Morning session

Discuss two modeling/solving issues:

1. *Handling the objective function in CP*
   - *Dealing with NP-hard sub-problems involving costs*
   - *Focus on Lagrangian based filtering*
2. *Granularity of models, choice of domains*

Two supporting applications:

1. *The Traveling Purchaser Problem*
2. *Multileaf collimator sequencing (in radiotherapy)*

Hadrien Cambazard
G-SCOP - Université de Grenoble

G-SCOP
SCIENCES POUR LA CONCEPTION, L'OPTIMISATION ET LA PRODUCTION

NP-hard sub-problems involving costs:
Lagrangian based filtering

Hadrien Cambazard

G-SCOP - Université de Grenoble

Plan

1. Context and motivation
   - Illustrative application: the Traveling Purchaser Problem
   - *Optimization* versus *Satisfaction*
   - *Combinatorial* versus *polyhedral* methods

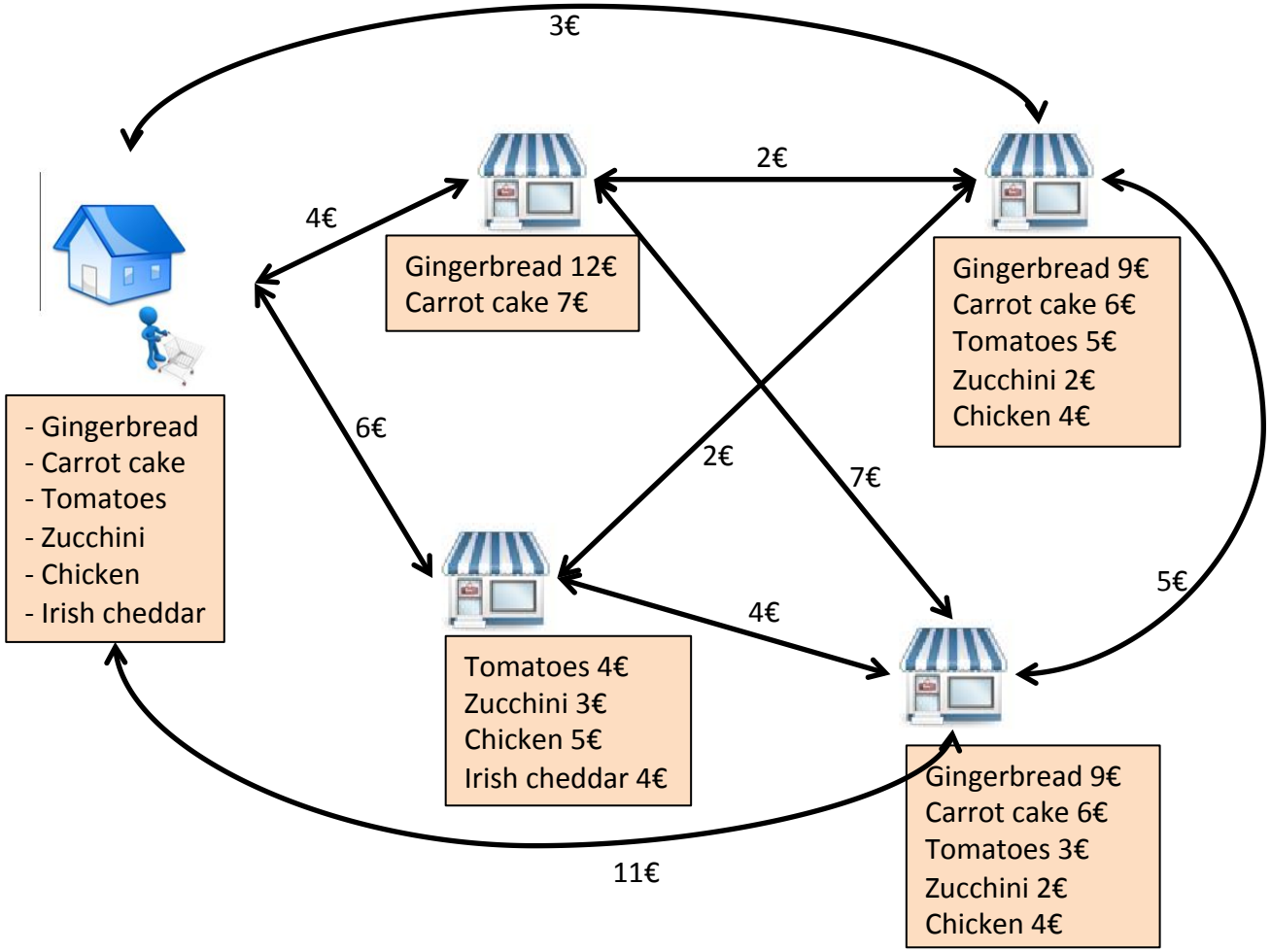
2. Propagation based on Lagrangian Relaxation
   - Lagrangian duality
   - Filtering using Lagrangian reduced costs
   - Let's try on the *Nvalue* global constraint

3. Overview of some NP-Hard Constraints with costs
   - *Multi-cost regular, Weighted-circuit, Weighted-Nvalue, Bin-packing with usage costs*

4. Examples of applications

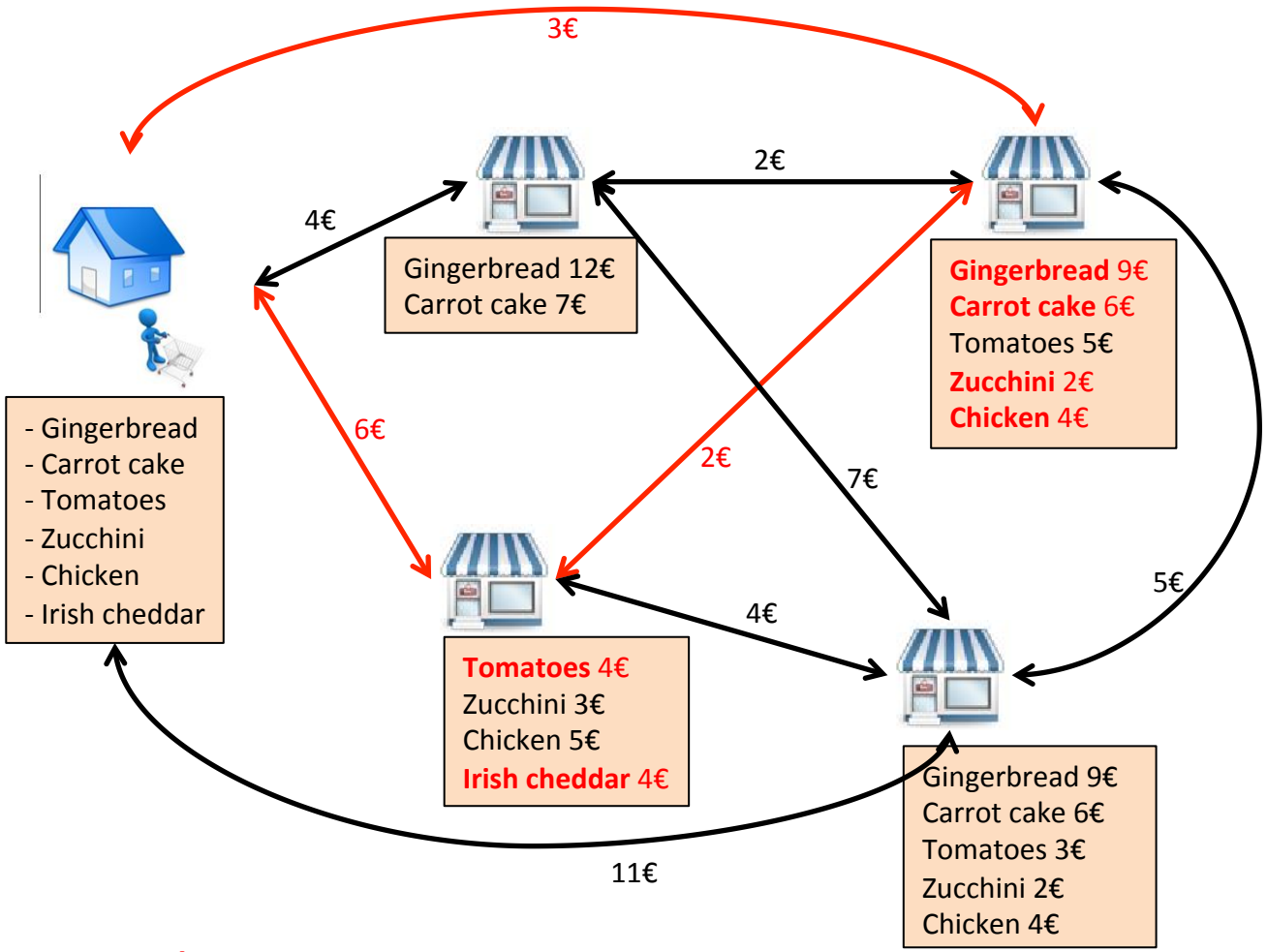
Illustrative Application
The Traveling Purchaser Problem

Traveling Purchaser Problem (TPP)



- **A set of items**
- **A set of markets**, each selling some of the items at different prices
- **The traveling costs** between markets (and from/to home)

Traveling Purchaser Problem (TPP)



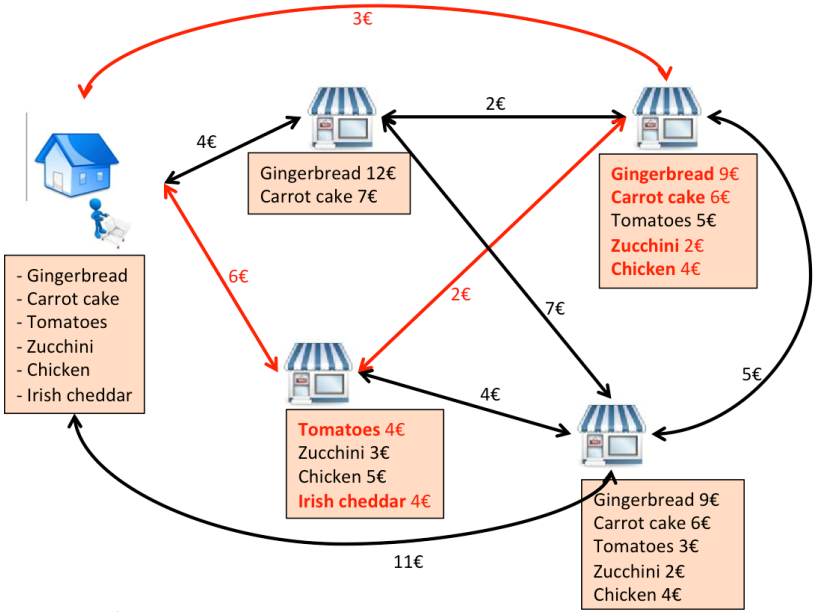
- **A set of items**
- **A set of markets**, each selling some of the items at different prices
- **The traveling costs** between markets (and from/to home)

Traveling cost = 6 + 2 + 3 = 11

Shopping cost = 4 + 4 + 9 + 6 + 2 + 4 = 29

Total cost = 40

Traveling Purchaser Problem (TPP)



Find the route minimizing the sum of traveling and shopping costs to buy all the items

Generalization of TSP

Numerous heuristics

Best known exact method based on Branch and Cut and Price. [G. Laporte, 2003]

[T. Ramesh, 1981]
[G. Laporte, 2003]
[J. Riera-Ledesma, 2006]
[L.Gouveia, 2011]

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip

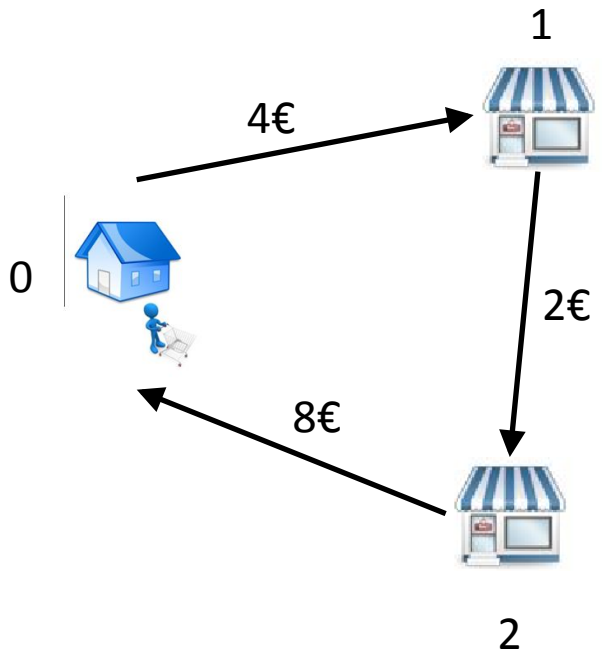
$next_i = i$ (i not visited)

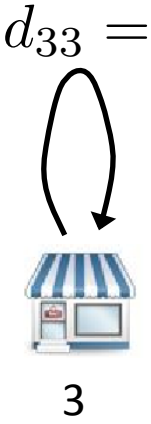
Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip

$next_i = i$ (i not visited)





$$next_0 = 1$$
$$next_1 = 2$$
$$next_2 = 0$$
$$next_3 = 3$$

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$Minimize \sum_i Ct_i + \sum_k Cs_k$

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$Minimize \sum_i Ct_i + \sum_k Cs_k$

Price of item k in market i

$\text{ELEMENT}(Cs_k, prices = [b_{k1}, \ldots b_{ki}, \ldots, b_{km}], s_k) \quad \forall k$

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$$Minimize \sum_i Ct_i + \sum_k Cs_k$$

Price of item k in market i

$$\textsc{Element}(Cs_k, prices = [b_{k1}, \ldots b_{ki}, \ldots, b_{km}], s_k) \quad \forall k$$

$$Cs_k = prices[s_k]$$

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$Minimize \sum_i Ct_i + \sum_k Cs_k$

Price of item k in market i

$\textsc{Element}(Cs_k, prices = [b_{k1}, \ldots, b_{ki}, \ldots, b_{km}], s_k) \quad \forall k$

$\textsc{Element}(Ct_i, [d_{i1}, \ldots, d_{ij}, \ldots, d_{in}], next_i) \quad \forall i$

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$Minimize \sum_i Ct_i + \sum_k Cs_k$

Price of item k in market i

$\text{ELEMENT}(Cs_k, prices = [b_{k1}, \ldots, b_{ki}, \ldots, b_{km}], s_k) \quad \forall k$

$\text{ELEMENT}(Ct_i, [d_{i1}, \ldots, d_{ij}, \ldots, d_{in}], next_i) \quad \forall i$

Traveling cost from i to j

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$Minimize \sum_i Ct_i + \sum_k Cs_k$

Price of item k in market i

$\text{ELEMENT}(Cs_k, prices = [b_{k1}, \ldots, b_{ki}, \ldots, b_{km}], s_k) \quad \forall k$

$\text{ELEMENT}(Ct_i, [d_{i1}, \ldots, d_{ij}, \ldots, d_{in}], next_i) \quad \forall i$

Traveling cost from i to j $(d_{ii} = 0)$

Let's start modeling

Variables:

$next_i \in \{0, 1, \ldots, n\}$: the successor of market **i** in the shopping trip
$next_i = i$ (i not visited)

$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought

$Cs_k \geq 0$: the price paid for item **k**

$Ct_i \geq 0$: the price paid for traveling from market **i** to its successor

$Minimize \sum_i Ct_i + \sum_k Cs_k$

Price of item k in market i

$\text{ELEMENT}(Cs_k, prices = [b_{k1}, \ldots, b_{ki}, \ldots, b_{km}], s_k) \quad \forall k$

$\text{ELEMENT}(Ct_i, [d_{i1}, \ldots, d_{ij}, \ldots, d_{in}], next_i) \quad \forall i$

Traveling cost from i to j $(d_{ii} = 0)$

… the $next$ variables must form a circuit + single loops …

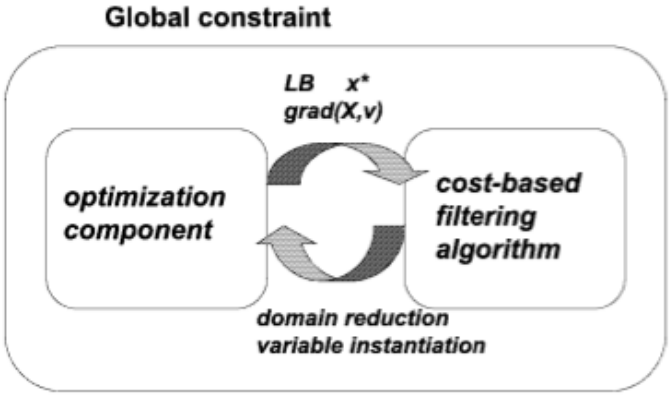
… channel s and $next$

Optimization in CP

- Objective is decomposed (using Element constraints):
  - Resulting lower bound is often very weak
  - *Infeasible* values are eliminated but not *sub-optimal* ones.

- *sub-optimal* = *infeasible* regarding the best known upper-bound

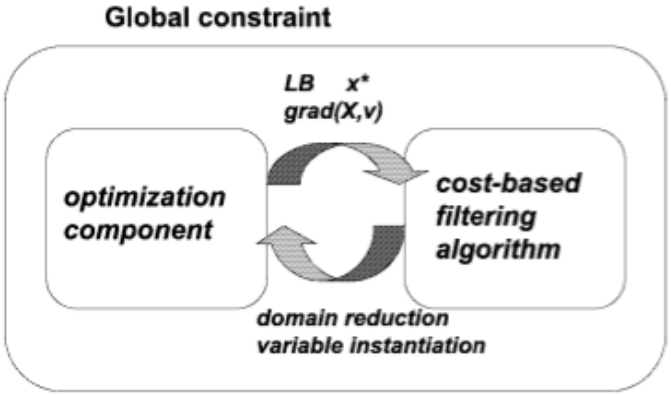
Optimization in CP



(Picture from [Foccaci, 2002])

- Cost-based filtering
  - [Focacci, Lodi, Milano, 2002]: *Embedding relaxations in global constraints for solving TSP and TSPTW*
  - Relaxations based on assignments, spanning tree

Optimization in CP

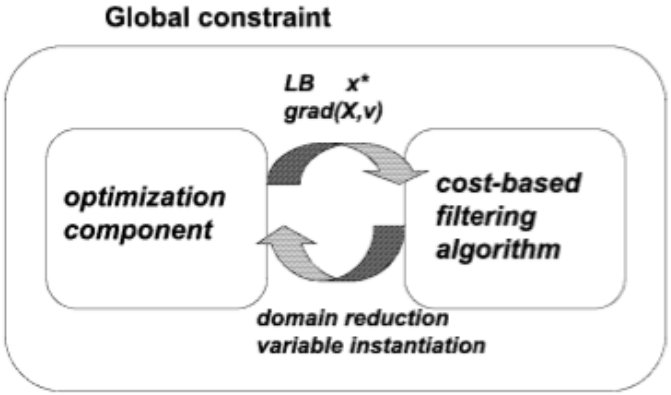


(Picture from [Foccaci, 2002])

- Cost-based filtering
  - [Focacci, Lodi, Milano, 2002]: *Embedding relaxations in global constraints for solving TSP and TSPTW*
  - Relaxations based on assignments, spanning tree

- Linear relaxation of global constraints
  - [Refalo, 2000]: *Linear formulation of Constraint Programming models and Hybrid Solvers*

Optimization in CP



(Picture from [Foccaci, 2002])

- Cost-based filtering
  - [Focacci, Lodi, Milano, 2002]: *Embedding relaxations in global constraints for solving TSP and TSPTW*
  - Relaxations based on assignments, spanning tree
- Linear relaxation of global constraints
  - [Refalo, 2000]: *Linear formulation of Constraint Programming models and Hybrid Solvers*
- Back to the TPP: what cost-based filtering can be done ?

TPP: cost based filtering ?

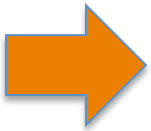
- A CP model reveals combinatorial structures of the problem
- The traveler has to visit *a minimum number of markets* to buy everything
  - Lower bound of traveling cost
- The traveler *can not visit too many markets* (traveling cost would be too high w.r.t to known upper bound)
  - Lower bound of shopping cost
- Number of markets visited: $Nvisit$

Problem structure 1 : *Hitting set*

- Look only at **feasibility**
- Can we buy everything in less than $\overline{Nvisit}$ markets ?

Problem structure 1 : *Hitting set*

- Look only at **feasibility**
- Can we buy everything in less than $\overline{Nvisit}$ markets ?



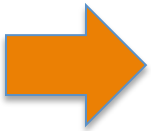
Hitting Set Problem

$$\overline{Nvisit} = 3$$

Gingerbread:	{M2, M3, M6, M7}
Carrot cake:	{M2, M5}
Organic tomatoes:	{M1, M2, M4, M6}
Zucchini:	{M3, M4, M7}
Chicken:	{M1, M4}
Irish cheddar:	{M8, M9}

Problem structure 1 : *Hitting set*

- Look only at **feasibility**
- Can we buy everything in less than $\overline{Nvisit}$ markets ?



Hitting Set Problem

$\overline{Nvisit} = 3$

{M2, M4, M8}

Gingerbread: {**M2**, M3, M6, M7}

Carrot cake: {**M2**, M5}

Organic tomatoes: {M1, **M2**, M4, M6}

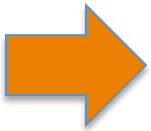
Zucchini: {M3, **M4**, M7}

Chicken: {M1, **M4**}

Irish cheddar: {**M8,** M9}

Problem structure 1 : *Hitting set*

- Look only at **feasibility**
- Can we buy everything in less than $\overline{Nvisit}$ markets ?



