香港中文大學
The Chinese University of Hong Kong

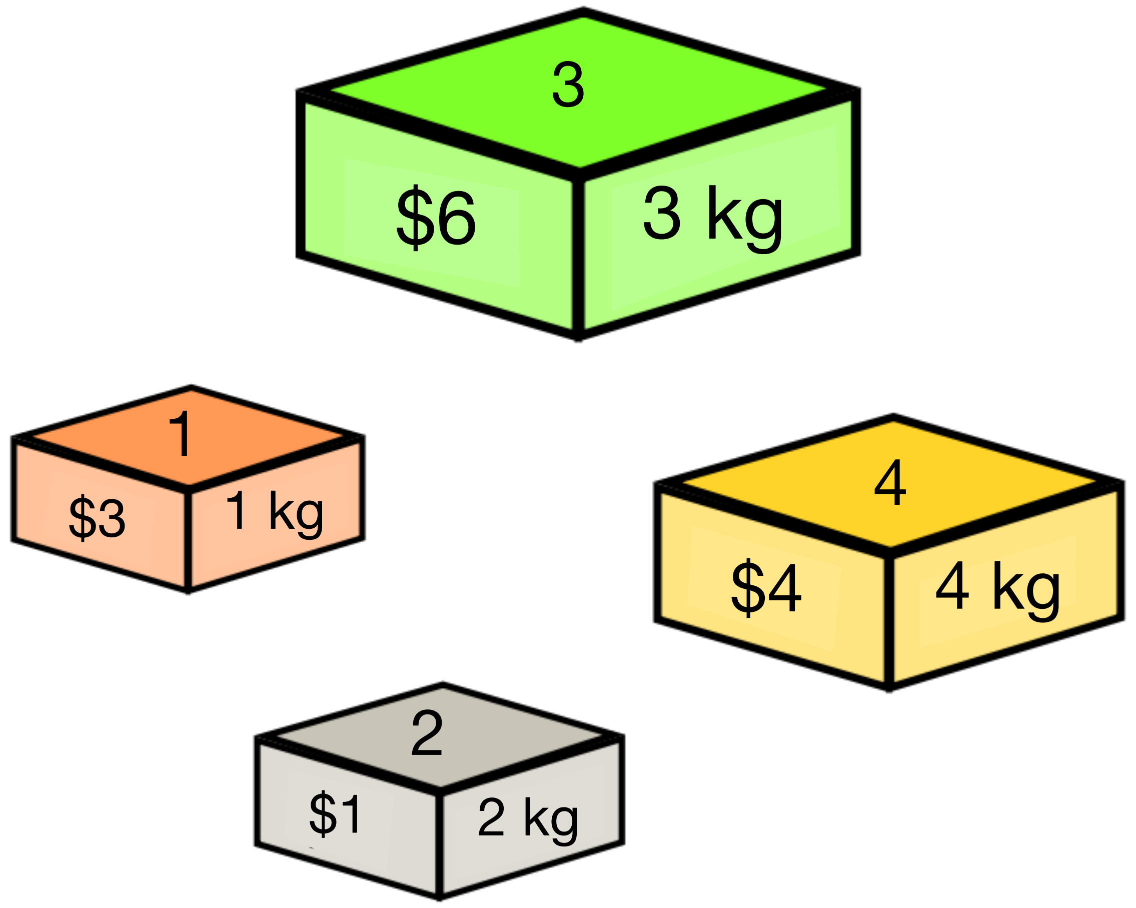
Automatic Generation of Dominance Breaking Nogoods for Constraint Optimisation

Jimmy H.M. Lee and Allen Z. Zhong
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Constraint Optimisation Problems (COPs)

- Example: 0-1 Knapsack
  - Variables: x_1, x_2, x_3, x_4
  - Domains: $x_i \in \{0,1\}$ for $i = 1,\ldots,4$
  - Constraint: $x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$
  - Objective: maximize $3x_1 + x_2 + 6x_3 + 4x_4$

