap	Percent gap
50 jobs				
1	12754	12752	2	0.02%
2	8468	8463	5	0.06%
3	9935	9935	0	0%
4	7373	7335	38	0.52%
5	8947	8938	9	0.10%
6	10221	10213	8	0.08%
7	12002	11981	21	0.17%
8	11154	11141	13	0.12%
9	10968	10965	3	0.03%
10	9652	9650	3	0.03%

Time = about 8 minutes per instance

Computational Results

Biskup-Feldman instances, 100 jobs

Instance	$(h_1, h_2) = (0.1, 0.2)$ Target	Bound	Gap	Percent gap
100 jobs				
1	139573	139556	17	0.01%
2	120484	120465	19	0.02%
3	124325	124289	36	0.03%
4	122901	122876	25	0.02%
5	119115	119101	14	0.01%
6	133545	133536	9	0.007%
7	129849	129830	19	0.01%
8	153965	153958	7	0.005%
9	111474	111466	8	0.007%
10	112799	112792	7	0.006%

Instance	$(h_1, h_2) = (0.2, 0.5)$ Target	Bound	Gap	Percent gap
100 jobs				
1	39495	39467	28	0.07%
2	35293	35266	27	0.08%
3	38174	38150	24	0.06%
4	35498	35467	31	0.09%
5	34860	34826	34	0.10%
6	35146	35123	23	0.07%
7	39336	39303	33	0.08%
8	44963	44927	36	0.08%
9	31270	31231	39	0.12%
10	34068	34048	20	0.06%

Time = about 65 minutes per instance

Computational Results

- **Biskup-Feldman results**
  - Bounds are very tight
    - perhaps even tighter wrt optimal values
  - 60 of 60 bounds < 2%
  - 59 of 60 bounds < 1%.
  - 44 of 60 bounds < 0.1%
  - 12 of 50 bounds < 0.01%
  - Optimality proved for 8 instances (closing these instances)

Future Work

- Explore DP models for job shop scheduling, etc.
  - Check if DD + Lagrangian relaxation is practical

- Extend to other DP models.

- Extend Lagrangian relaxation to stochastic DDs.
  - They currently provide weak bounds.

Future Work

- Explore DP models for job shop scheduling, etc.
  - Check if DD + Lagrangian relaxation is practical
- Extend to other DP models.
- Extend Lagrangian relaxation to stochastic DDs.
  - They currently provide weak bounds.

For more details:
J. N. Hooker, Improved job sequencing bounds from decision diagrams, *Proceedings of Principles and Practice of Constraint Programming*, 2019