Hitting Set Problem

$\overline{Nvisit} = 3$

{M2, M4, M8}

Gingerbread: {**M2**, M3, M6, M7}

Carrot cake: {**M2**, M5}

Organic tomatoes: {M1, **M2**, M4, M6}

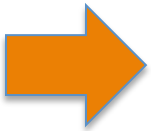
Zucchini: {M3, **M4**, M7}

Chicken: {M1, **M4**}

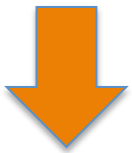
Irish cheddar: {**M8,** M9}

Problem structure 1 : *Hitting set*

- Look only at **feasibility**
- Can we buy everything in less than $\overline{Nvisit}$ markets ?



Hitting Set Problem



In CP:

AtMostNValue

Gingerbread: {**M2**, M3, M6, M7}

Carrot cake: {**M2**, M5}

Organic tomatoes: {M1, **M2**, M4, M6}

Zucchini: {M3, **M4**, M7}

Chicken: {M1, **M4**}

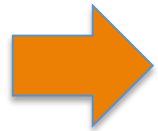
Irish cheddar: {**M8,** M9}

Problem structure 2 : *p-median*

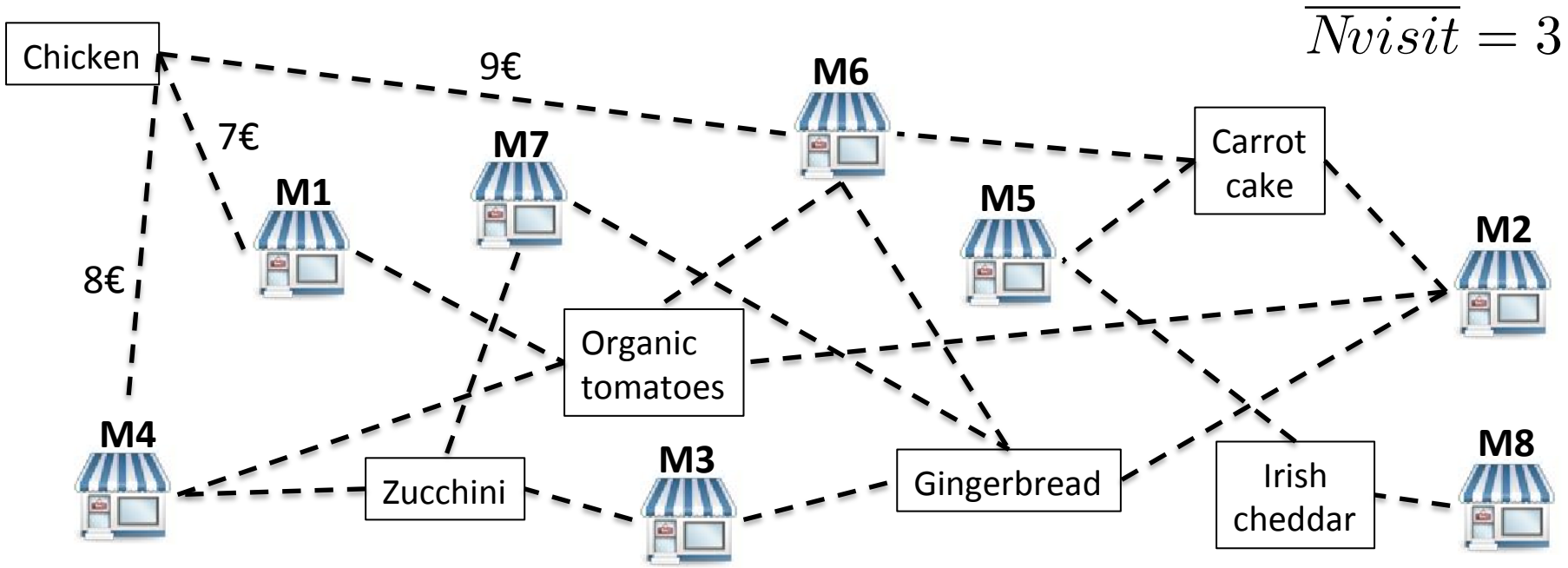
- Look only at **feasibility + shopping cost**
- What is the cheapest way to buy everything in less than $\overline{Nvisit}$ markets ?

Problem structure 2 : *p-median*

- Look only at **feasibility + shopping cost**
- What is the cheapest way to buy everything in less than $\overline{Nvisit}$ markets ?

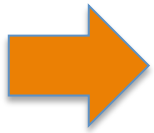


p-median Problem

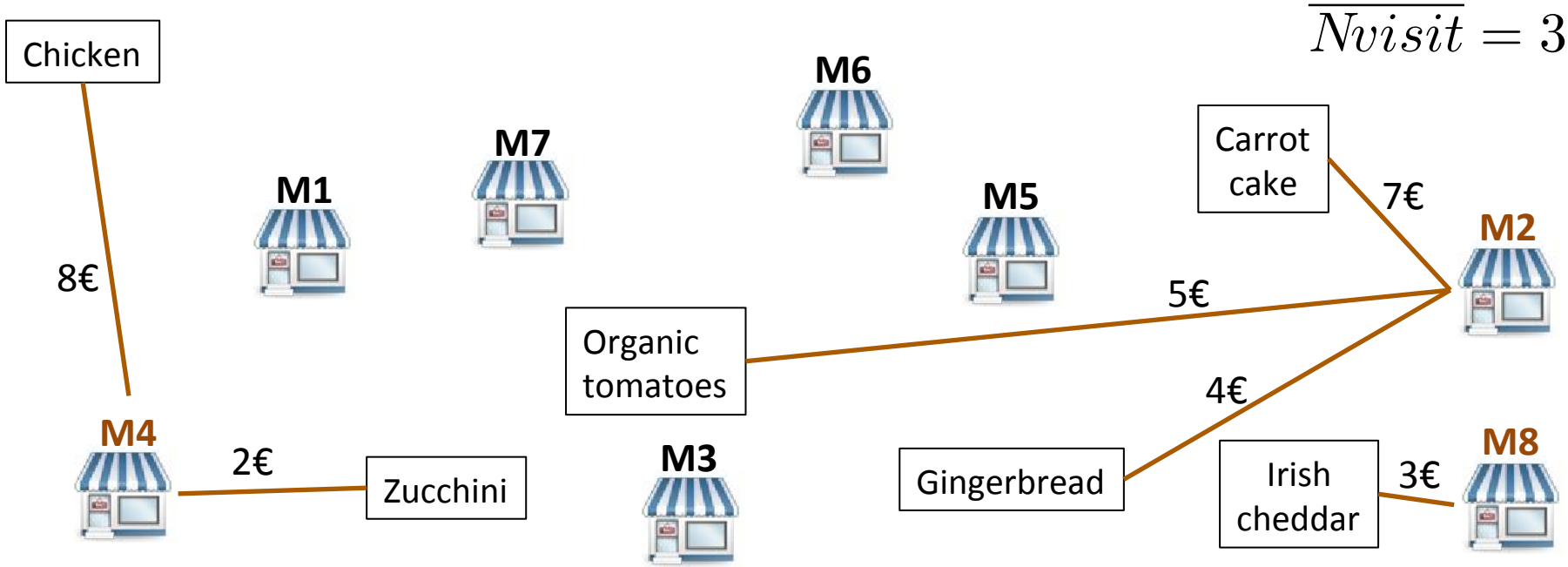


Problem structure 2 : *p-median*

- Look only at **feasibility + shopping cost**
- What is the cheapest way to buy everything in less than $\overline{Nvisit}$ markets ?



p-median Problem



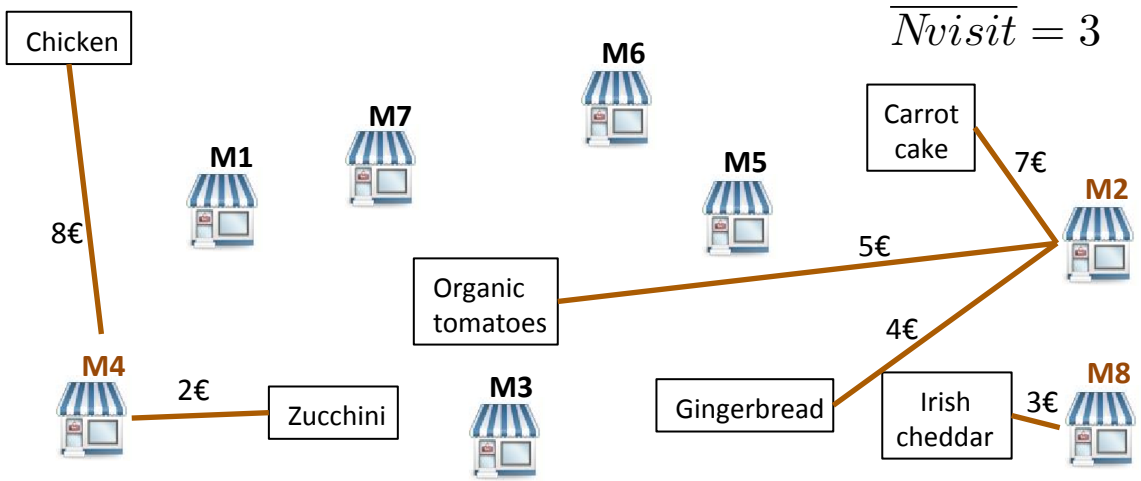
Problem structure 2 : *p-median*

- Look only at **feasibility + shopping cost**
- What is the cheapest way to buy everything in less than $\overline{Nvisit}$ markets ?

p-median Problem

In CP:

AtMostNValue with costs ?



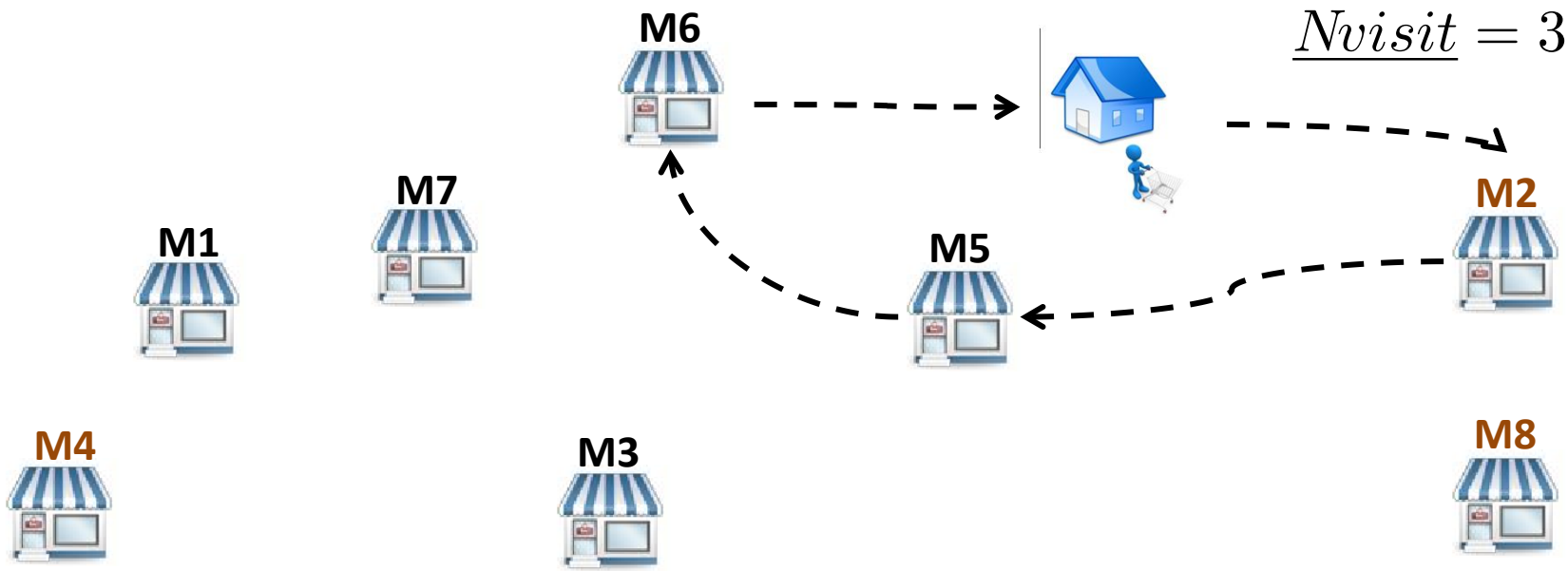
Problem structure 3 : *k-TSP*

- Look only at **traveling cost**
- What is the cheapest way to visit at least $\underline{Nvisit}$ markets ?

Problem structure 3 : *k-TSP*

- Look only at **traveling cost**
- What is the cheapest way to visit at least $\underline{Nvisit}$ markets ?

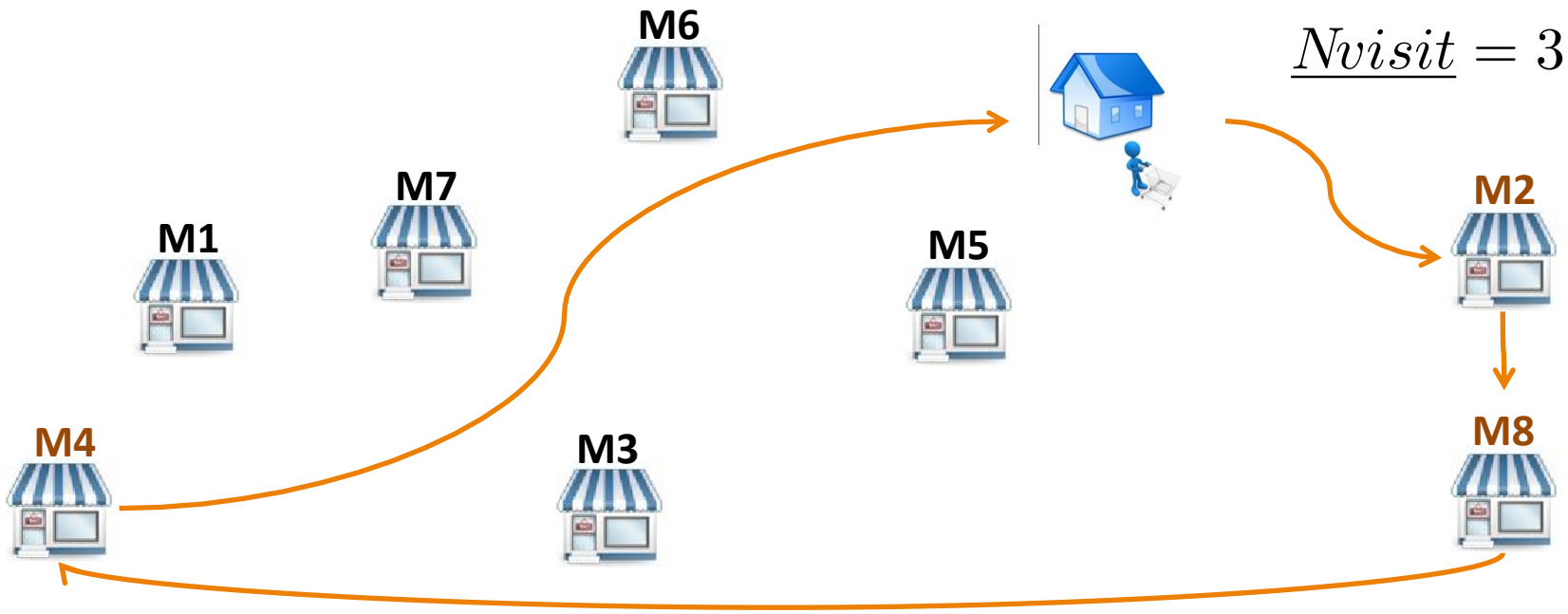
k-TSP problem



Problem structure 3 : *k-TSP*

- Look only at **traveling cost**
- What is the cheapest way to visit at least $\underline{Nvisit}$ markets ?

k-TSP problem



Problem structures

$Nvisit \in \{1, \ldots, B\}$: Number of visited markets

$TotalCost = TravelingCost + ShoppingCost$

Relaxation	Nature of the problem	Value of the parameter	How to solve / propagate it ?	Key propagation
Feasibility	Hitting Set	$\overline{Nvisit}$ (cardinality)		$\underline{Nvisit}$
Feasibility + Shopping cost	p-median	$p = \overline{Nvisit}$		$\frac{ShoppingCost}{\underline{Nvisit}}$
Traveling Cost	k-TSP	$k = \underline{Nvisit}$		$\frac{TravelingCost}{\overline{Nvisit}}$

Problem structures

$Nvisit \in \{1, \ldots, B\}$: Number of visited markets

$TotalCost = TravelingCost + ShoppingCost$

Relaxation	Nature of the problem	Value of the parameter	How to solve / propagate it ?	Key propagation
Feasibility	Hitting Set AtMostNValue	$\overline{Nvisit}$ (cardinality)		$\underline{Nvisit}$
Feasibility + Shopping cost	p-median Weighted-NValue	$p = \overline{Nvisit}$		$\frac{ShoppingCost}{\underline{Nvisit}}$
Traveling Cost	k-TSP Close to Weighted-Circuit	$k = \underline{Nvisit}$		$\frac{TravelingCost}{\overline{Nvisit}}$

Problem structures

$Nvisit \in \{1, \ldots, B\}$: Number of visited markets
$TotalCost = TravelingCost + ShoppingCost$

Relaxation	Nature of the problem	Value of the parameter	How to solve / propagate it ?	Key propagation
Feasibility	Hitting Set AtMostNValue	$\overline{Nvisit}$ (cardinality)		$\underline{Nvisit}$
Feasibility + Shopping cost	p-median Weighted-NValue	$p = \overline{Nvisit}$		$\frac{ShoppingCost}{\underline{Nvisit}}$
Traveling Cost	k-TSP Close to Weighted-Circuit	$k = \underline{Nvisit}$		$\frac{TravelingCost}{\overline{Nvisit}}$

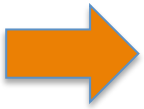
A CP model reveals combinatorial structures of the problem

So far on the TPP

- How to reason about NP-Hard sub-problems involving costs ?

So far on the TPP

- How to reason about NP-Hard sub-problems involving costs ?
- Can CP be competitive with "*advanced linear programming methods*" ?

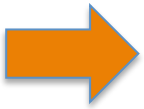


Best known exact method based on Branch and Cut and Price. [G. Laporte, 2003]

So far on the TPP

- How to reason about NP-Hard sub-problems involving costs ?

- Can CP be competitive with "*advanced linear programming methods*" ?



Best known exact method based on Branch and Cut and Price. [G. Laporte, 2003]

- Branch and Cut and Price is the state of the art exact framework for a large class of problems related to routing :

TSP, TSPTW, TPP, TTP, VRP, …

Can we question that ?