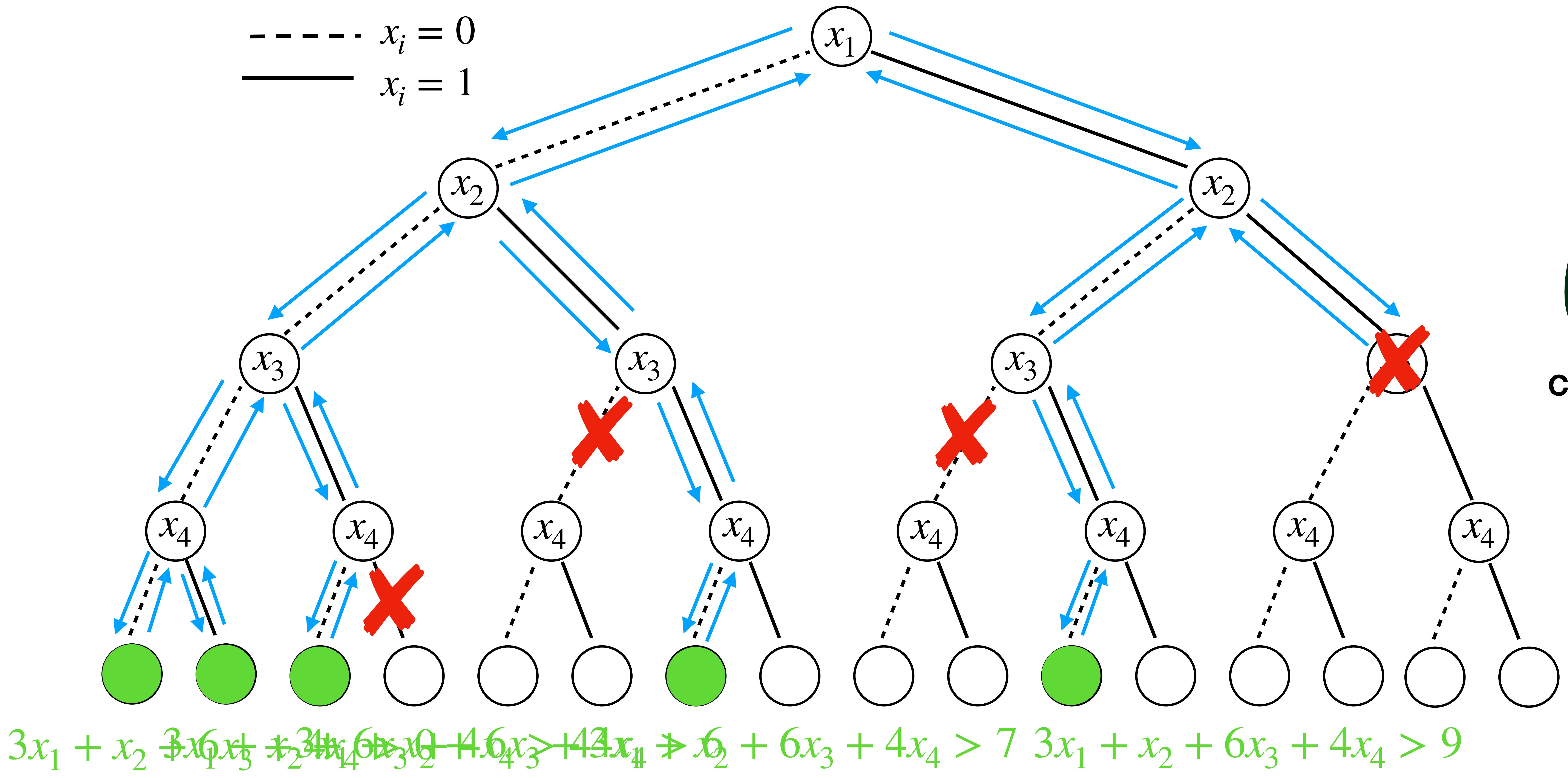
Capacity: 5 kg



- Goal: find an assignment of values to variables such that
  - all constraints are satisfied, and
  - the objective is optimized