Plan

1. Context and motivation
   - Illustrative application: the Traveling Purchaser Problem
   - *Optimization* versus *Satisfaction*
   - *Combinatorial* versus *polyhedral* methods

2. Propagation based on Lagrangian Relaxation
   - Principles of Lagrangian duality
   - Filtering using Lagrangian reduced costs
   - Let's try on the *Nvalue* global constraint

3. Overview of some NP-Hard Constraints with costs
   - *Multi-cost regular, Weighted-circuit, Weighted-Nvalue, Bin-packing with usage costs*

4. Examples of applications

Propagation based on Lagrangian Relaxation

Principles, filtering, Experimentations with *NValue*

2- Lagrangian relaxation

Shortest path with resource constraints

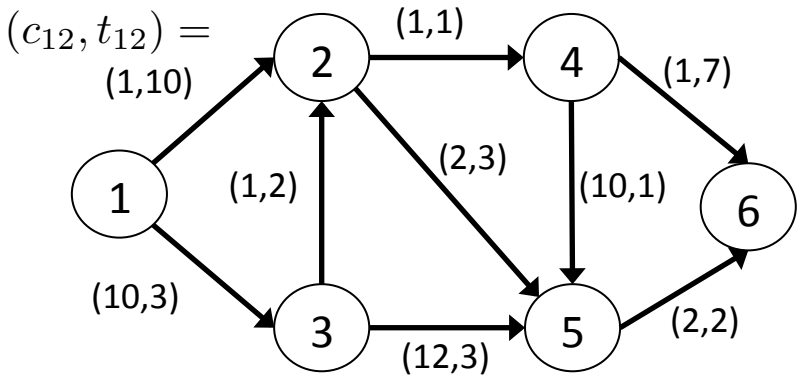
$$\text{Min } z = \sum c_{ij} x_{ij}$$

$$\text{path conservation} \quad (1)$$

$$\sum t_{ij} x_{ij} \leq T \quad (2)$$

$$x_{ij} \in \{0, 1\}$$

P



Simplified example taken from *Network flows* of Ahuja, Magnanti, Orlin

2- Lagrangian relaxation

Shortest path with resource constraints

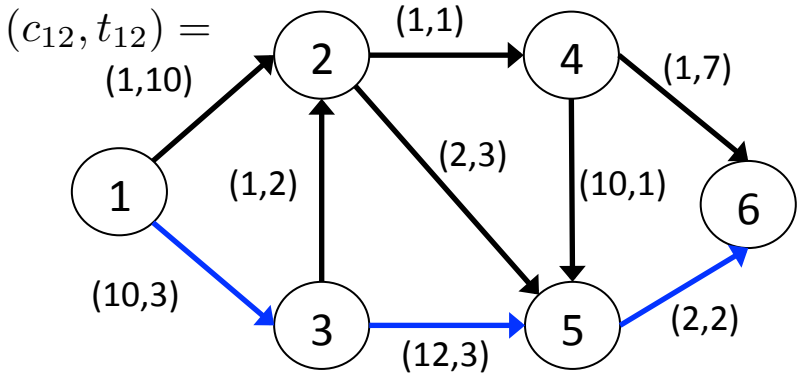
$$\text{Min } z = \sum c_{ij} x_{ij}$$

$$\text{path conservation} \quad (1)$$

$$\sum t_{ij} x_{ij} \leq T \quad (2)$$

$$x_{ij} \in \{0, 1\}$$

P



T = 10

A solution:

$$x_{13} = 1, x_{35} = 1, x_{56} = 1$$
$$z = 10 + 12 + 2 = 24$$
$$time = 3 + 3 + 2 \leq 10$$

2- Lagrangian relaxation

Shortest path with resource constraints

$$\text{Min } z = \sum c_{ij}x_{ij}$$

$$\text{path conservation} \quad (1)$$

$$\sum t_{ij}x_{ij} \leq T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

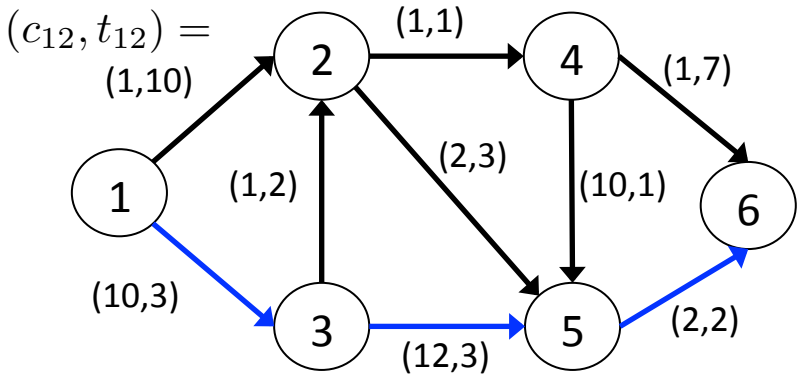
For all $\lambda \geq 0$: **Shortest path**

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

$$\text{path conservation} \quad (1)$$

$$x_{ij} \in \{0,1\}$$

$L(\lambda)$



T = 10

A solution:

$$x_{13} = 1, x_{35} = 1, x_{56} = 1$$
$$z = 10 + 12 + 2 = 24$$
$$time = 3 + 3 + 2 \leq 10$$

2- Lagrangian relaxation

Shortest path with resource constraints

$$\text{Min } z = \sum c_{ij} x_{ij}$$

$$\text{path conservation} \quad (1)$$

$$\sum t_{ij} x_{ij} \leq T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

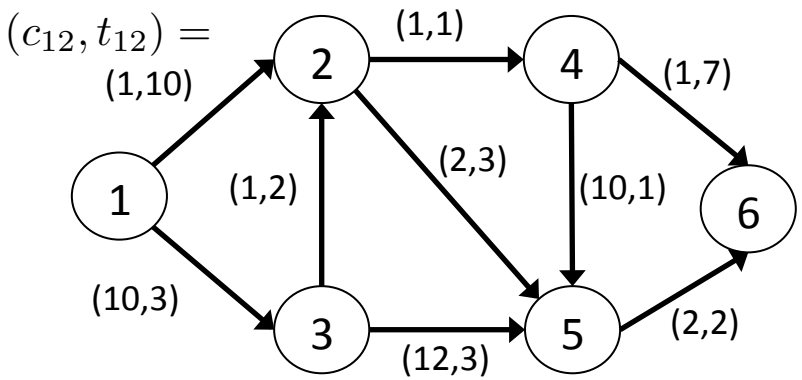
For all $\lambda \geq 0$: **Shortest path**

$$\text{Min } w(\lambda) = \sum c_{ij} x_{ij} - \lambda(T - \sum t_{ij} x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij}) x_{ij} - \lambda T$$

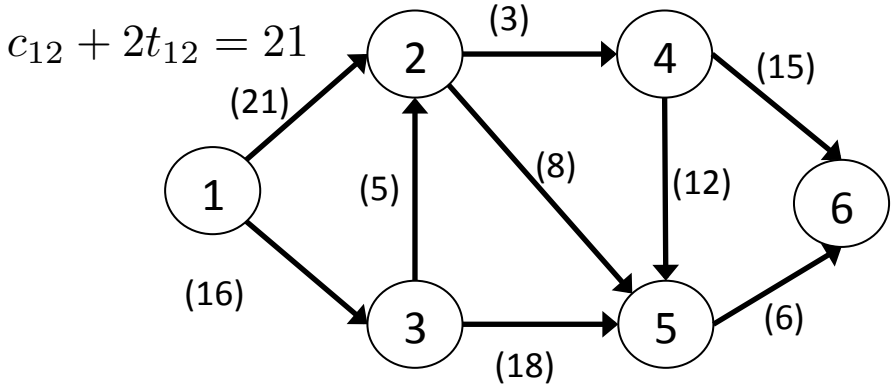
$$\text{path conservation} \quad (1)$$

$$x_{ij} \in \{0,1\}$$

$L(\lambda)$



Lagrangian sub-problem for $\lambda = 2$



2- Lagrangian relaxation

$$\text{Min } z = \sum c_{ij}x_{ij}$$

path conservation (1)

$$\sum t_{ij}x_{ij} \le T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

T = 10

For all $\lambda \ge 0$:

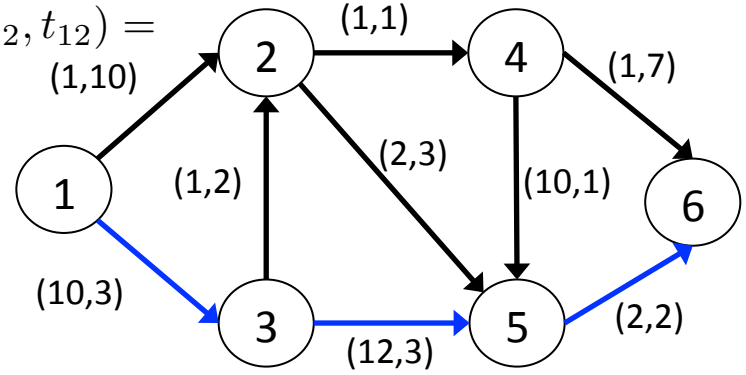
$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

path conservation (1)

$$x_{ij} \in \{0,1\}$$

$L(\lambda)$

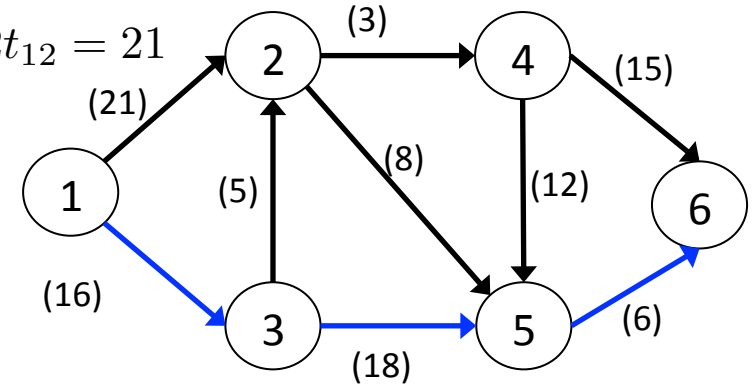
$(c_{12}, t_{12}) =$ (1,10)



$\overline{z} = 10 + 12 + 2 = 24$

$\lambda = 2$

$c_{12} + 2t_{12} = 21$



$\overline{w} = 16 + 18 + 6 - 20 = 20$

For all $\lambda \ge 0$:

Any feasible solution $\overline{x}$ of P is also feasible for $L(\lambda)$ and $\overline{z} \ge \overline{w}(\lambda)$

So we have : $z^* \ge w^*(\lambda)$

2- Lagrangian relaxation

$$\text{Min } z = \sum c_{ij}x_{ij}$$

path conservation (1)

$$\sum t_{ij}x_{ij} \le T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

T = 10

For all $\lambda \ge 0$:

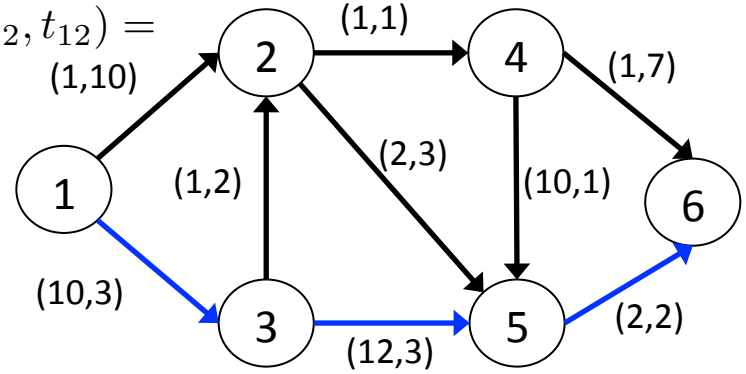
$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

path conservation (1)

$$x_{ij} \in \{0,1\}$$

$L(\lambda)$

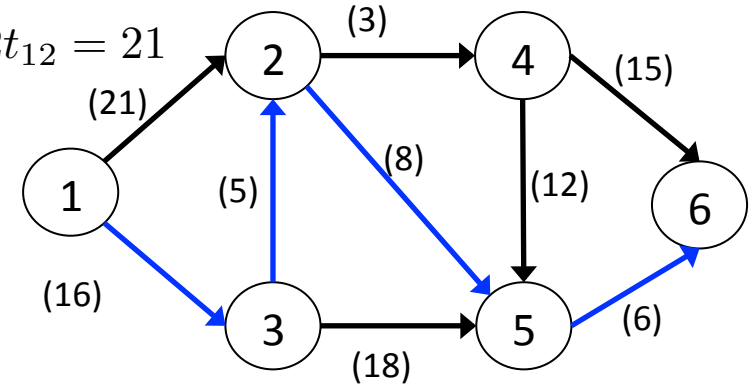
$(c_{12}, t_{12}) =$ (1,10)



$\overline{z} = 10 + 12 + 2 = 24$

$\lambda = 2$

$c_{12} + 2t_{12} = 21$



$w^*(2) = 35 - 20 = 15$

For all $\lambda \ge 0$:

Any feasible solution $\overline{x}$ of P is also feasible for $L(\lambda)$ and $\overline{z} \ge \overline{w}(\lambda)$

So we have : $z^* \ge w^*(\lambda)$

2- Lagrangian relaxation

$$\text{Min } z = \sum c_{ij}x_{ij}$$

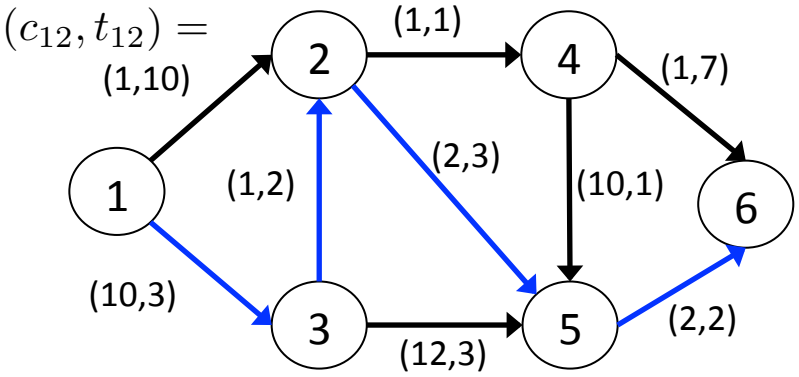
path conservation (1)

$$\sum t_{ij}x_{ij} \le T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

T = 10



$\overline{z} = 15 = z^*$

For all $\lambda \ge 0$:

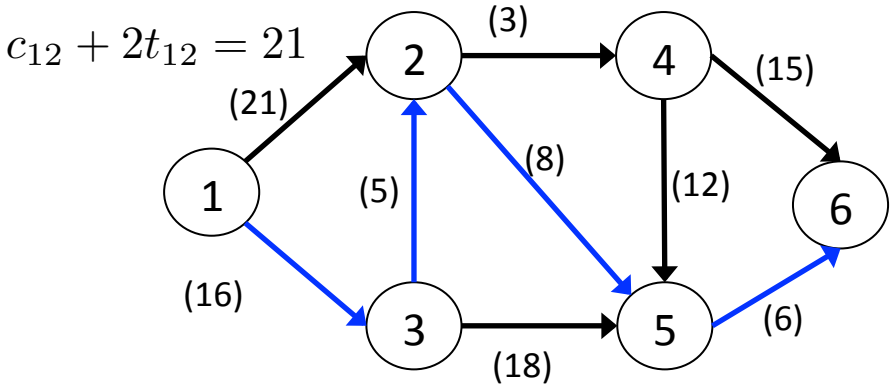
$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

path conservation (1)

$$x_{ij} \in \{0,1\}$$

$L(\lambda)$

$\lambda = 2$



$w^*(2) = 35 - 20 = 15$

For all $\lambda \ge 0$:

Any feasible solution $\overline{x}$ of P is also feasible for $L(\lambda)$ and $\overline{z} \ge \overline{w}(\lambda)$

So we have : $z^* \ge w^*(\lambda)$

2- Lagrangian relaxation

$$\text{Min } z = \sum c_{ij}x_{ij}$$

path conservation (1)

$$\sum t_{ij}x_{ij} \leq T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

For all $\lambda \geq 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

path conservation (1)

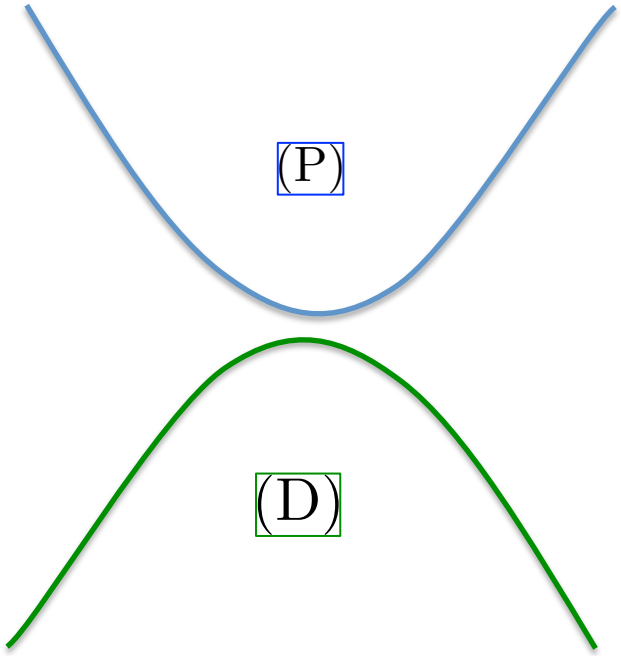
$$x_{ij} \in \{0,1\}$$

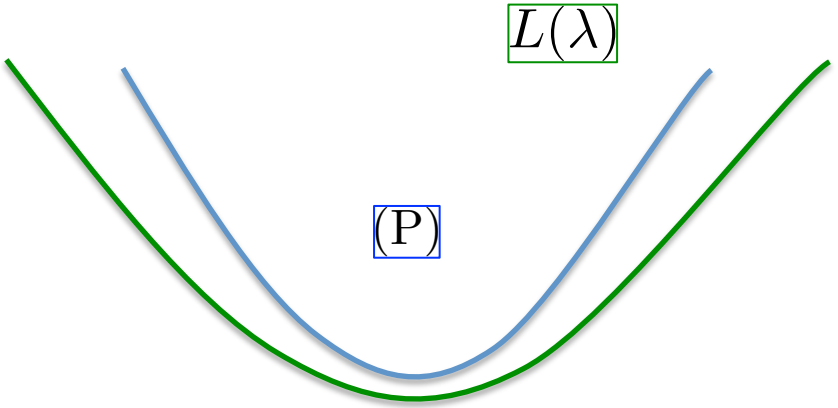
$L(\lambda)$

For all $\lambda \geq 0$:

Any feasible solution $\overline{x}$ of P is also feasible for $L(\lambda)$ and $\overline{z} \geq \overline{w}(\lambda)$

So we have : $z^* \geq w^*(\lambda)$





2- Lagrangian relaxation

$$\text{Min } z = \sum c_{ij}x_{ij}$$

path conservation (1)

$$\sum t_{ij}x_{ij} \le T \quad (2)$$

$$x_{ij} \in \{0,1\}$$

P

For all $\lambda \ge 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

path conservation (1)

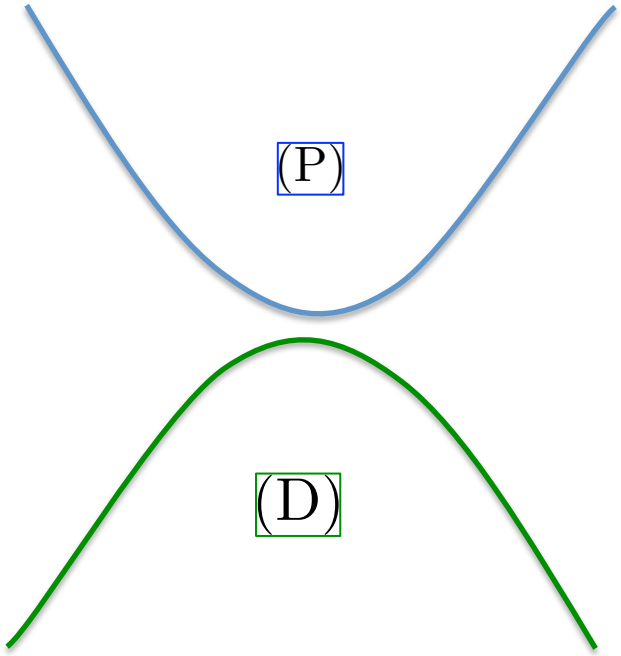
$$x_{ij} \in \{0,1\}$$

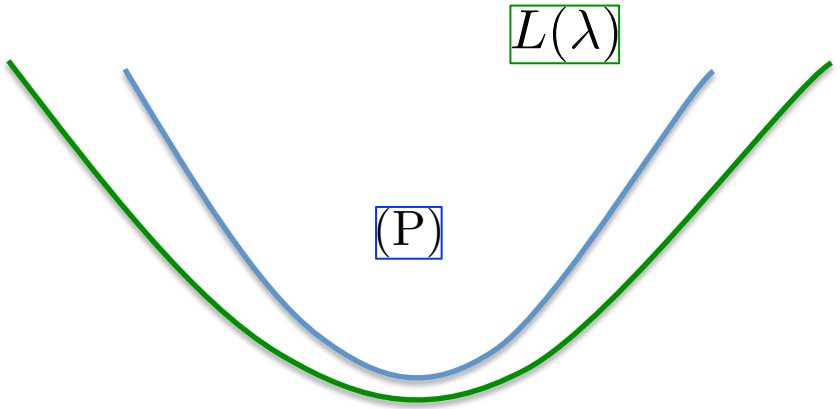
$L(\lambda)$

For all $\lambda \ge 0$:

Any feasible solution $\overline{x}$ of P is also feasible for $L(\lambda)$ and $\overline{z} \ge \overline{w}(\lambda)$

So we have : $z^* \ge w^*(\lambda)$





Lagrangian Dual:

$$L^* = \max_{\lambda \ge 0} w^*(\lambda)$$

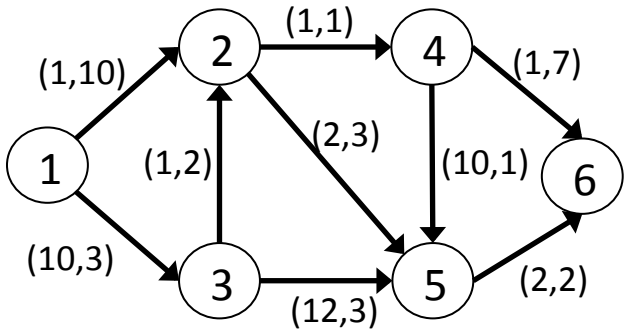
2- Lagrangian relaxation

For all $\lambda \geq 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum(c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$
$$\text{path conservation } (1)$$
$$x_{ij} \in \{0,1\}$$

$L(\lambda)$

$$L^* = \max_{\lambda \geq 0} w^*(\lambda)$$



- *Note that:*
  - Changing λ does not affect the set of feasible solutions of $L(\lambda)$
  - So the cost of given solution of $L(\lambda)$ can be seen as a linear function of λ

2- Lagrangian relaxation

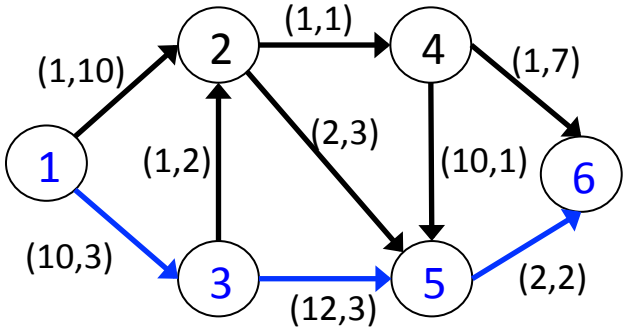
For all $\lambda \geq 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum (c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

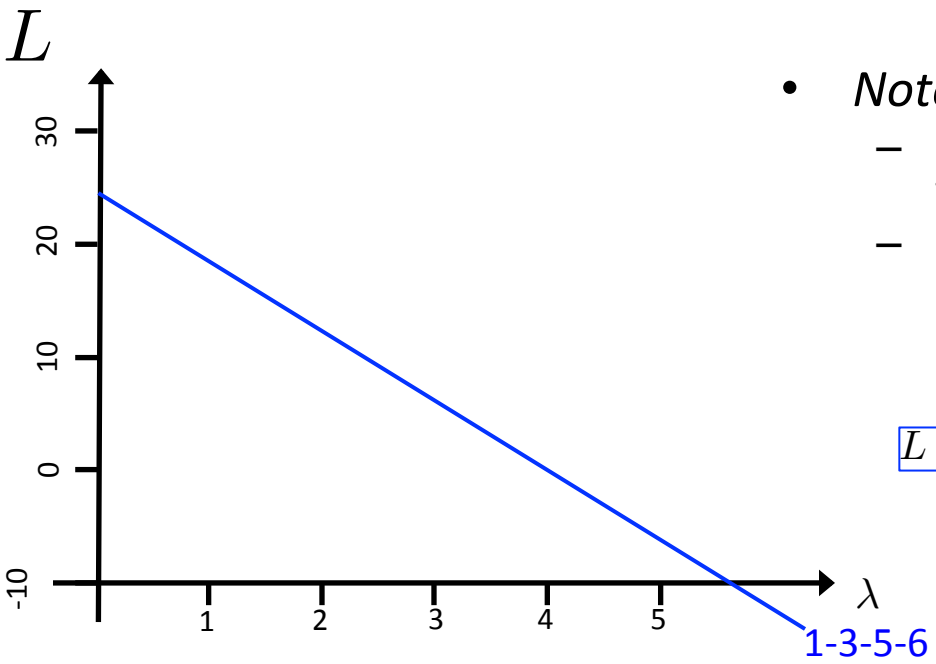
path conservation (1)

$x_{ij} \in \{0,1\}$ $L(\lambda)$

$$L^* = \max_{\lambda \geq 0} w^*(\lambda)$$



T = 14



- *Note:*
  - Changing λ does not affect the set of feasible solutions of $L(\lambda)$
  - So the cost of given solution of $L(\lambda)$ can be seen as a linear function of λ

$$L \leq (10 + 3\lambda) + (12 + 3\lambda) + (2 + 2\lambda) - 14\lambda$$
$$= 24 - 6\lambda \quad \text{(1-3-5-6)}$$

2- Lagrangian relaxation

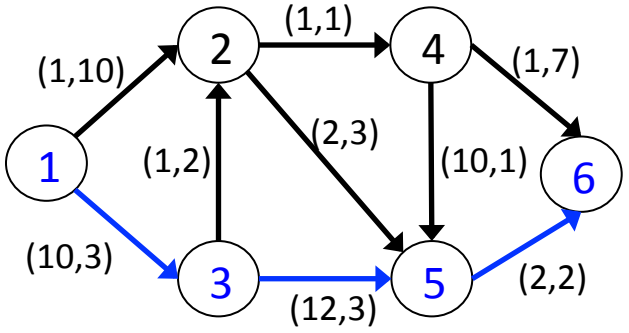
For all $\lambda \geq 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum(c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

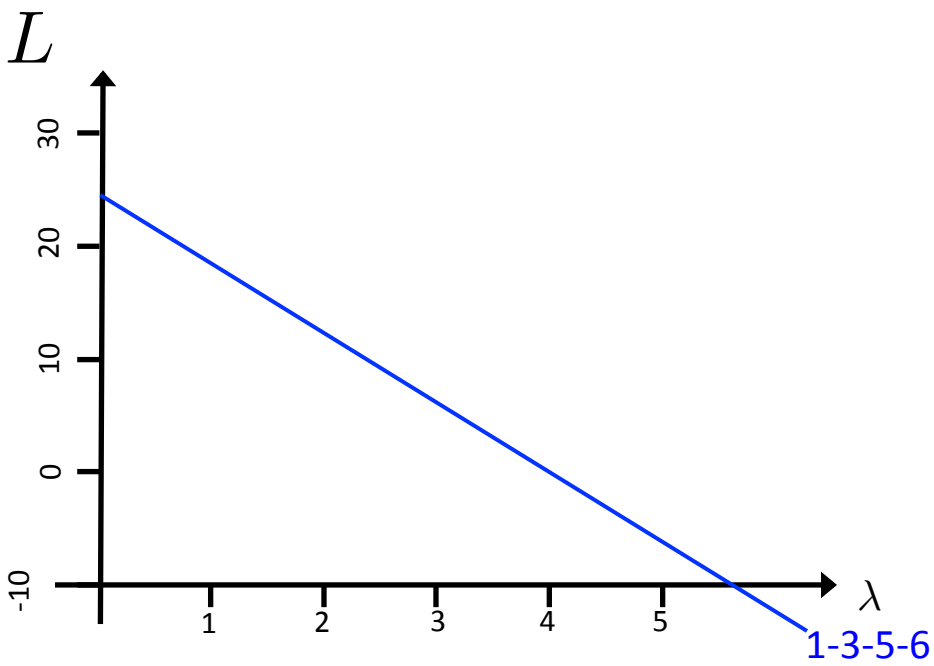
path conservation (1)

$x_{ij} \in \{0,1\}$ $\quad L(\lambda)$

$$L^* = \max_{\lambda \geq 0} w^*(\lambda)$$



T = 14



Max L

$$L \leq (10 + 3\lambda) + (12 + 3\lambda) + (2 + 2\lambda) - 14\lambda$$
$$= 24 - 6\lambda \quad \text{(1-3-5-6)}$$

2- Lagrangian relaxation

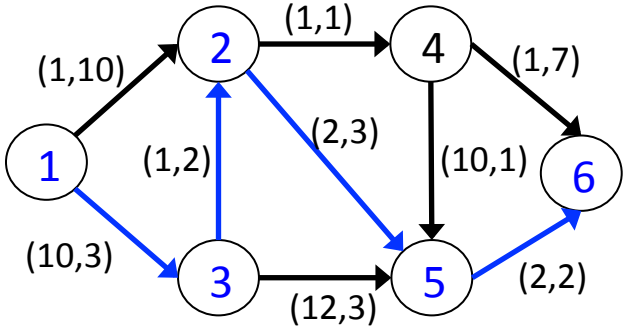
For all $\lambda \geq 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum(c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

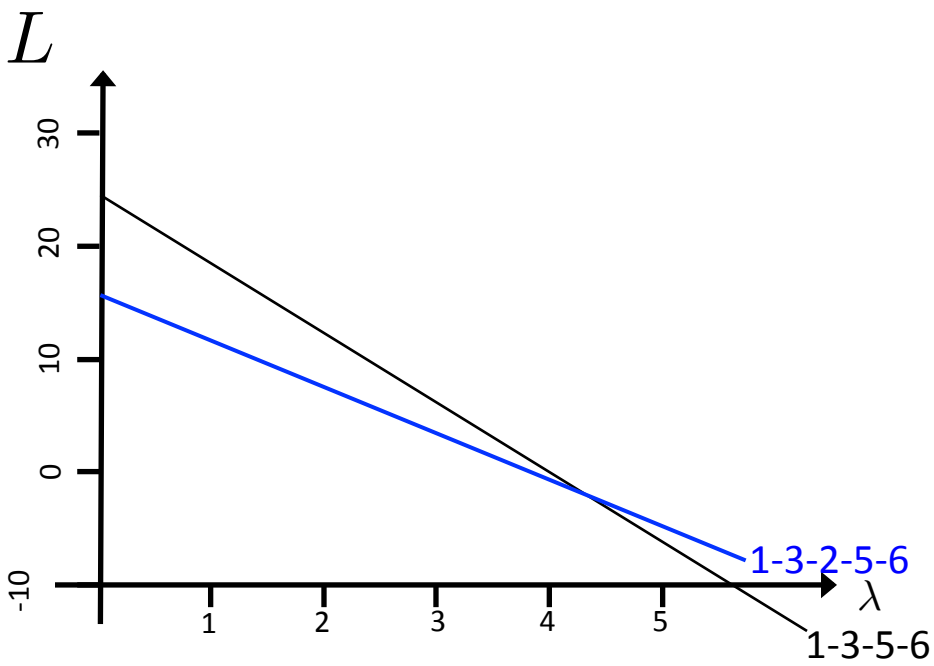
path conservation (1)

$x_{ij} \in \{0,1\}$ $L(\lambda)$

$$L^* = \max_{\lambda \geq 0} w^*(\lambda)$$



T = 14



Max L

$$L \leq (10 + 3\lambda) + (12 + 3\lambda) + (2 + 2\lambda) - 14\lambda$$
$$= 24 - 6\lambda \quad \text{(1-3-5-6)}$$

$L \leq 15 - 4\lambda$ (1-3-2-5-6)

2- Lagrangian relaxation

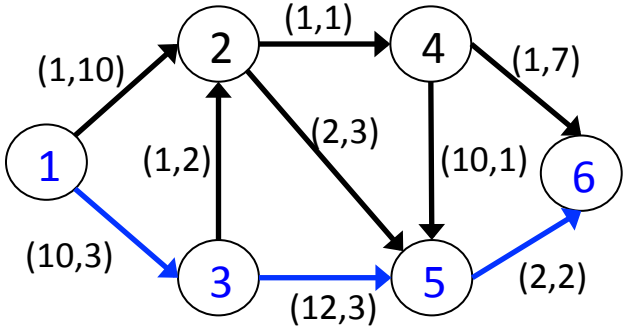
For all $\lambda \geq 0$:

$$\text{Min } w(\lambda) = \sum c_{ij}x_{ij} - \lambda(T - \sum t_{ij}x_{ij})$$
$$= \sum(c_{ij} + \lambda t_{ij})x_{ij} - \lambda T$$

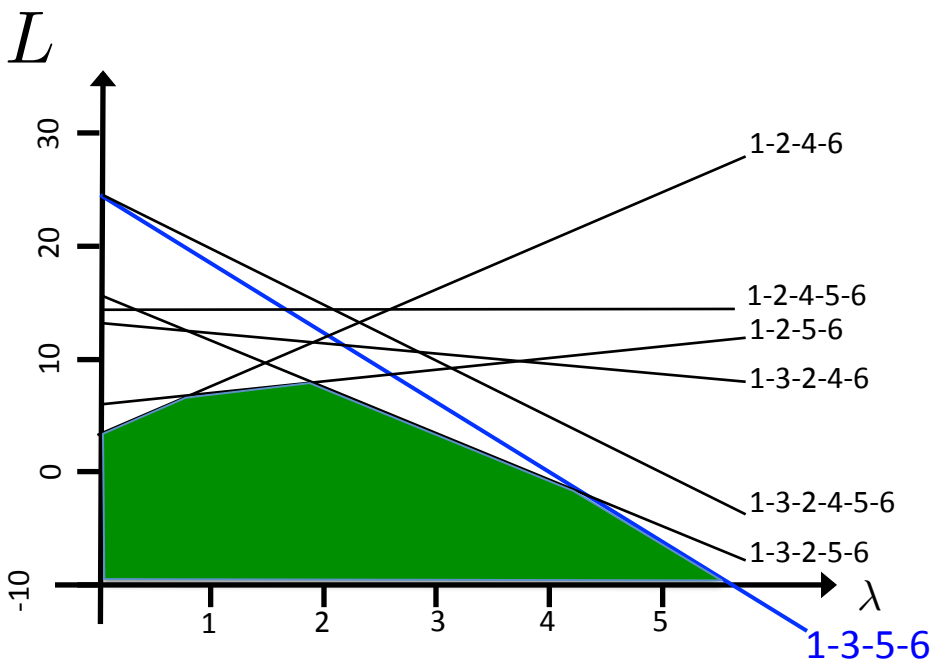
path conservation (1)

$x_{ij} \in \{0,1\}$ $L(\lambda)$

$$L^* = \max_{\lambda \geq 0} w^*(\lambda)$$



T = 14

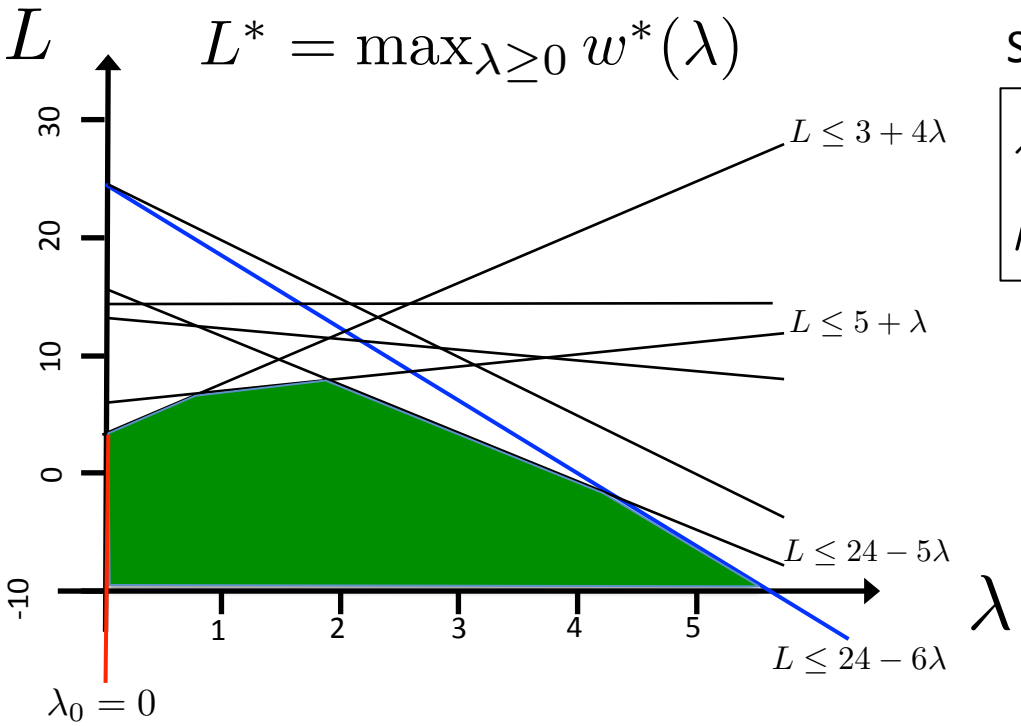


Max L

$L \leq 3 + 4\lambda$ (1-2-4-6)
$L \leq 14$ (1-2-4-5-6)
$L \leq 5 + \lambda$ (1-2-5-6)
$L \leq 13 - \lambda$ (1-3-2-4-6)
$L \leq 24 - 5\lambda$ (1-3-2-4-5-6)
$L \leq 15 - 4\lambda$ (1-3-2-5-6)

$L \leq (10 + 3\lambda) + (12 + 3\lambda) + (2 + 2\lambda) - 14\lambda$
$= 24 - 6\lambda$ (1-3-5-6)

2- Lagrangian relaxation



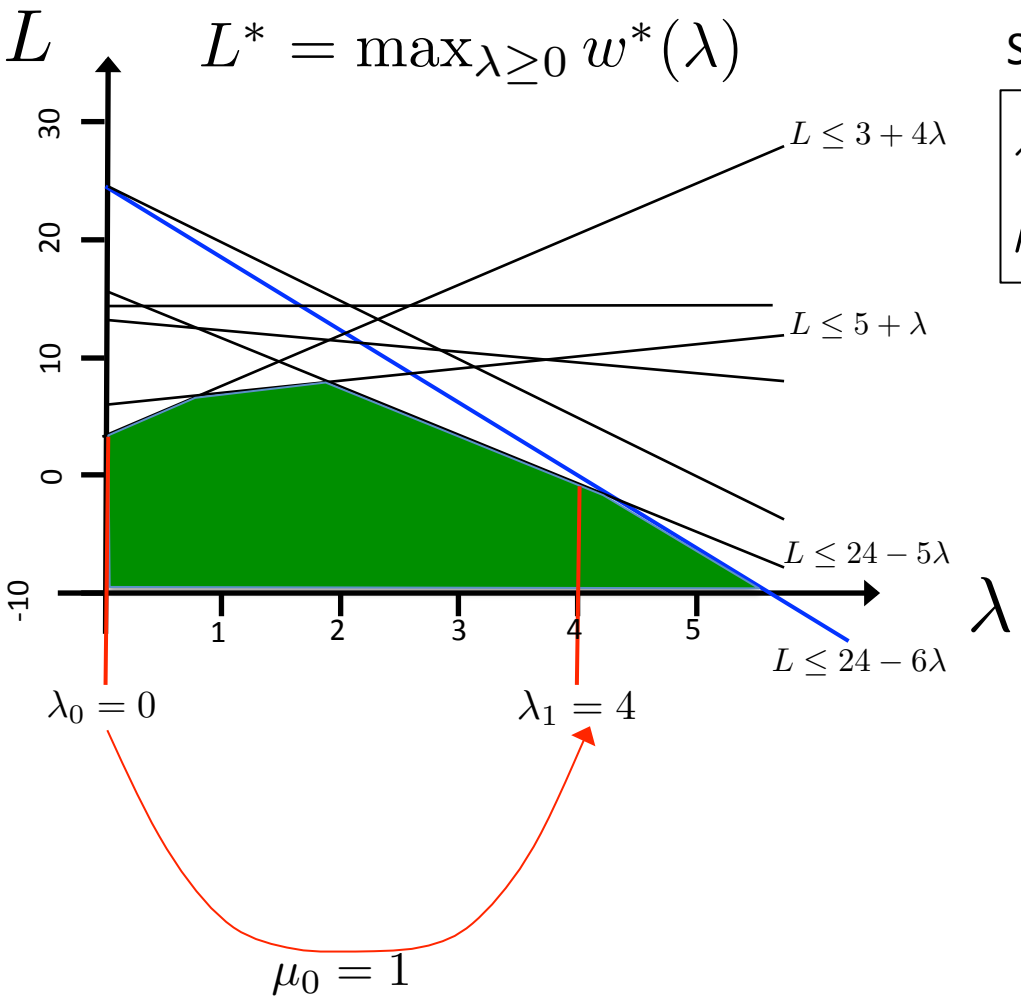
Subgradient algorithm:

$$\lambda_{k+1} \leftarrow \max(0, \lambda_k + \mu(\sum t_{ij} x^k - T))$$

$$\mu_{k+1} = \mu_0 (3/5)^k$$

$$\lambda_0 = 0$$
$$\mu_0 = 1$$

2- Lagrangian relaxation



Subgradient algorithm:

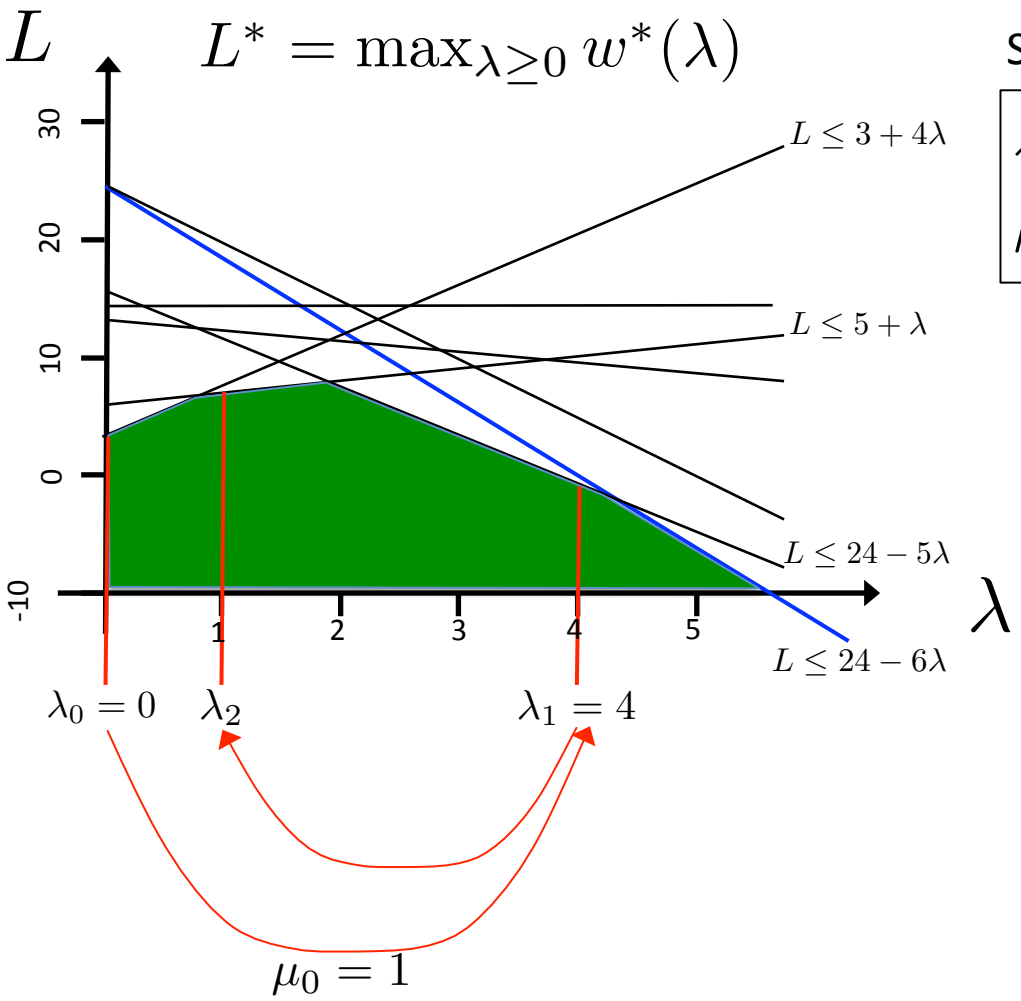
$$\lambda_{k+1} \leftarrow \max(0, \lambda_k + \mu(\sum t_{ij} x^k - T))$$

$$\mu_{k+1} = \mu_0 (3/5)^k$$

$\lambda_0 = 0$
$\mu_0 = 1$

$\lambda_1 = 4$
$\mu_1 = 0.6$

2- Lagrangian relaxation



Subgradient algorithm:

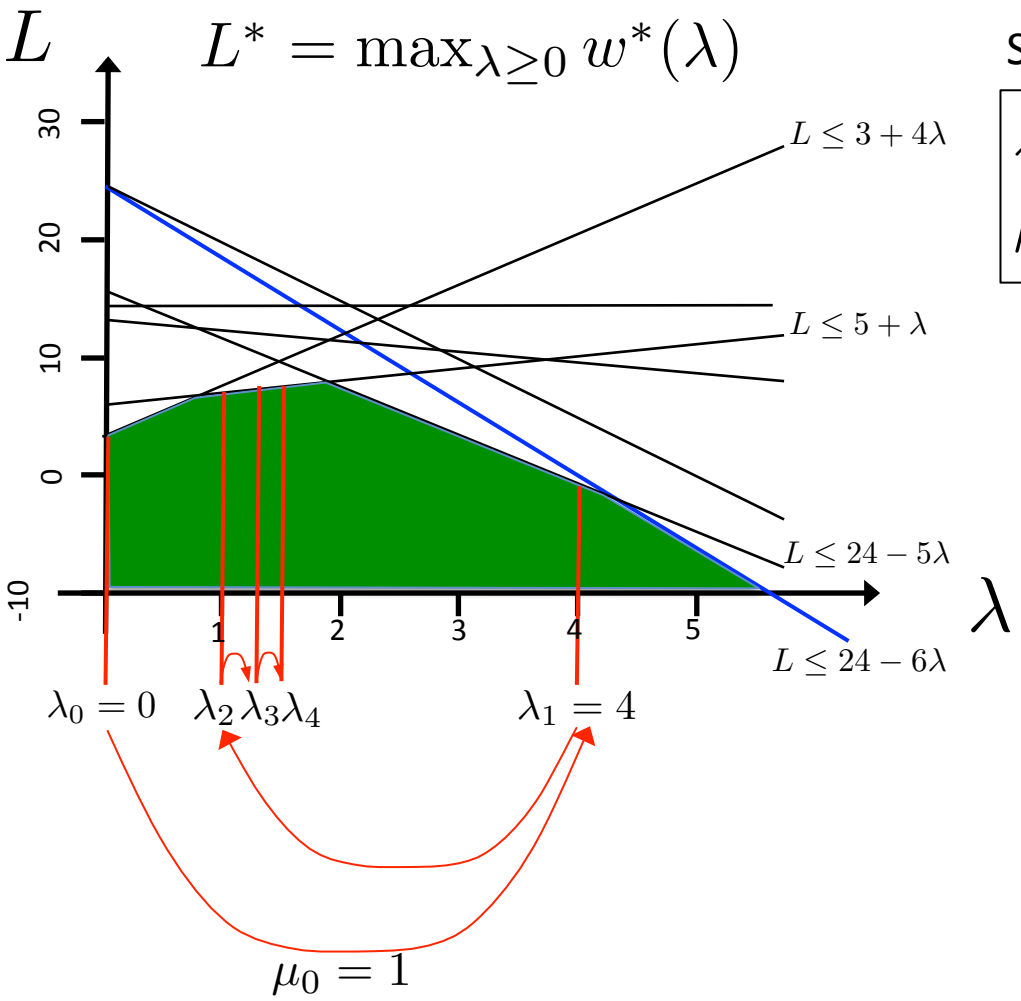
$$\lambda_{k+1} \leftarrow \max(0, \lambda_k + \mu(\sum t_{ij} x^k - T))$$
$$\mu_{k+1} = \mu_0 (3/5)^k$$

$\lambda_0 = 0$
$\mu_0 = 1$

$\lambda_1 = 4$
$\mu_1 = 0.6$

$\lambda_2 = 1$
$\mu_2 = 0.6^2 = 0.36$

2- Lagrangian relaxation



Subgradient algorithm:

$$\lambda_{k+1} \leftarrow \max(0, \lambda_k + \mu(\sum t_{ij} x^k - T))$$
$$\mu_{k+1} = \mu_0 (3/5)^k$$

$\lambda_0 = 0$
$\mu_0 = 1$

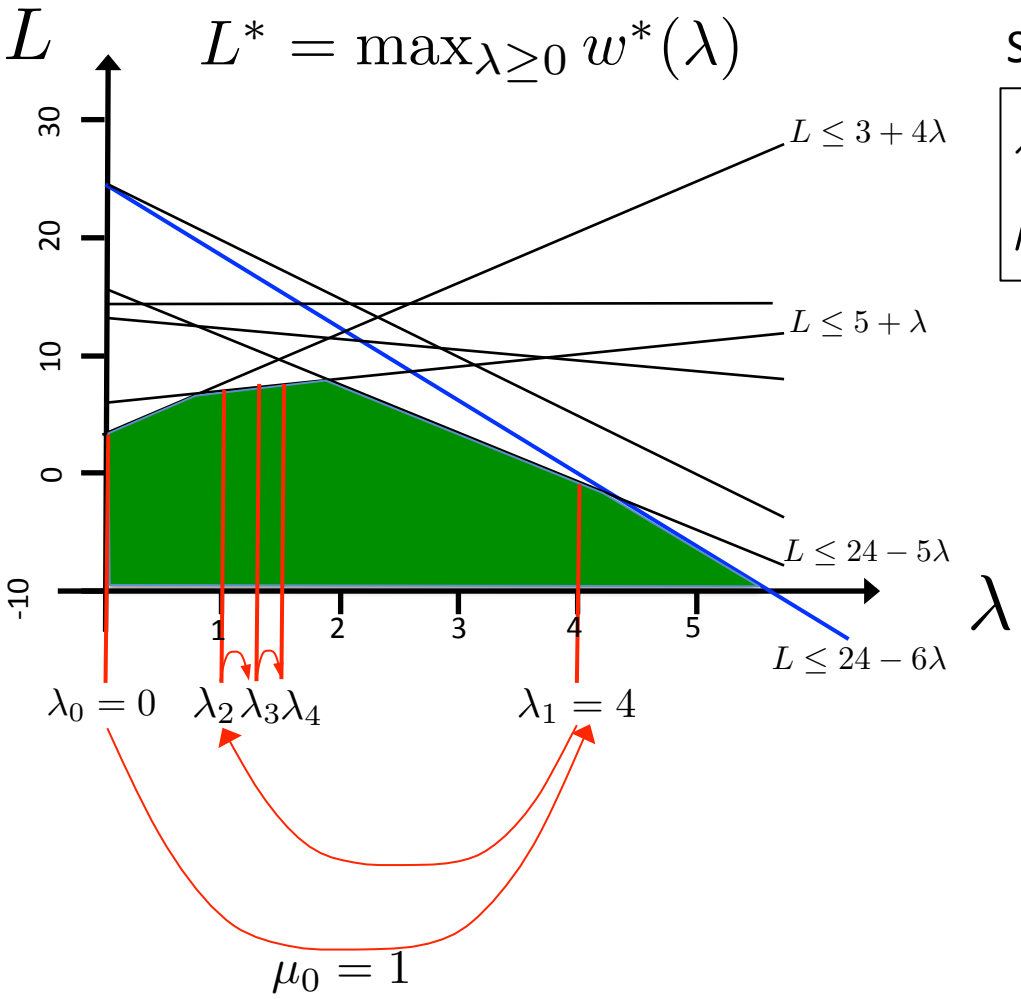
$\lambda_1 = 4$
$\mu_1 = 0.6$

$\lambda_2 = 1$
$\mu_2 = 0.6^2 = 0.36$

$\lambda_3 = 1.36$
$\mu_3 = 0.6^3 = 0.216$

$\lambda_4 = 1.57$
$\cdots$

2- Lagrangian relaxation



Subgradient algorithm:

$$\lambda_{k+1} \leftarrow \max(0, \lambda_k + \mu(\sum t_{ij} x^k - T))$$
$$\mu_{k+1} = \mu_0 (3/5)^k$$

$\lambda_0 = 0$
$\mu_0 = 1$

$\lambda_1 = 4$
$\mu_1 = 0.6$

$\lambda_2 = 1$
$\mu_2 = 0.6^2 = 0.36$

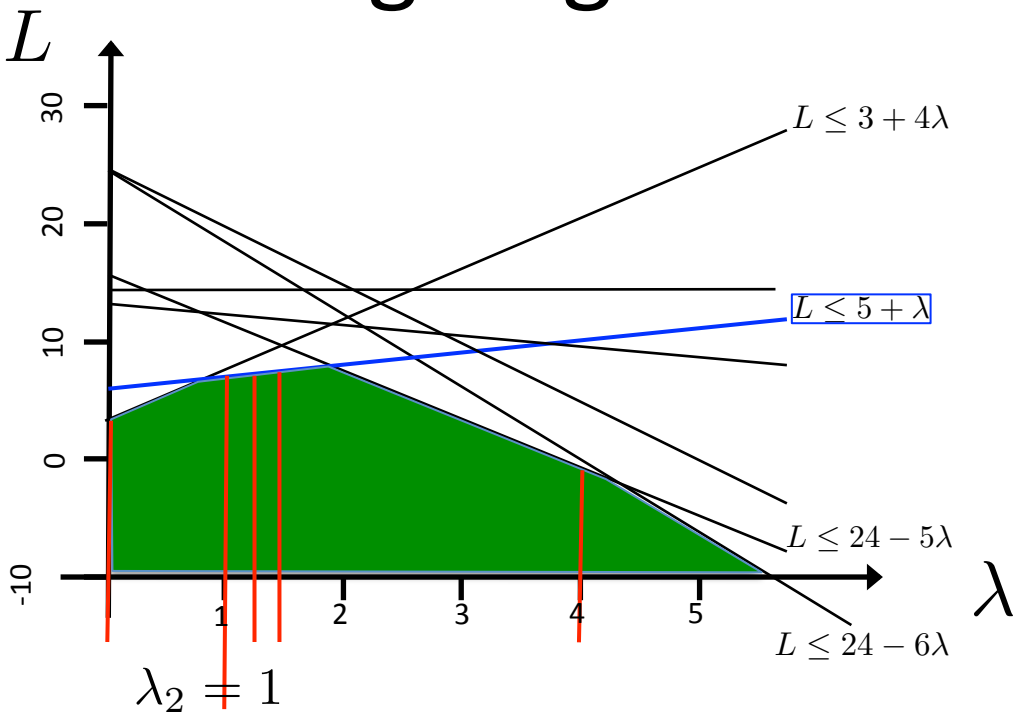
$\lambda_3 = 1.36$
$\mu_3 = 0.6^3 = 0.216$

$\lambda_4 = 1.57$
$\cdots$

To ensure convergence, **we should have**:

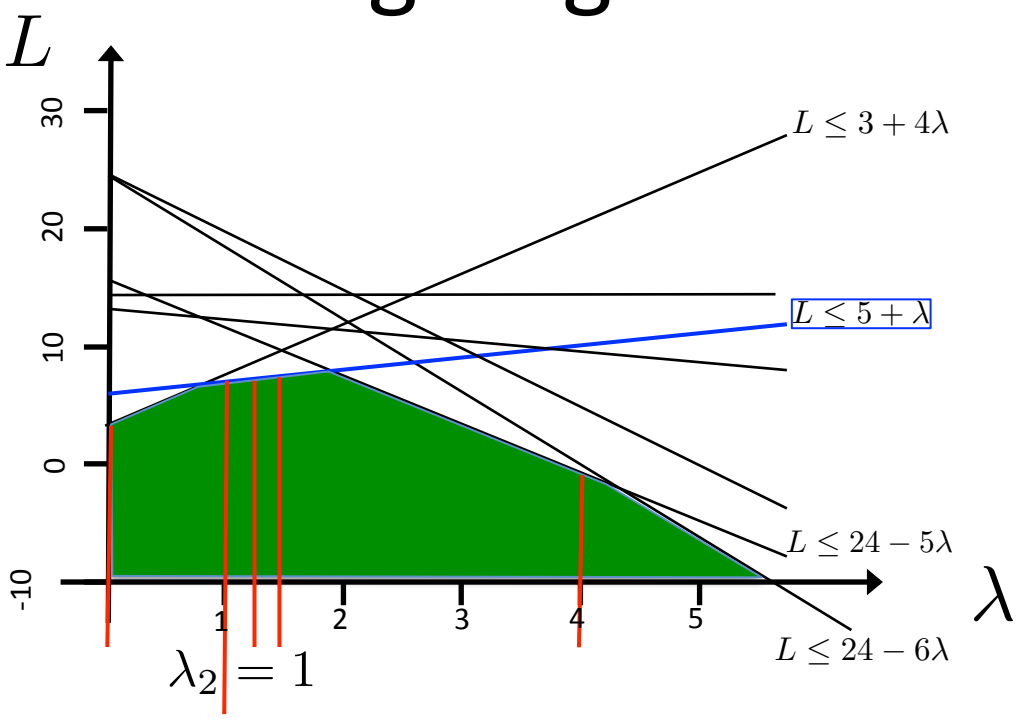
$$\mu_k \to 0 \text{ and } \sum_{j=1}^{k} \mu_j \to \infty$$

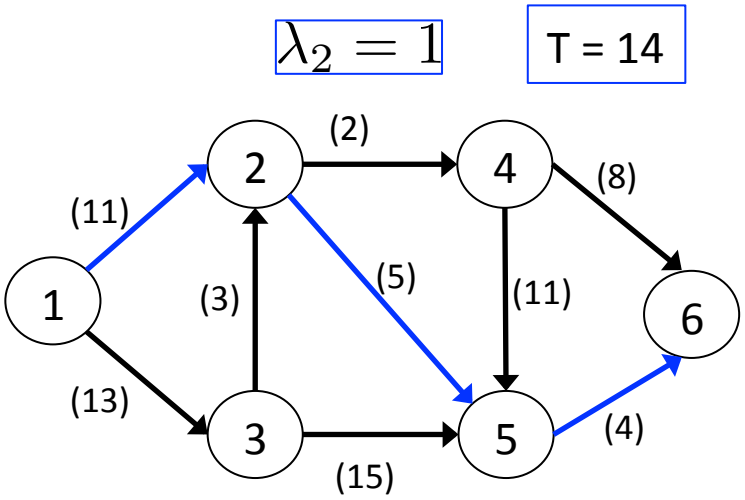
2- Lagrangian relaxation - Filtering



– We can filter at any iteration of this algorithm using the current Lagrangian subproblem and its $w^*(\lambda)$