Dominance Breaking

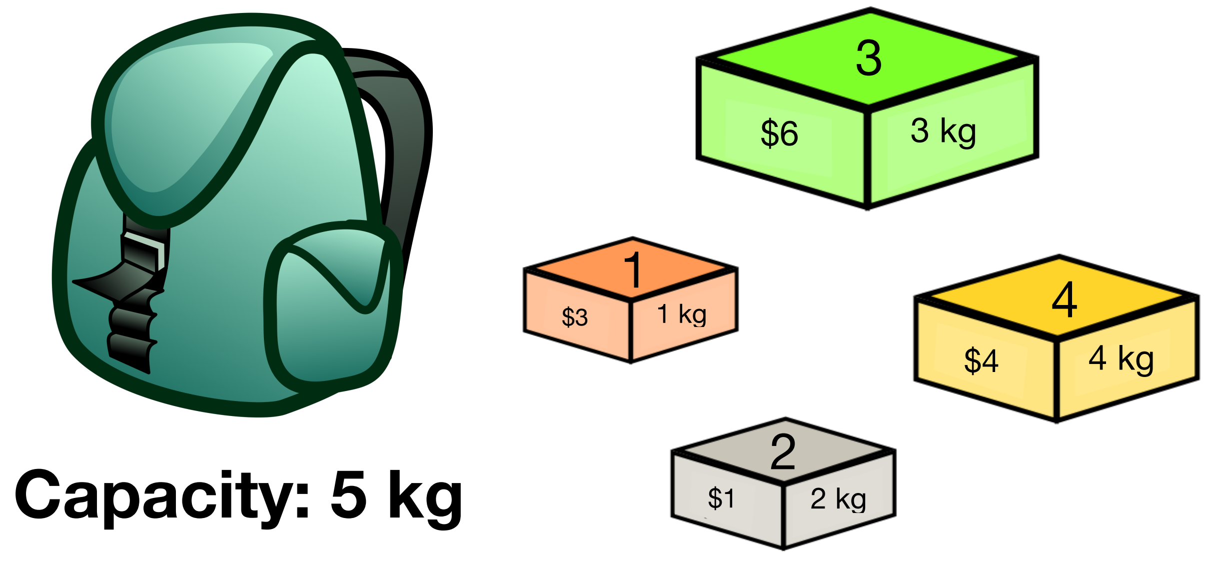
- Branch and bound:



maximize $3x_1 + x_2 + 6x_3 + 4x_4$

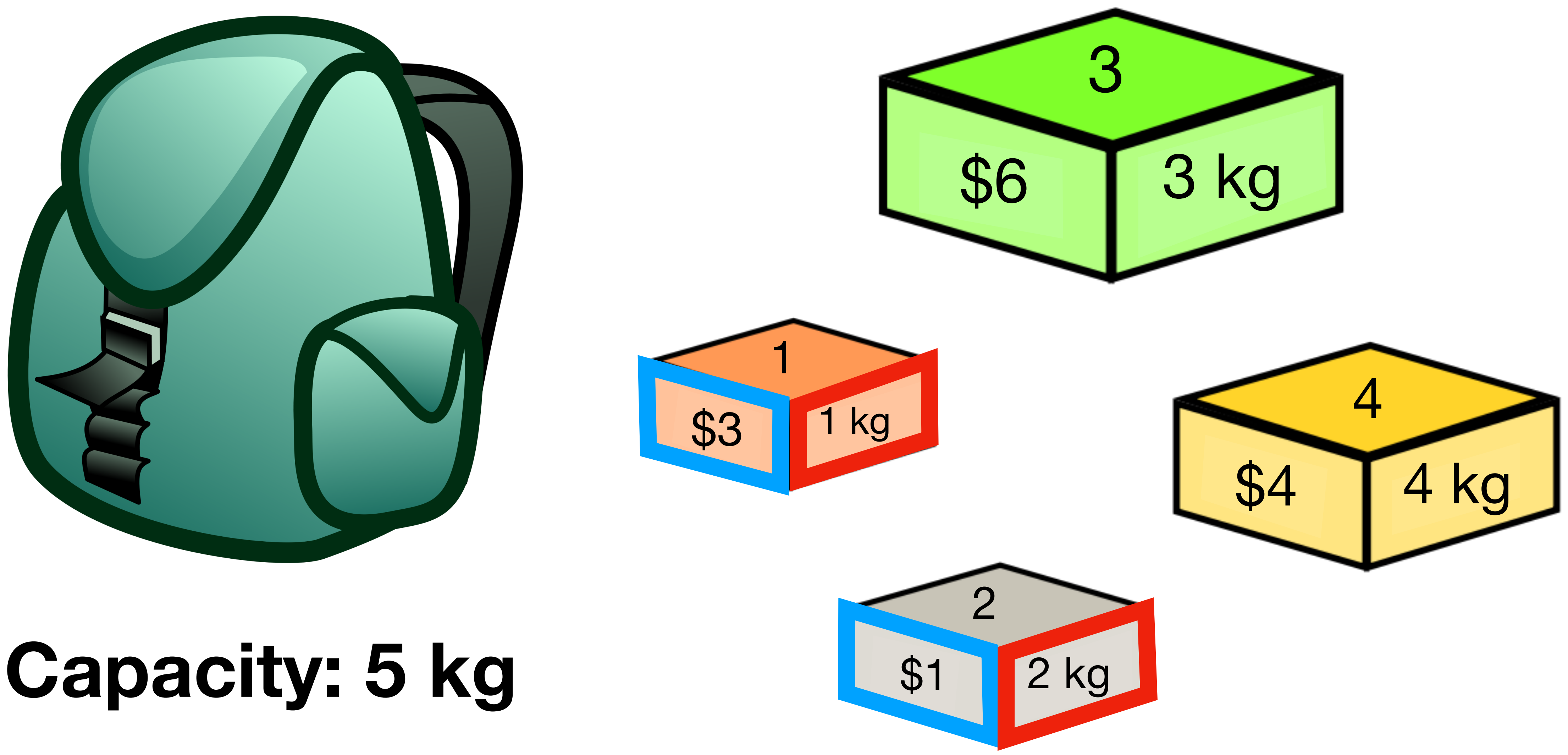
s.t. $x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$