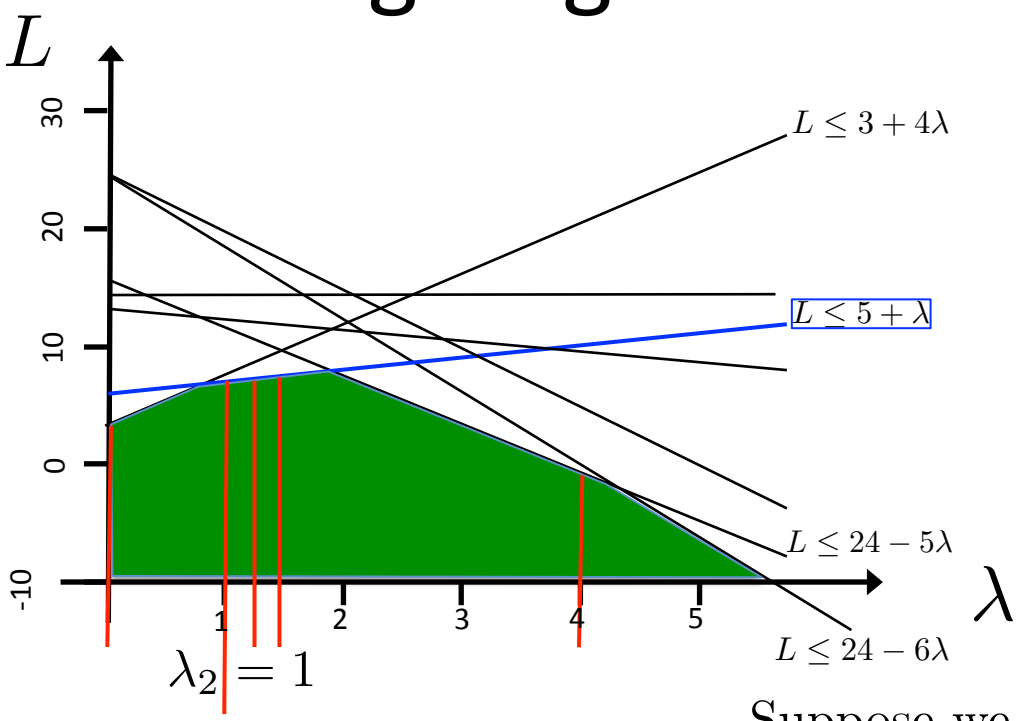
2- Lagrangian relaxation - Filtering





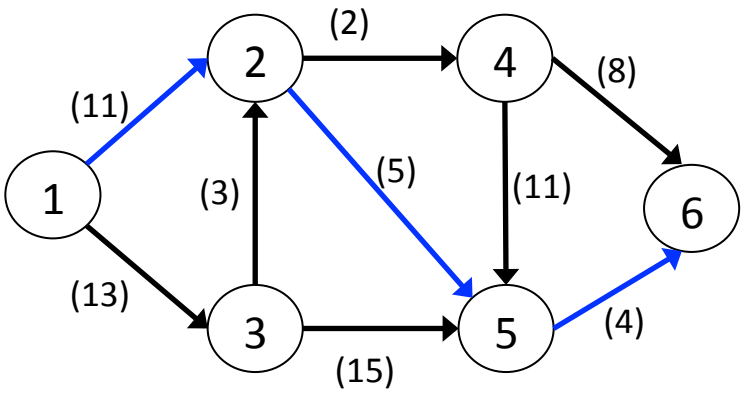
$$w^*(1) = 20 - 14 = 6 \quad \leq z^*$$

2- Lagrangian relaxation - Filtering



$\lambda_2 = 1$

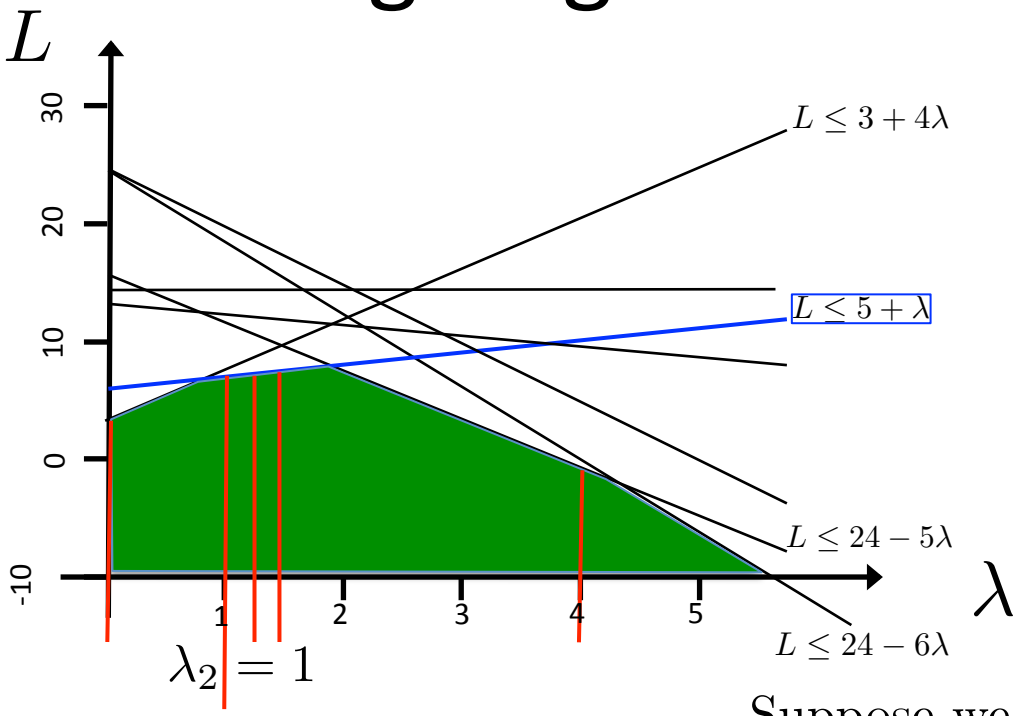
T = 14

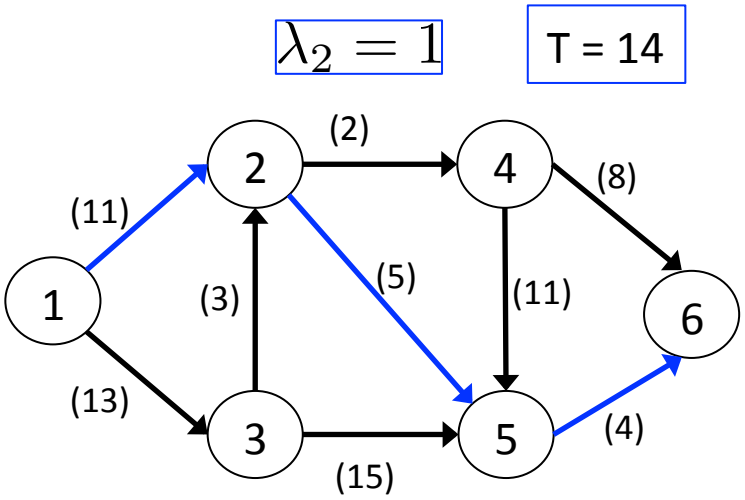


$w^*(1) = 20 - 14 = 6 \quad \leq z^*$

Suppose we know an upper bound of $\overline{z} = 15$

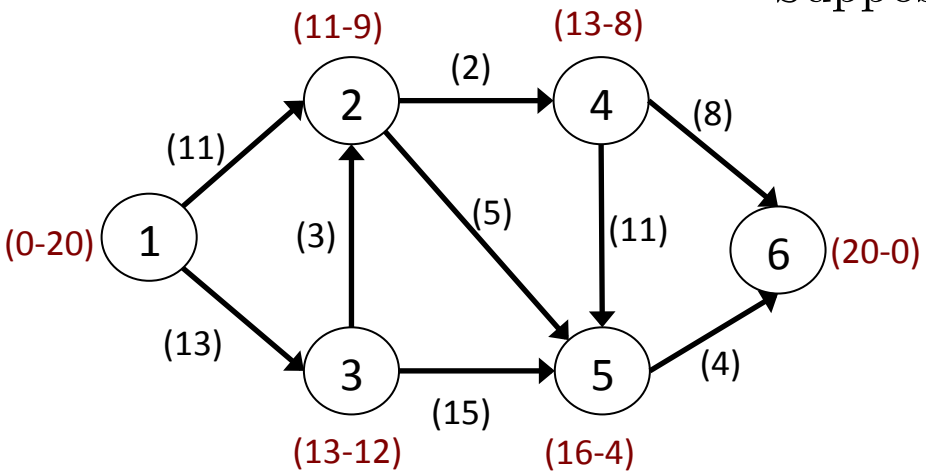
2- Lagrangian relaxation - Filtering





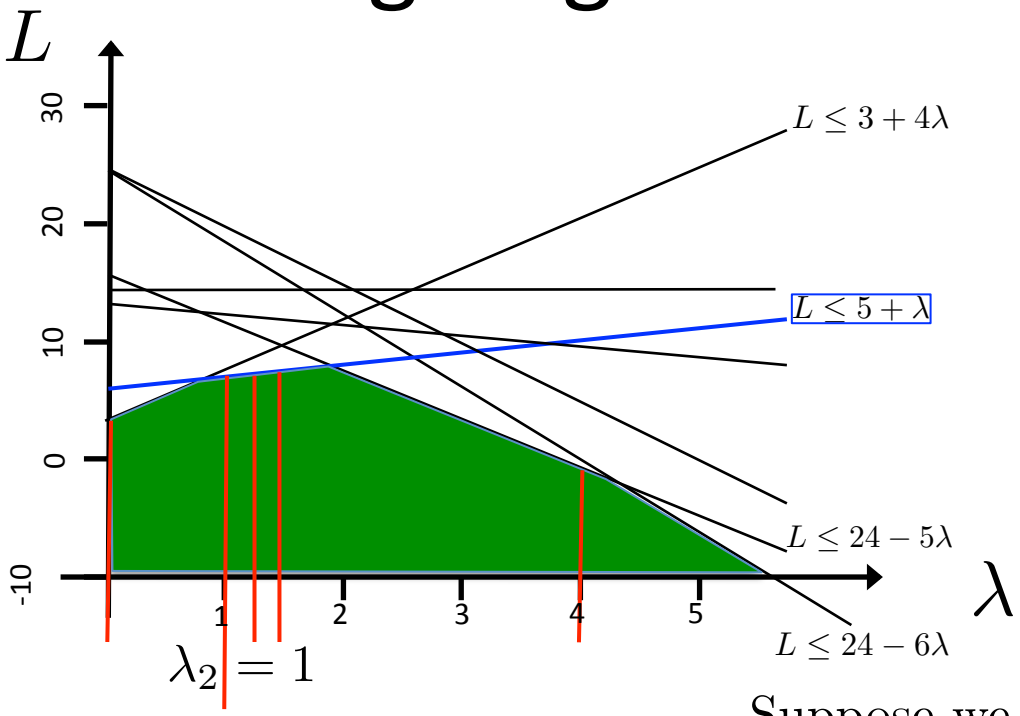
$w^*(1) = 20 - 14 = 6 \quad \leq z^*$

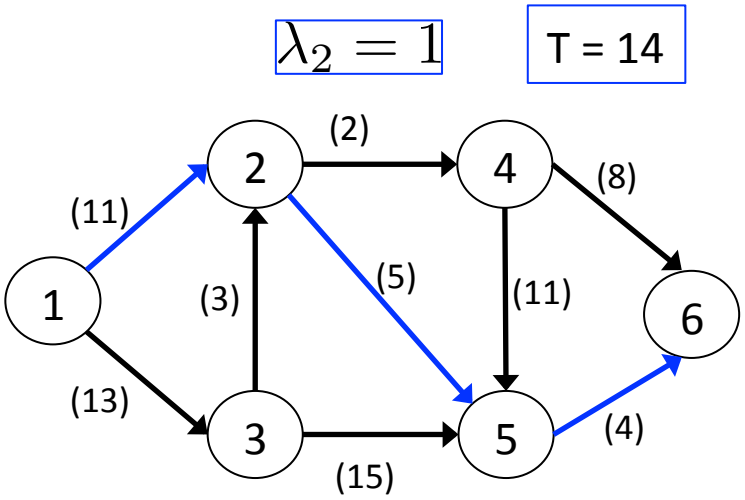
Suppose we know an upper bound of $\overline{z} = 15$



We compute shortest path from source to all other nodes and from all other nodes to sink

2- Lagrangian relaxation - Filtering

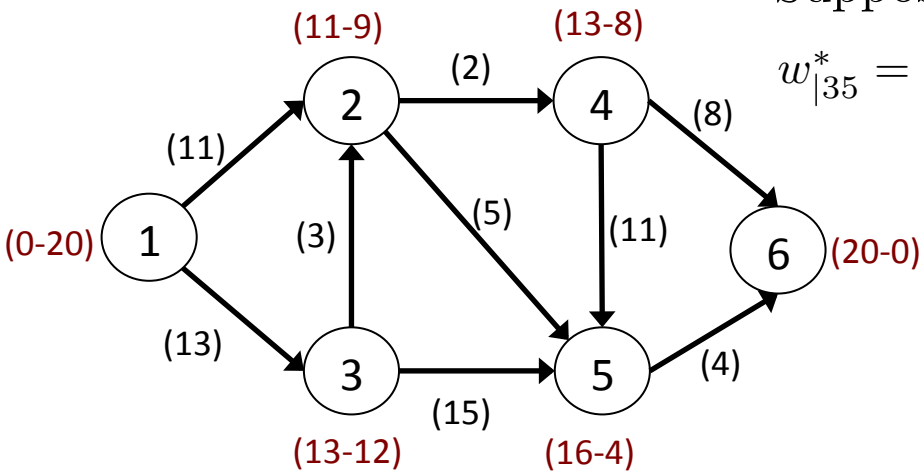




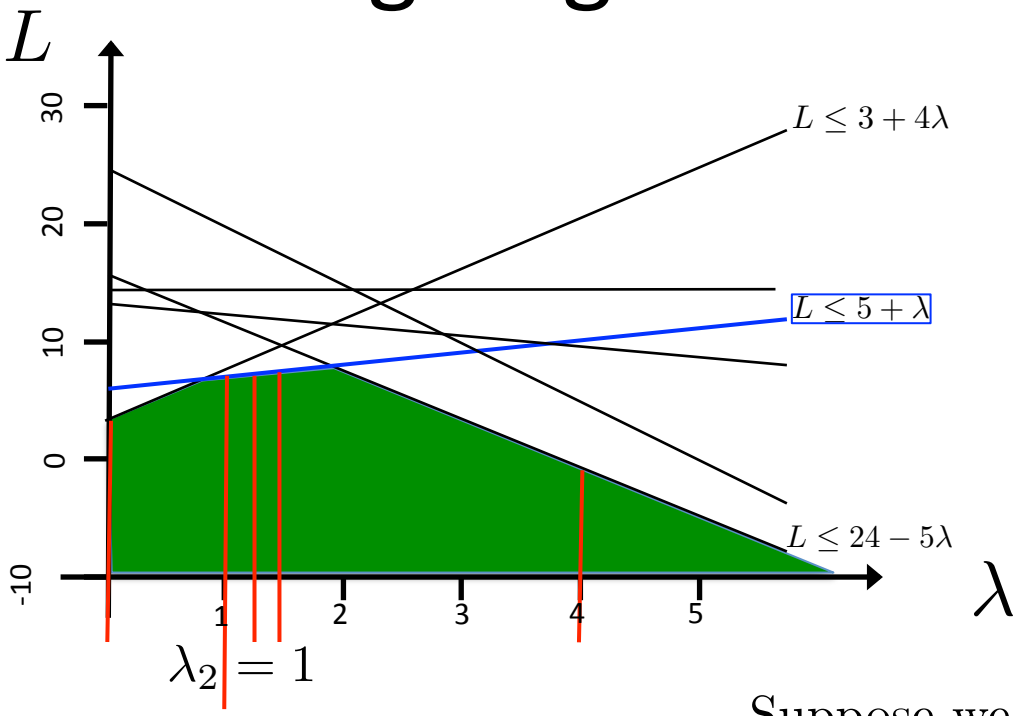
$w^*(1) = 20 - 14 = 6 \quad \leq z^*$

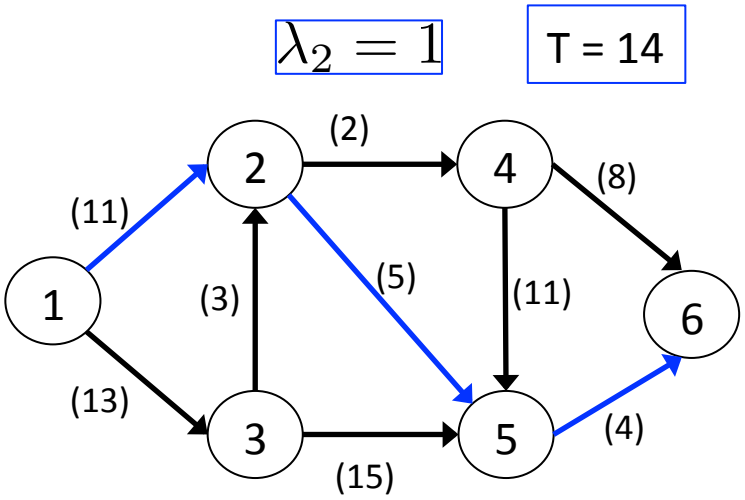
Suppose we know an upper bound of $\overline{z} = 15$

$w^*_{|35} = 13 + (15) + 4 - 14 = 18 > \overline{z} = 15 \Rightarrow x_{35} = 0$



2- Lagrangian relaxation - Filtering

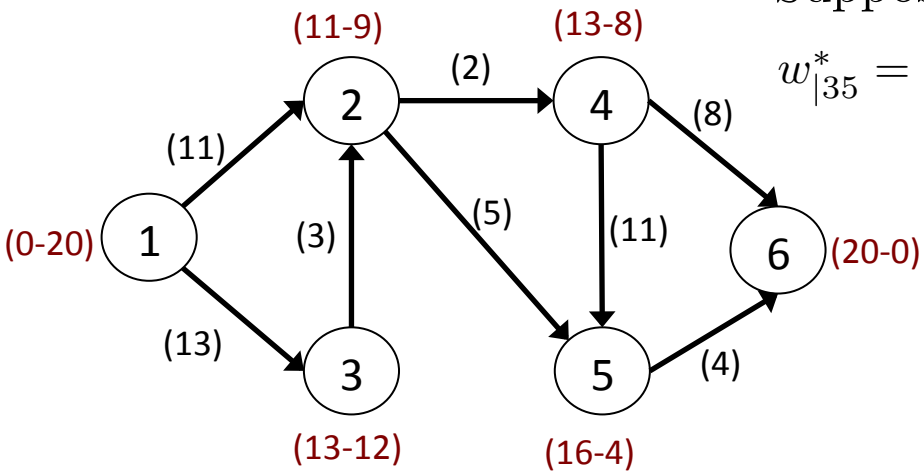




$w^*(1) = 20 - 14 = 6 \quad \leq z^*$

Suppose we know an upper bound of $\overline{z} = 15$

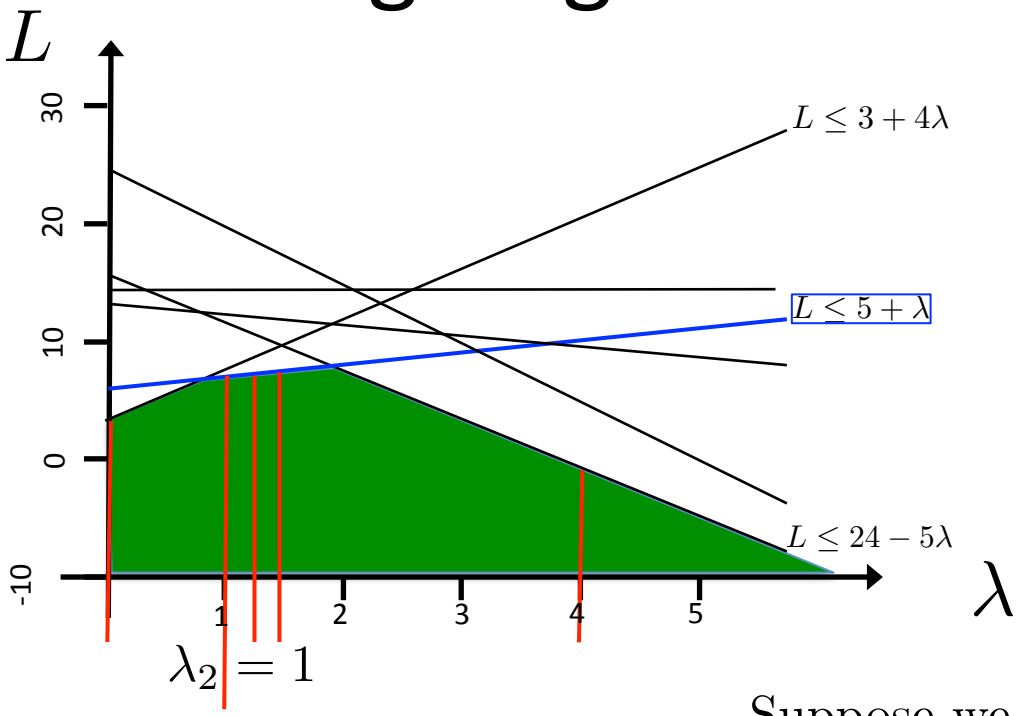
$w^*_{|35} = 13 + (15) + 4 - 14 = 18 > \overline{z} = 15 \Rightarrow x_{35} = 0$



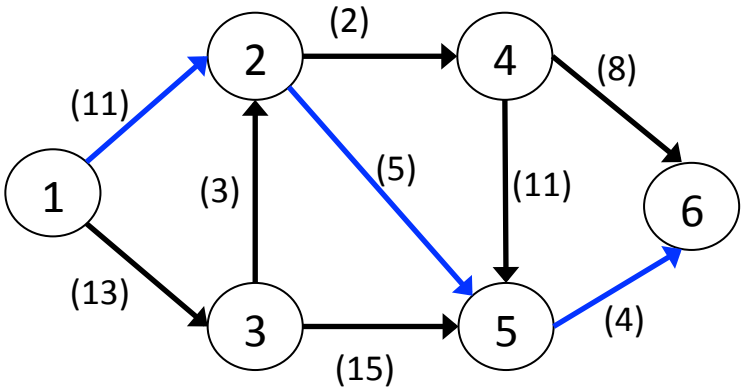
[Sellmann, 2004]

– Lagrangian dual is changed ! does it affect convergence ?

2- Lagrangian relaxation - Filtering



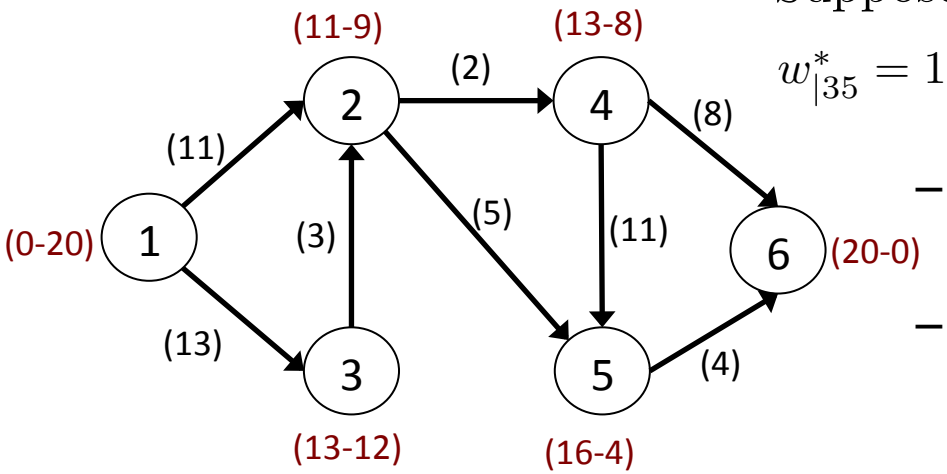
$\lambda_2 = 1$ T = 14



$w^*(1) = 20 - 14 = 6 \quad \leq z^*$

Suppose we know an upper bound of $\overline{z} = 15$

$w^*_{|35} = 13 + (15) + 4 - 14 = 18 > \overline{z} = 15 \Rightarrow x_{35} = 0$



- Lagrangian dual is changed does it affect convergence ? [Sellmann, 2004]
- Filtering takes place near L* most of the time but not necessarily

What values of λ are good for filtering ?

Plan

1. Context and motivation
   - Illustrative application: the Traveling Purchaser Problem
   - *Optimization* versus *Satisfaction*
   - *Combinatorial* versus *polyhedral* methods

2. Propagation based on Lagrangian Relaxation
   - Lagrangian duality
   - Filtering using Lagrangian reduced costs
   - Let's try on the *Nvalue* global constraint

3. Overview of some NP-Hard Constraints with costs
   - *Multi-cost regular, Weighted-circuit, Weighted-Nvalue, Bin-packing with usage costs*

4. Examples of applications

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

- Enforce N to be the number of distinct values appearing in the set X of variables

$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

$\text{NVALUE}(2, [2, 2, 2, 2, 4, 4, 2])$

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

- Enforce N to be the number of distinct values appearing in the set X of variables

$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

$\text{NVALUE}(2, [2, 2, 2, 2, 4, 4, 2])$

$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

– Enforce N to be the number of distinct values appearing in the set X of variables

$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

$\text{NVALUE}(2, [2, 2, 2, 2, 4, 4, 2])$

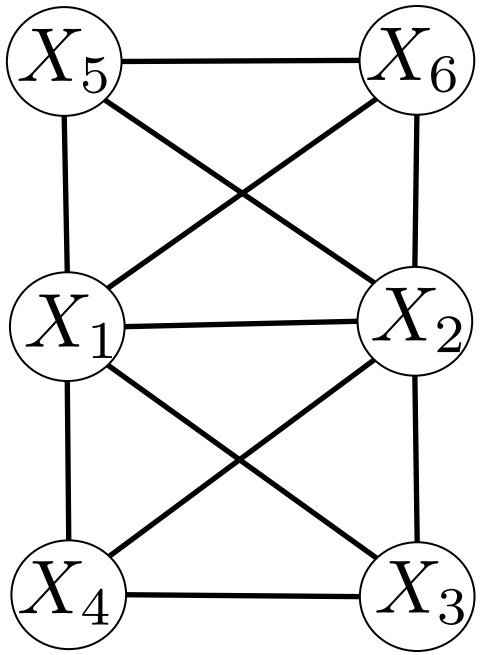
$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

- Enforcing GAC is NP-Hard
- Several lower bounds proposed by [Hebrard et al, 2006]

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

– Enforce N to be the number of distinct values appearing in the set X of variables



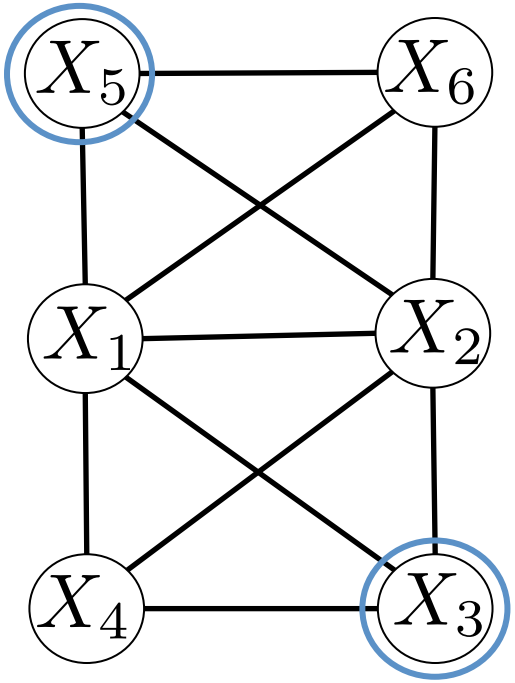
$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

- Enforcing GAC is NP-Hard
- Lower bound of N obtained by a greedy computing an independent set [Hebrard et al, 2006]

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

– Enforce N to be the number of distinct values appearing in the set X of variables



$D(X_1)$ = {1, 2, 3, 4, 5, 6}
$D(X_2)$ = {2, 4}
$D(X_3)$ = {1, 2}
$D(X_4)$ = {1, 2, 3}
$D(X_5)$ = {4,5}
$D(X_6)$ = {4,5}
$D(N)$ = {1, 2}

- Enforcing GAC is NP-Hard
- Lower bound of N obtained by a greedy computing an independent set [Hebrard et al, 2006]

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

- Propagating **a sharp lower bound of N** is NP-Hard
- The best lower bound proposed in [Bessière et al, 2006] is based on LP-relaxation of:

$$\begin{array}{ll} \text{Min } \sum_{i=1}^{m} y_i & \\ \sum_{i \in D(X_j)} y_i \geq 1 & \forall j = 1, \ldots, n \\ y_i \in \{0, 1\} & \forall i \in V \end{array}$$

m: number of values
n: number of variables

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

- Propagating **a sharp lower bound of N** is NP-Hard
- The best lower bound proposed in [Bessière et al, 2006] is based on LP-relaxation of:

$$\begin{aligned} &\text{Min } \textstyle\sum_{i=1}^{m} y_i \\ &\textstyle\sum_{i \in D(X_j)} y_i \geq 1 \qquad \forall j = 1, \ldots, n \\ &y_i \in \{0, 1\} \qquad \forall i \in V \end{aligned}$$

For all $(\lambda_1, \ldots, \lambda_n) \geq 0$

$$\begin{aligned} \text{Min } w_\lambda &= \textstyle\sum_{i=1}^{m} y_i + \sum_{j=1}^{n} \lambda_j (1 - \sum_{i \in D(X_j)} y_i) \\ &= \textstyle\sum_{i=1}^{m} (1 - \sum_{j \mid i \in D(X_j)} \lambda_j) y_i + \sum_{j=1}^{n} \lambda_j \\ y_i &\in \{0, 1\} \qquad \forall i \in V \end{aligned}$$

m: number of values
n: number of variables

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

- Propagating **a sharp lower bound of N** is NP-Hard
- The best lower bound proposed in [Bessière et al, 2006] is based on LP-relaxation of:

$$\begin{aligned} &\text{Min } \textstyle\sum_{i=1}^{m} y_i \\ &\textstyle\sum_{i \in D(X_j)} y_i \geq 1 \qquad \forall j = 1, \ldots, n \\ &\qquad y_i \in \{0,1\} \qquad \forall i \in V \end{aligned}$$

m: number of values
n: number of variables

For all $(\lambda_1, \ldots, \lambda_n) \geq 0$

$$\begin{aligned} &\text{Min } w_\lambda = \textstyle\sum_{i=1}^{m} y_i + \sum_{j=1}^{n} \lambda_j (1 - \sum_{i \in D(X_j)} y_i) \\ &\qquad = \textstyle\sum_{i=1}^{m} (1 - \sum_{j \mid i \in D(X_j)} \lambda_j) y_i + \sum_{j=1}^{n} \lambda_j \\ &\qquad y_i \in \{0,1\} \qquad \forall i \in V \end{aligned}$$

- No constraints in the Lagrangian subproblem
- Easily solved by inspection :

$$\text{Set } y_i \text{ to 1 if } (1 - \textstyle\sum_{j \mid i \in D(X_j)} \lambda_j) < 0$$

- Filtering is also done “for free”

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

- Propagating **a sharp lower bound of N** is NP-Hard
- The best lower bound proposed in [Bessière et al, 2006] is based on LP-relaxation of:

$$\begin{array}{ll} \text{Min } \sum_{i=1}^{m} y_i & \\ \sum_{i \in D(X_j)} y_i \geq 1 & \forall j = 1, \ldots, n \\ y_i \in \{0,1\} & \forall i \in V \end{array}$$

For all $(\lambda_1, \ldots, \lambda_n) \geq 0$

$$\begin{array}{l} \text{Min } w_\lambda = \sum_{i=1}^{m} y_i + \sum_{j=1}^{n} \lambda_j (1 - \sum_{i \in D(X_j)} y_i) \\ \qquad = \sum_{i=1}^{m} (1 - \sum_{j \mid i \in D(X_j)} \lambda_j) y_i + \sum_{j=1}^{n} \lambda_j \\ y_i \in \{0,1\} \qquad \forall i \in V \end{array}$$

m: number of values
n: number of variables

- No constraints in the Lagrangian subproblem
- Easily solved by inspection :

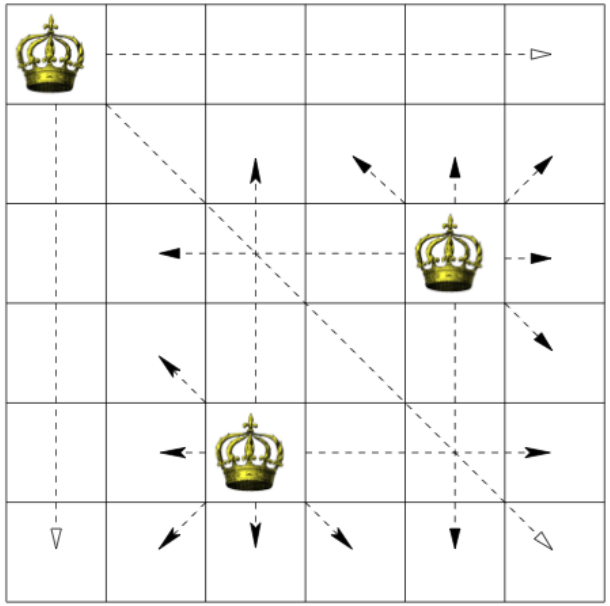
$$\text{Set } y_i \text{ to 1 if } (1 - \sum_{j \mid i \in D(X_j)} \lambda_j) < 0$$

- Filtering is also done "for free"

[Mouthy, Deville, Dooms, JFPC 2007]
A global constraint for the set covering problem

3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

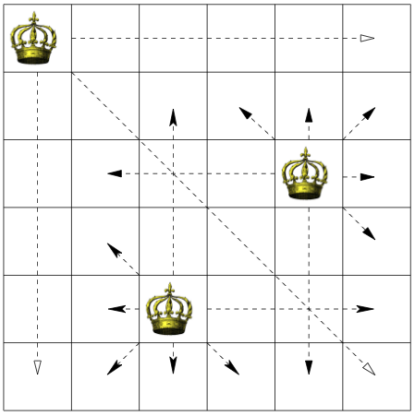


dominating set of queens
(picture from [Hebrard et al, 2006])

$x_i \in S_i \subset \{1, \ldots, n^2\}$: the queen attacking cell i

Minimize z

$$\text{NVALUE}(z, [x_1, \ldots, x_{n^2}]),$$
$$x_i \in S_i \subset \{1, \ldots, n^2\}$$



3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

$precision : 10^{-4}$
$maxIter : 1000$

Solve the Linear relaxation + reduced cost filtering

Greedy bound + filtering

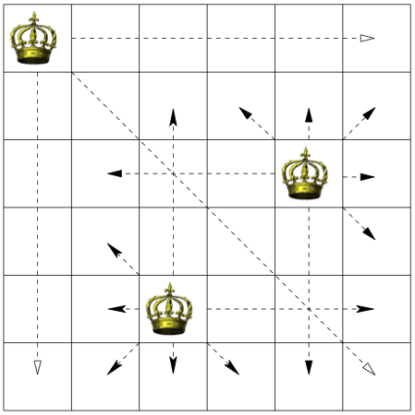
$\mu_k = 1/k$

$\mu_0 = 10^3$
$\mu_k = \mu_0(0.95)^k$

	Q	N	MD		LR1		LR2		LP	
			Back	Time (s)	Back	Time (s)	Back	Time (s)	Back	Time (s)
SAT	6	3	15	**0.01**	**7**	0.15	12	0.1	10	0.4
SAT	7	4	386	**0.13**	**55**	0.6	128	0.3	120	3.5
SAT	8	5	2541	**0.6**	**97**	0.9	233	**0.6**	287	13.5
UNSAT	8	4	1273232	70.5	**1791**	28.9	2948	**9.8**	2656	167
SAT	9	5	1076891	81.4	**862**	15.8	1862	**7.2**	894	123

Branching lexicographic

- LR can be fast (faster than LP)
- LR can filter a more than LP (even if the bound is theoretically the same)



3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

$precision : 10^{-4}$
$maxIter : 1000$

Solve the Linear relaxation + reduced cost filtering

Greedy bound + filtering

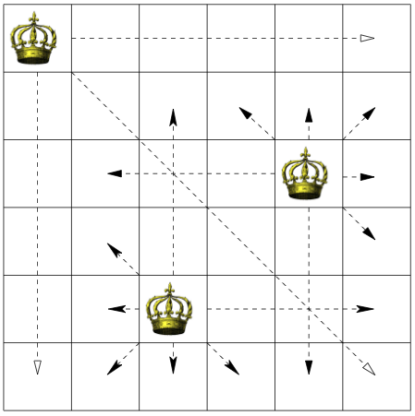
$\mu_k = 1/k$

$\mu_0 = 10^3$
$\mu_k = \mu_0(0.95)^k$

	Q	N	MD		LR1		LR2		LP	
			Back	Time (s)	Back	Time (s)	Back	Time (s)	Back	Time (s)
SAT	6	3	15	**0.01**	**7**	0.15	12	0.1	10	0.4
SAT	7	4	386	**0.13**	**55**	0.6	128	0.3	120	3.5
SAT	8	5	2541	**0.6**	**97**	0.9	233	**0.6**	287	13.5
UNSAT	8	4	2546241	70.5 26.7	**1791**	28.9	2948	**9.8**	2656	167
SAT	9	5	2153565	81.4 22.5	**862**	15.8	1862	**7.2**	894	123

Branching lexicographic

- LR can be fast (faster than LP)
- LR can filter a more than LP (even if the bound is theoretically the same)



3- NValue

$$\text{NVALUE}(N, [X_1, \ldots, X_n])$$

Perform “singleton” filtering

Solve the Linear relaxation + “singleton” filtering

$\mu_0 = 1$
$\mu_k = 1/k$

$\mu_0 = 10^3$
$\mu_k = \mu_0(0.95)^k$

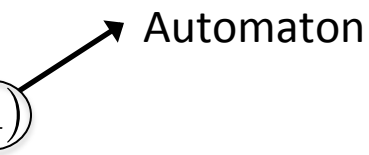
Q	N	MD		Strong LR1		Strong LR2		Strong LP	
		Back	Time (s)	Back	Time (s)	Back	Time (s)	Back	Time (s)
6	3	15	**0.01**	**0**	0.2	**0**	0.1	**0**	0.1
7	4	386	**0.1**	4	4.4	4	0.9	**3**	0.8
8	5	2541	**0.6**	3	8.6	7	2.5	**2**	1.7
8	4	1273232	70.5	**20**	14.2	21	**4.2**	**20**	5.1
9	5	1076891	81.4	**5**	18.2	**5**	**4.3**	**5**	5.7

- Instead of using Lagrangian/linear reduced costs, we fix the assignment and recompute the bound in a “singleton” manner
- LP has a better incremental behaviour

3- Multi-cost regular

- Regular : $\textsc{Regular}([X_1, \ldots, X_n], A)$ [Pesant, 2004]
  - Propagation based on breath-first-search in the unfolded automaton

3- Multi-cost regular



- Regular : $\textsc{Regular}([X_1, \ldots, X_n], A)$ [Pesant, 2004]
  - Propagation based on breath-first-search in the unfolded automaton

3- Multi-cost regular

- Regular : $\textsc{Regular}([X_1, \ldots, X_n], A)$ [Pesant, 2004]
  - Propagation based on breath-first-search in the unfolded automaton

- Cost regular : $\textsc{Regular}([X_1, \ldots, X_n], A) \wedge \sum_{i=1}^{n} c_{iX_i} = Z$
  - Propagation based on shortest/longest path in the unfolded automaton [Demassey, Pesant, Rousseau, 2004]

3- Multi-cost regular

- Regular : $\textsc{Regular}([X_1, \ldots, X_n], A)$ [Pesant, 2004]
  - Propagation based on breath-first-search in the unfolded automaton

- Cost regular : $\textsc{Regular}([X_1, \ldots, X_n], A) \wedge \sum_{i=1}^{n} c_{iX_i} = Z$
  - Propagation based on shortest/longest path in the unfolded automaton [Demassey, Pesant, Rousseau, 2004]

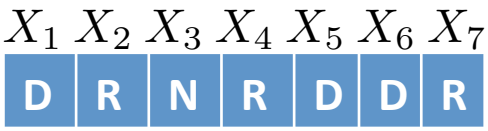
- Multi-cost regular : $\textsc{Multi-cost Regular}([X_1, \ldots, X_n], [Z^1, \ldots, Z^R], A)$
  $\textsc{Regular}([X_1, \ldots, X_n], A) \wedge (\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R)$
  - Propagation based on **resource constrained shortest/longest path**
  - Sequencing and counting at the same time [Menana, Demassey, 2009]
    - Personnel scheduling
    - Routing
  - Example: combine Regular and GCC

3- Multi-cost regular

- Multi-cost regular :

$$\textsc{Regular}([X_1, \ldots, X_n], A) \wedge (\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R)$$

- Example:
  - Schedule 7 shifts of type: **night (N), day (D), rest (R)**
  - (1) "A **Rest must follow a Night** shift"
  - (2) "**Exactly 3 day shifts** and **1 night shift** must take place in the week"



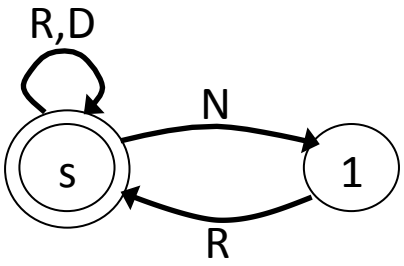
3- Multi-cost regular

- Multi-cost regular :

$$\textsc{Regular}([X_1, \ldots, X_n], A) \wedge (\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R)$$

- Example:
  - Schedule 7 shifts of type: **night (N), day (D), rest (R)**
  - (1) "A **Rest must follow a Night** shift"
  - (2) "**Exactly 3 day shifts** and **1 night shift** must take place in the week"

X_1	X_2	X_3	X_4	X_5	X_6	X_7
D	R	N	R	D	D	R



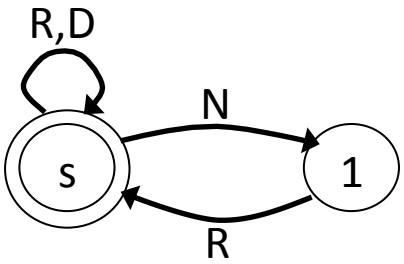
3- Multi-cost regular

- Multi-cost regular :

$$\textsc{Regular}([X_1, \ldots, X_n], A) \wedge (\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R)$$

- Example:
  - Schedule 7 shifts of type: **night (N), day (D), rest (R)**
  - (1) "A **Rest must follow a Night** shift"
  - (2) "**Exactly 3 day shifts** and **1 night shift** must take place in the week"

X_1	X_2	X_3	X_4	X_5	X_6	X_7
D	R	N	R	D	D	R



D R N

$$\mathrm{GCC}([X_1, \ldots, X_7], [3, 0, 1], [3, 7, 1])$$

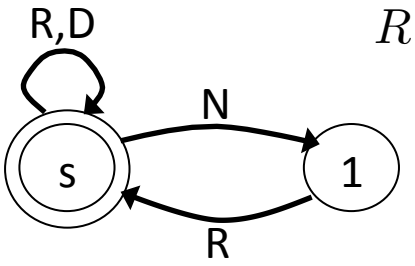
3- Multi-cost regular

- Multi-cost regular :

$$\text{REGULAR}([X_1, \ldots, X_n], A) \wedge \left(\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R\right)$$

- Example:
  - Schedule 7 shifts of type: **night (N), day (D), rest (R)**
  - (1) "A **Rest must follow a Night** shift"
  - (2) "**Exactly 3 day shifts** and **1 night shift** must take place in the week"

X_1	X_2	X_3	X_4	X_5	X_6	X_7
D	R	N	R	D	D	R



$R = 2$

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
c^1_D	1	1	1	1	1	1	1
c^1_N	0	0	0	0	0	0	0
c^1_R	0	0	0	0	0	0	0

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
c^2_D	0	0	0	0	0	0	0
c^2_N	1	1	1	1	1	1	1
c^2_R	0	0	0	0	0	0	0

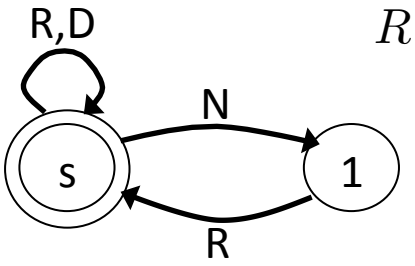
3- Multi-cost regular

- Multi-cost regular :

$$\text{REGULAR}([X_1, \ldots, X_n], A) \wedge \left(\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R\right)$$

- Example:
  - Schedule 7 shifts of type: **night (N), day (D), rest (R)**
  - (1) "A **Rest must follow a Night** shift"
  - (2) "**Exactly 3 day shifts** and **1 night shift** must take place in the week"

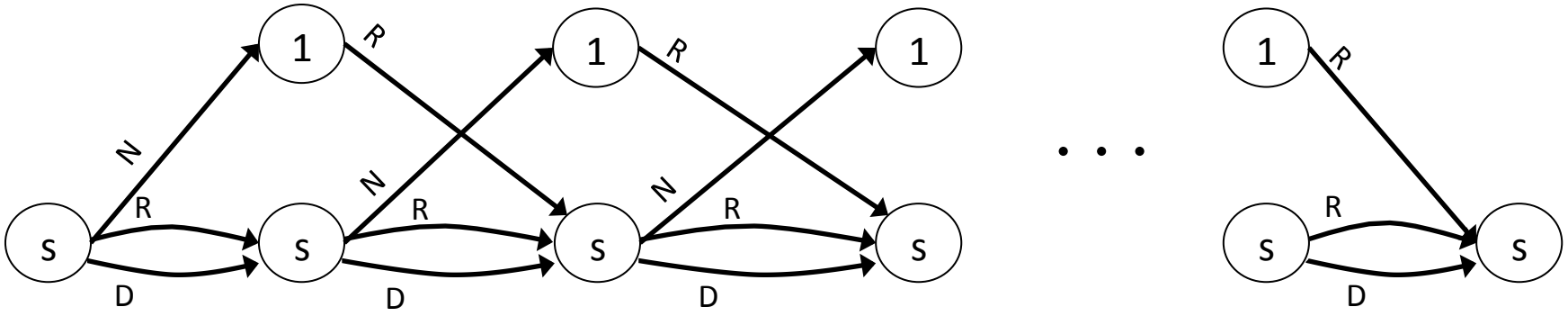
X_1	X_2	X_3	X_4	X_5	X_6	X_7
D	R	N	R	D	D	R



$R = 2$

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
c^1_D	1	1	1	1	1	1	1
c^1_N	0	0	0	0	0	0	0
c^1_R	0	0	0	0	0	0	0

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
c^2_D	0	0	0	0	0	0	0
c^2_N	1	1	1	1	1	1	1
c^2_R	0	0	0	0	0	0	0



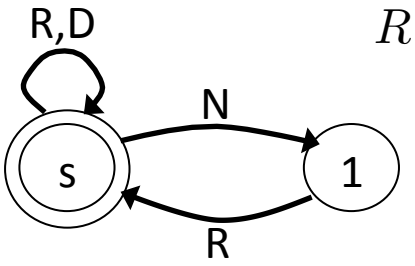
3- Multi-cost regular

- Multi-cost regular :

$$\text{REGULAR}([X_1, \ldots, X_n], A) \wedge \left(\sum_{i=1}^{n} c^r_{iX_i} = Z^r, \forall\, r = 0, \ldots, R\right)$$

- Example:
  - Schedule 7 shifts of type: **night (N), day (D), rest (R)**
  - (1) "A **Rest must follow a Night** shift"
  - (2) "**Exactly 3 day shifts** and **1 night shift** must take place in the week"

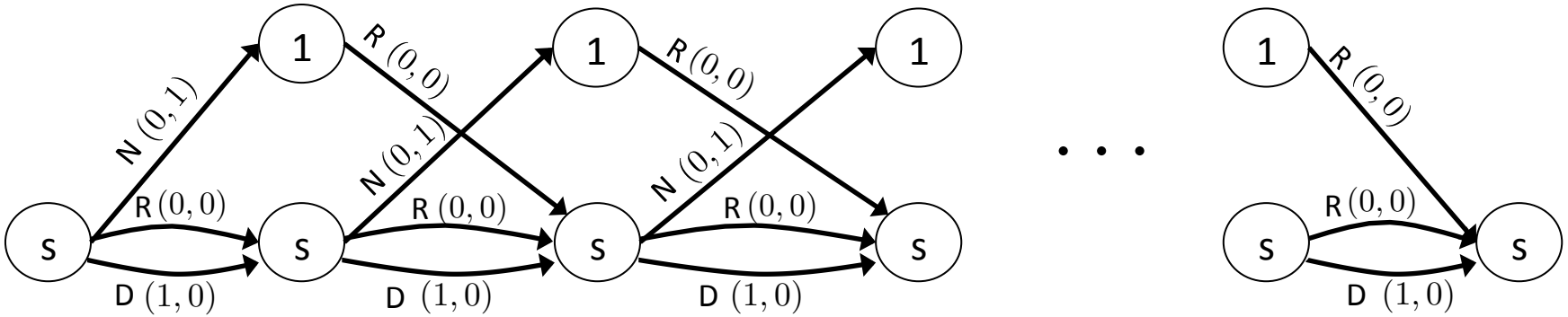
X_1	X_2	X_3	X_4	X_5	X_6	X_7
D	R	N	R	D	D	R