$x_i \in \{0,1\}$ for $i = 1,\ldots,4$



Dominance Breaking

- Dominance breaking is a technique to prune suboptimal assignments.



Capacity: 5 kg

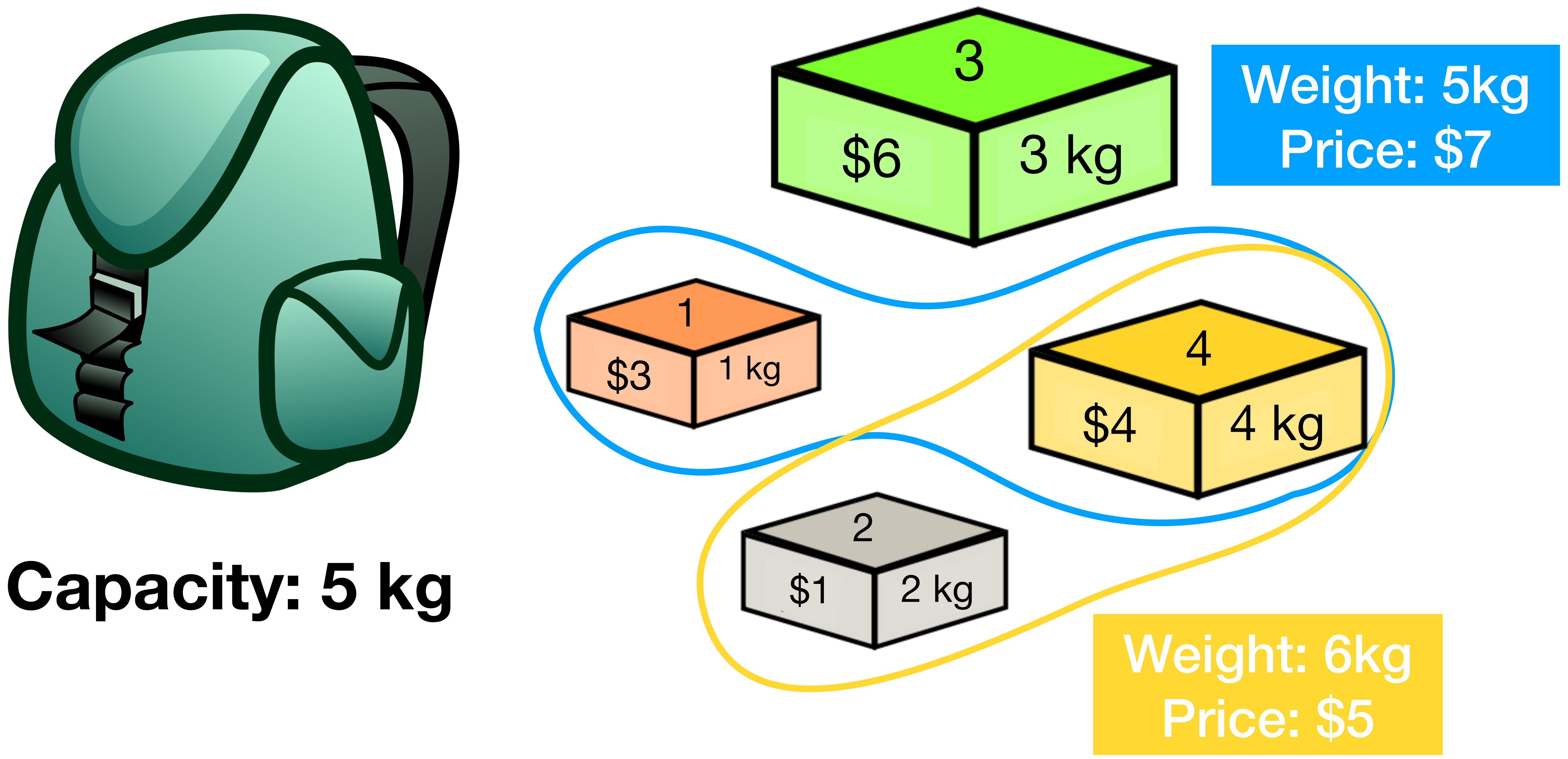
$$\text{maximize } 3x_1 + x_2 + 6x_3 + 4x_4$$