$R = 2$

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
c^1_D	1	1	1	1	1	1	1
c^1_N	0	0	0	0	0	0	0
c^1_R	0	0	0	0	0	0	0

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
c^2_D	0	0	0	0	0	0	0
c^2_N	1	1	1	1	1	1	1
c^2_R	0	0	0	0	0	0	0



3- Weighted-circuit

$$\text{WEIGHTED-CIRCUIT}(X = [X_1, \ldots, X_n], Z)$$

$$X_i = j \text{ means } j \text{ is the successor of } i$$

- Enforce X to be a Hamiltonian tour of weight at most Z

$$\text{CIRCUIT}(X = [X_1, \ldots, X_n]) \wedge \sum_{i=1}^{n} c_{iX_i} \leq Z$$

3- Weighted-circuit

$$\text{WEIGHTED-CIRCUIT}(X = [X_1, \ldots, X_n], Z)$$
$$X_i = j \text{ means } j \text{ is the successor of } i$$

[*Benchimol et al, 2012]: is using a set variable instead to represent the edges in the circuit*

- Enforce X to be a Hamiltonian tour of weight at most Z

$$\text{CIRCUIT}(X = [X_1, \ldots, X_n]) \wedge \sum_{i=1}^{n} c_{iX_i} \leq Z$$

3- Weighted-circuit

$$\text{WEIGHTED-CIRCUIT}(X = [X_1, \ldots, X_n], Z)$$

$$X_i = j \text{ means } j \text{ is the successor of } i$$

- Enforce X to be a Hamiltonian tour of weight at most Z

$$\text{CIRCUIT}(X = [X_1, \ldots, X_n]) \wedge \sum_{i=1}^{n} c_{iX_i} \leq Z$$

- Filtering based on graph structure [Fages et al, 2012] [Caseau et al, 1997]

3- Weighted-circuit

$$\text{WEIGHTED-CIRCUIT}(X = [X_1, \ldots, X_n], Z)$$

$$X_i = j \text{ means } j \text{ is the successor of } i$$

- Enforce X to be a Hamiltonian tour of weight at most Z

$$\text{CIRCUIT}(X = [X_1, \ldots, X_n]) \wedge \sum_{i=1}^{n} c_{iX_i} \leq Z$$

[Fages et al, 2012]

- Filtering based on graph structure [Caseau et al, 1997]
- Filtering based on the Held and Karp 1-Tree relaxation [Benchimol et al, 2012]
  - Relax the **tour** into a **1-tree** (a tree over all nodes except one + 2 edges connected to the ignored node)
  - Lagrangian subproblem based on a minimum spanning tree

3- Weighted-circuit

$$\text{WEIGHTED-CIRCUIT}(X = [X_1, \ldots, X_n], Z)$$

$$X_i = j \text{ means } j \text{ is the successor of } i$$

- Enforce X to be a Hamiltonian tour of weight at most Z

$$\text{CIRCUIT}(X = [X_1, \ldots, X_n]) \wedge \sum_{i=1}^{n} c_{iX_i} \leq Z$$

[Fages et al, 2012]

- Filtering based on graph structure

[Caseau et al, 1997]

- Filtering based on the Held and Karp 1-Tree relaxation

[Benchimol et al, 2012]

  - Relax the **tour** into a **1-tree** (a tree over all nodes except one + 2 edges connected to the ignored node)
  - Lagrangian subproblem based on a minimum spanning tree

- Use "Lagrangian reduced-cost" to identify:
  - Edges that must be in the tour
  - Edges that can not be in a "better" tour

[Benchimol et al, 2012]

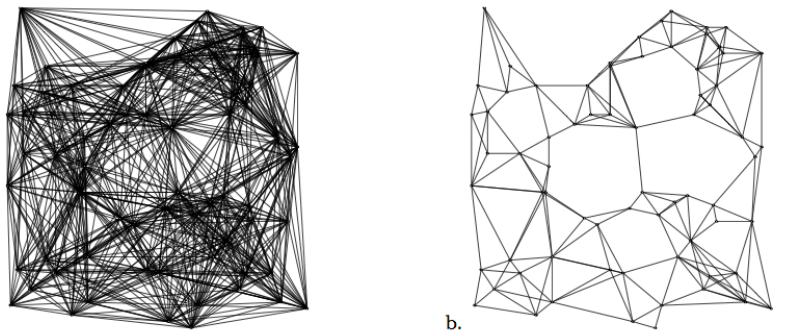


Fig. 3 The filtered graph for **st70** with respect to an upper bound of 700 (a) and 675 (b).

3- Weighted-circuit

$$\text{WEIGHTED-CIRCUIT}(X = [X_1, \ldots, X_n], Z)$$

$$X_i = j \text{ means } j \text{ is the successor of } i$$

- Enforce X to be a Hamiltonian tour of weight at most Z

$$\text{CIRCUIT}(X = [X_1, \ldots, X_n]) \wedge \sum_{i=1}^{n} c_{iX_i} \leq Z$$

[Fages et al, 2012]

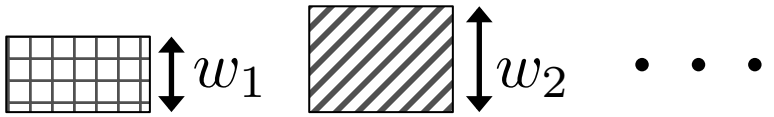
- Filtering based on graph structure [Caseau et al, 1997]
- Filtering based on the Held and Karp 1-Tree relaxation [Benchimol et al, 2012]
  - Relax the **tour** into a **1-tree** (a tree over all nodes except one + 2 edges connected to the ignored node)
  - Lagrangian subproblem based on a minimum spanning tree

- Strong filtering based on dynamic programming when the number of visited nodes is small (around 15-20 : very common in wide a range of applications) [Cambazard et al, 2012]

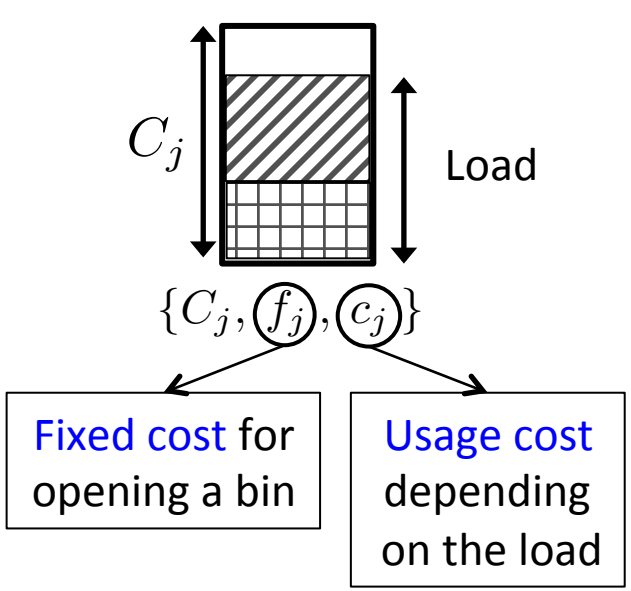
3- Bin Packing with Usage Costs

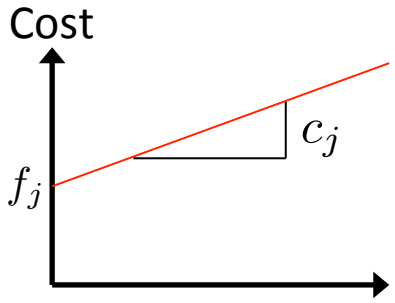
$\textsc{BinPackingUsageCost}([X_1, \ldots, X_n], [L_1, \ldots, L_m], [Y_1, \ldots, Y_m], T, B, S)$

- A set of items $\mathrm{S} = \{w_1, \ldots, w_n\}$



- A set of bins $B = \{\{C_1, f_1, c_1\}, \ldots, \{C_m, f_m, c_m\}\}$





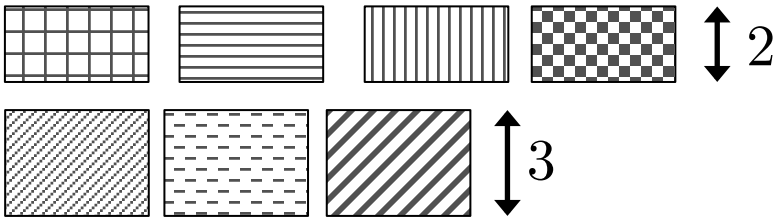
$$cost_j = f_j + Load_j c_j$$

- Minimize $\sum_{j=1|Load_j>0}^{m} cost_j$

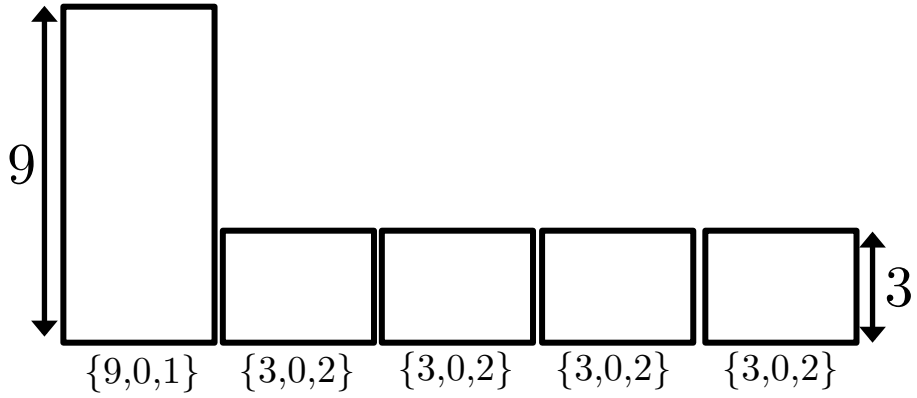
3- Bin Packing with Usage Costs

$\textsc{BinPackingUsageCost}([X_1, \ldots, X_n], [L_1, \ldots, L_m], [Y_1, \ldots, Y_m], T, B, S)$

- A set of items $\mathrm{S} = \{w_1, \ldots, w_n\}$
- A set of bins $B = \{\{C_1, f_1, c_1\}, \ldots, \{C_m, f_m, c_m\}\}$
- Minimize the sum of the costs of the used bins



Items

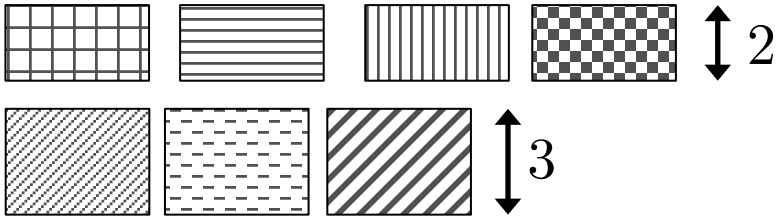


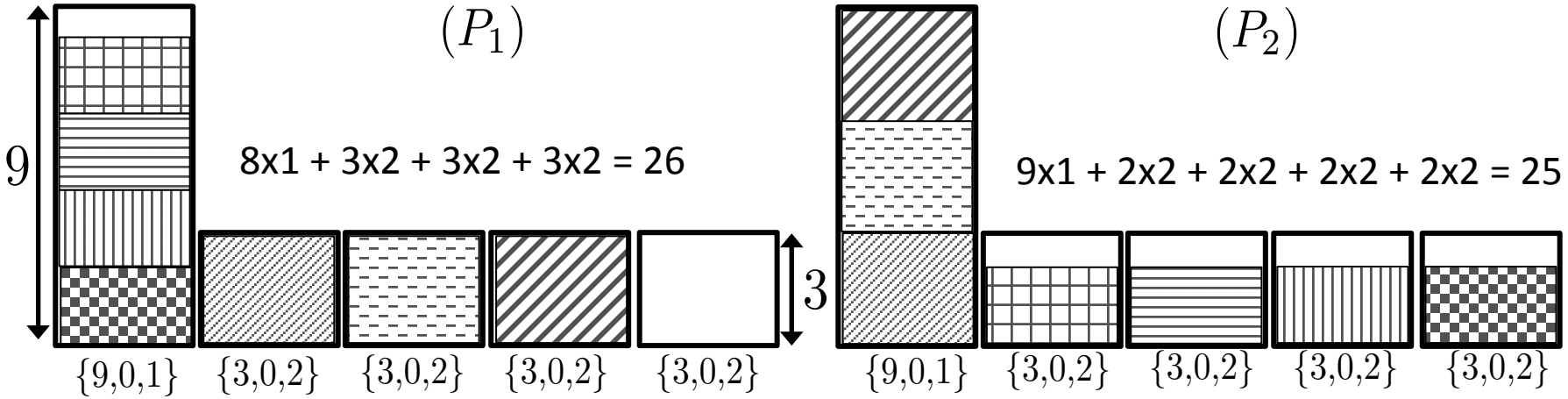
Bins

3- Bin Packing with Usage Costs

$\textsc{BinPackingUsageCost}([X_1, \ldots, X_n], [L_1, \ldots, L_m], [Y_1, \ldots, Y_m], T, B, S)$

- A set of items $\mathrm{S} = \{w_1, \ldots, w_n\}$
- A set of bins $B = \{\{C_1, f_1, c_1\}, \ldots, \{C_m, f_m, c_m\}\}$
- Minimize the sum of the costs of the used bins





3- Bin Packing with Usage Costs

$\textsc{BinPackingUsageCost}([X_1, \ldots, X_n], [L_1, \ldots, L_m], [Y_1, \ldots, Y_m], T, B, S)$

- LP relaxation easy to characterize and fast cost filtering can be done [Cambazard et al, 2013]

- Stronger filtering can be achieved using Lagrangian relaxation
  - Relax the constraint enforcing an item to occur in exactly one bin.
  - Lagrangian sub-problem is a knapsack and dynamic Programming provides the reduced costs.

Overview of Lagrangian based filtering for NP-Hard global constraints

Constraint	Lagrangian Subproblem	Examples of applications	References
Multi-cost-regular	Shortest/Longest Path	Personnel Scheduling	[Menana et al, 2009]
Weighted-circuit	1-Tree (Spanning Tree)	Traveling Salesman Problem Traveling Purchaser Problem Traveling Tournament	[Caseau et al, 1997] [Benoist et al, 2001] [Benchimol et al, 2012] [Fages et al, 2012] [Cambazard et al, 2012]
Weighted - atMostNValue	Sorting	Traveling Purchaser Problem Warehouse location	[Cambazard et al, 2012]
atMostNValue	Inspection	…	
Bin-Packing with usage costs	Knapsack	Energy optimization in data-centers	
Shortest Path in DAG with resource constraints	Shortest Path	Multileaf collimator sequencing	[Sellmann, 2005] [Cambazard et al, 2010]

Other applications:

- Golomb rulers [Van Hoove, 2013],
- Automated Recording Problem [Sellmann, 2003]
- Capacitated Network Design [Sellmann, 2002]

Plan

1. Context and motivation
   - Illustrative application: the Traveling Purchaser Problem
   - *Optimization* versus *Satisfaction*
   - *Combinatorial* versus *polyhedral* methods

2. Propagation based on Lagrangian Relaxation
   - Lagrangian duality
   - Filtering using Lagrangian reduced costs
   - Let's try on the *Nvalue* global constraint

3. Overview of some NP-Hard Constraints with costs
   - *Multi-cost regular, Weighted-circuit, Weighted-Nvalue, Bin-packing with usage costs*

4. Examples of applications

Back to the Traveling Purchaser Problem

Problem structures

$Nvisit \in \{1, \ldots, B\}$: Number of visited markets
$TotalCost = TravelingCost + ShoppingCost$

Relaxation	Nature of the problem	Value of the parameter	How to solve / propagate it ?	Key propagation
Feasibility	Hitting Set AtMostNValue	$\overline{Nvisit}$ (cardinality)	[Bessière et al, 2006]	$\underline{Nvisit}$
Feasibility + Shopping cost	p-median Weighted-NValue	$p = \overline{Nvisit}$	Lagrangian relaxation	$\frac{\underline{ShoppingCost}}{\underline{Nvisit}}$
Traveling Cost	k-TSP Close to Weighted-Circuit	$k = \underline{Nvisit}$	Dynamic Programming ? Lagrangian relaxation	$\frac{\underline{TravelingCost}}{\overline{Nvisit}}$

CP Model for the TPP

$Nvisit \in \{1, \ldots, B\}$: Number of visited markets
$y_i \in \{0, 1\}$: do we visit market **i** ?
$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought
$Cs_k \geq 0$: the price paid for item **k**

$$Minimize\ TravelingCost + ShoppingCost$$

$$Cs_k = \text{ELEMENT}([b_{k1}, \ldots, b_{ki}, \ldots, b_{km}], s_k)$$

$$\exists i \mid s_k = i \Leftrightarrow y_i = 1$$ (channeling s_k and y_i)

$$\text{NVALUE}([s_1, \ldots, s_m], Nvisit)$$

$$\text{TSP}([y_1, \ldots, y_n], Nvisit, TravelingCost, \ldots)$$

$$\text{Weighted-NValue}([s_1, \ldots, s_m], Nvisit, ShoppingCost, \ldots)$$

CP Model for the TPP

$Nvisit \in \{1, \ldots, B\}$: Number of visited markets
$y_i \in \{0, 1\}$: do we visit market **i** ?
$s_k \in \{i | v_i \in M_k\}$: the market where item **k** is bought
$Cs_k \geq 0$: the price paid for item **k**

$$Minimize\ TravelingCost + ShoppingCost$$

$$Cs_k = \text{ELEMENT}([b_{k1}, \ldots, b_{ki}, \ldots, b_{km}], s_k)$$

$$\exists i \mid s_k = i \Leftrightarrow y_i = 1$$ (channeling s_k and y_i)

$$\text{NVALUE}([s_1, \ldots, s_m], Nvisit)$$

$$\text{TSP}([y_1, \ldots, y_n], Nvisit, TravelingCost, \ldots)$$ → Close to WEIGHTED-CIRCUIT

$$\text{WEIGHTED-NVALUE}([s_1, \ldots, s_m], Nvisit, ShoppingCost, \ldots)$$

Overview of results on TPP

- Benchmark (Laporte class3):
  - 100 instances: up to 250 markets and 200 items
  - 11 open instances

- Very efficient when the optimal solution contains few markets
- Very complementary to [Laporte and al]

n	m	Nvisit	Obj BCP	Time BCP	Obj CP	Time CP
250	50	5	3161	17399 s	3161	**0.6 s**
250	150	18	2121	> 18000 s	1531	**417 s**
150	200	25	2594	**1317 s**	2594	5677 s
200	100	28	3161	**8599 s**	3178	> 7200 s

- CP only fails to prove optimality on 10 instances
- Closes 8 instances out of the 11 open instances (improves 10 best known solutions)

Conclusion

- A CP model reveals combinatorial structures of the problem
- Some applications require strong reasoning involving costs (and key NP-Hard sub-problems).
- Lagrangian relaxation (LR) can provide a suitable filtering mechanism without the need of an LP solver:
  - LR can be faster than LP to compute the bound
  - LR can provide more filtering
  - Drawbacks of LR:
    - It needs parameters (when using a sub-gradient algorithm)
    - It can experience issues for converging

References

- Y. Caseau, F. Laburthe*, Solving Small TSPs with Constraints*, ICLP 1997
- P. Refalo, *Linear formulation of Constraint Programming models and Hybrid Solvers* CP 2000
- T. Benoist, F. Laburthe, B. Rottembourg *Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems,* CP-AI-OR 2001
- F. Focacci, A. Lodi, M. Milano, *Embedding relaxations in global constraints for solving TSP and TSPTW*, Ann. Math. Artif. Intell., 2002
- M. Sellmann, T. Fahle: *Constraint Programming Based Lagrangian Relaxation for the Automatic Recording Problem*. Annals OR 118(1-4): 17-33, 2003
- M. Sellmann: *Theoretical Foundations of CP-Based Lagrangian Relaxation*. CP 2004: 634-647
- T. Gellermann, M. Sellmann, R. Wright: *Shorter Path Constraints for the Resource Constrained Shortest Path Problem*. CPAIOR 2005: 201-216
- J. Menana, S. Demassey: *Sequencing and Counting with the multicost-regular Constraint*. CPAIOR 2009: 178-192
- Hadrien Cambazard, Eoin O'Mahony, Barry O'Sullivan: *Hybrid Methods for the Multileaf Collimator Sequencing Problem*. CPAIOR 2010: 56-70
- P. Benchimol, W.J. van Hoeve, J.C. Régin, L.M. Rousseau, M. Rueher, Improved filtering for weighted circuit constraints. Constraints 17(3): 205-233, 2012
- J.G. Fages, X. Lorca*: Improving the Asymmetric TSP by Considering Graph Structure*. CoRR abs/1206.3437, 2012
- H. Cambazard, B. Penz: A Constraint Programming Approach for the Traveling Purchaser Problem. CP 2012: 735-749
- H. Cambazard, E. O'Mahony, B. O'Sullivan: *A shortest path-based approach to the multileaf collimator sequencing problem*. Discrete Applied Mathematics 160(1-2): 81-99 (2012)
- M. R. Slusky, W.J. van Hoeve*: A Lagrangian Relaxation for Golomb Rulers*. CPAIOR 2013: 251-267