$$\text{s.t. } x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$$

$$x_i \in \{0,1\} \text{ for } i = 1,\ldots,4$$

Dominance Breaking

- Dominance breaking is a technique to prune suboptimal assignments.



maximize $3x_1 + x_2 + 6x_3 + 4x_4$

s.t. $x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$

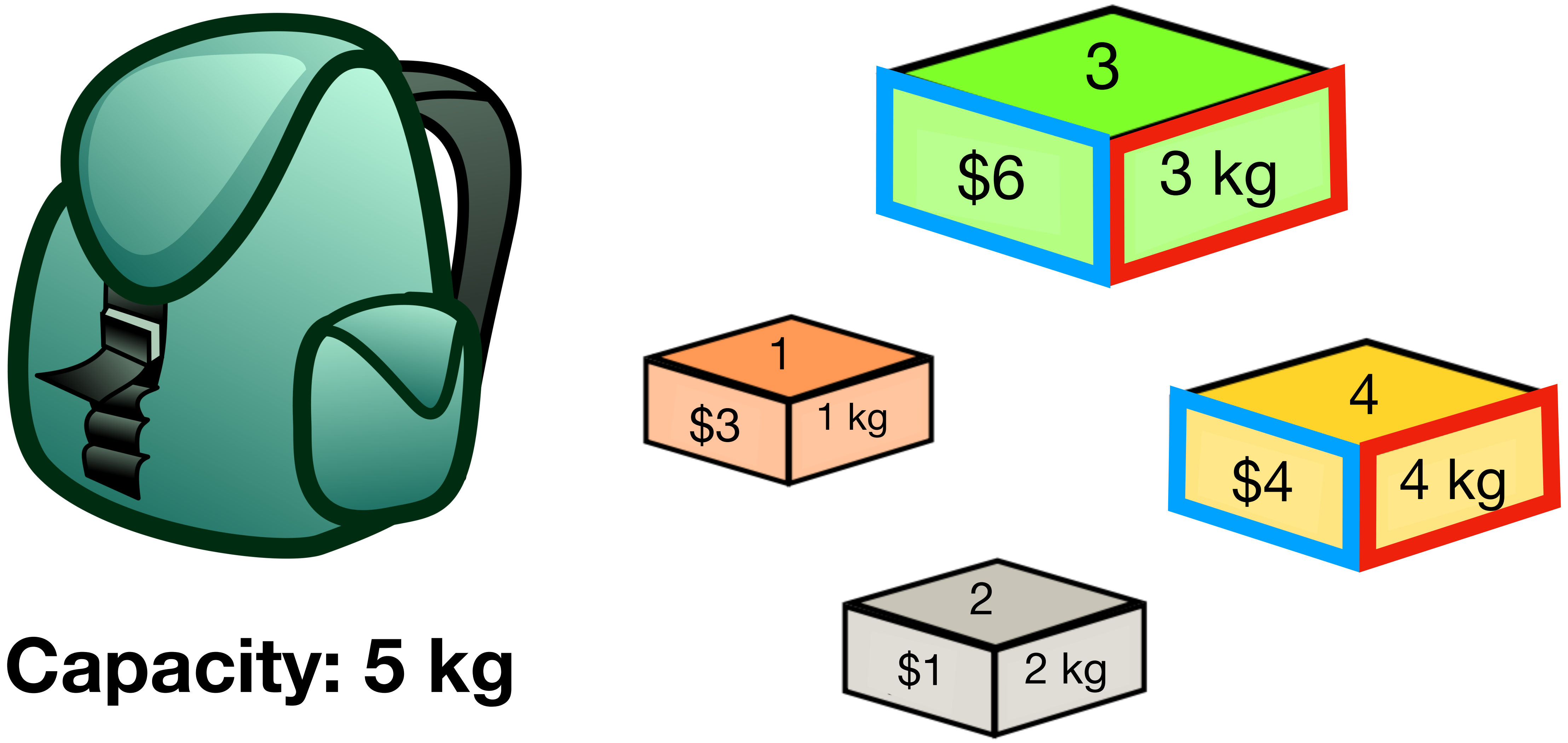
$x_i \in \{0,1\}$ for $i = 1,\ldots,4$

Observation:

Any assignment selecting item 2 but not item 1 must be suboptimal.

Dominance Breaking

- Dominance breaking is a technique to prune suboptimal assignments.



Capacity: 5 kg

maximize $3x_1 + x_2 + 6x_3 + 4x_4$

s.t. $x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$

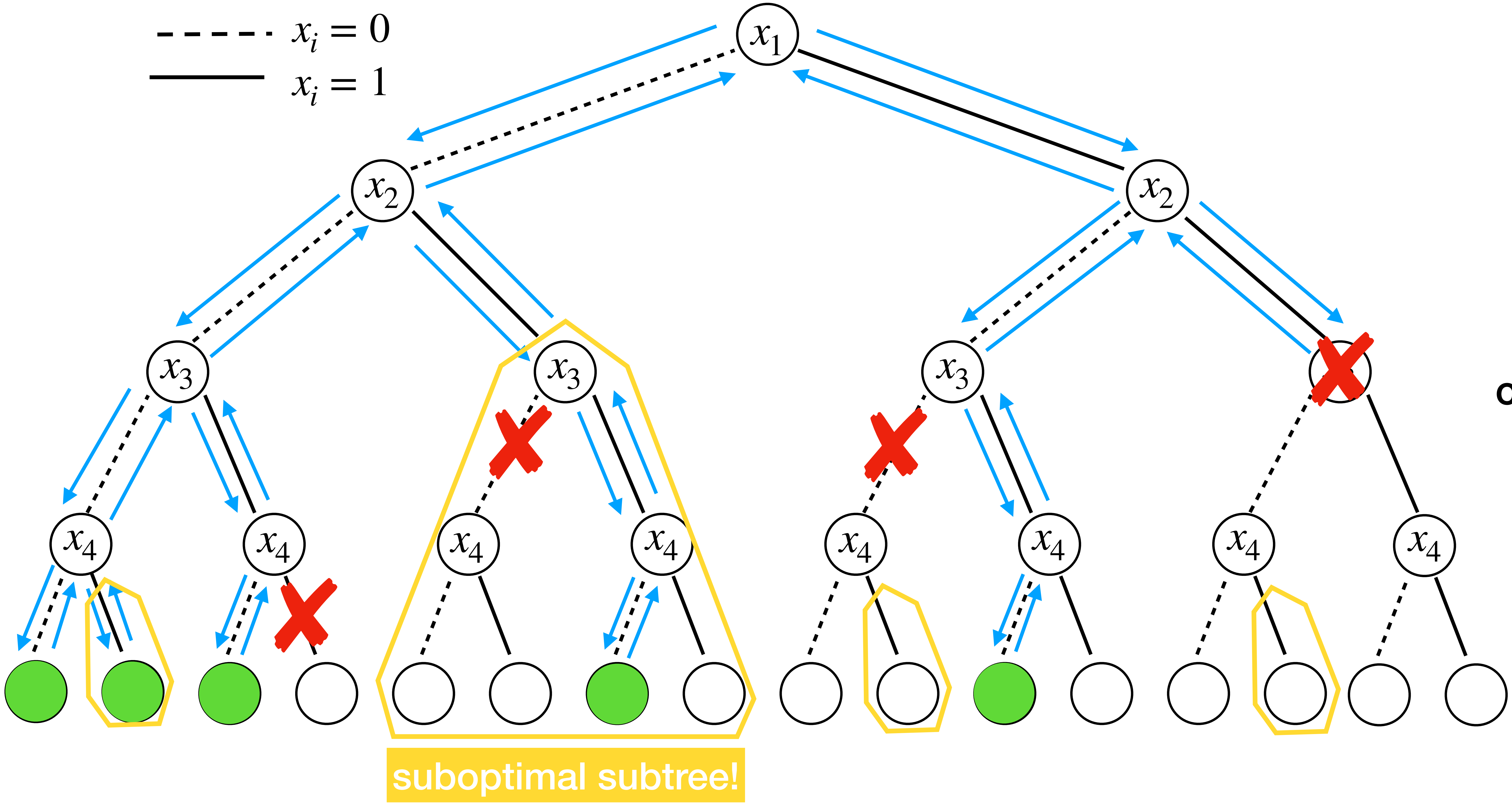
$x_i \in \{0,1\}$ for $i = 1,\ldots,4$

$\boxed{x_2 \leq x_1,\ x_4 \leq x_3}$

Dominance breaking constraints

Dominance Breaking

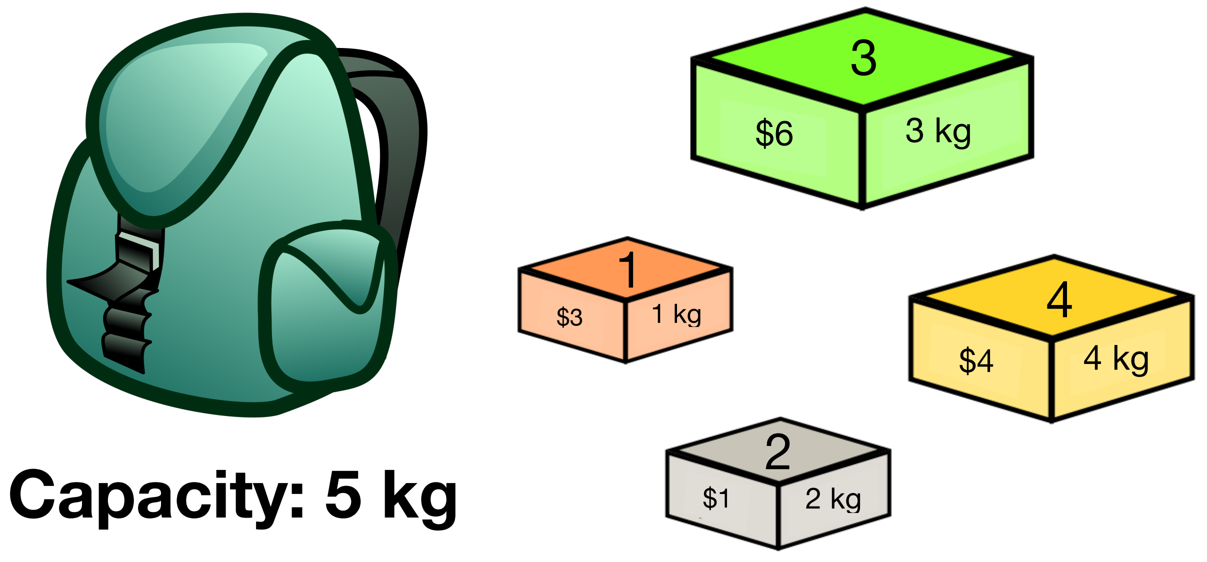
- Branch and bound:



maximize $3x_1 + x_2 + 6x_3 + 4x_4$

s.t. $x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$

$x_i \in \{0,1\}$ for $i = 1,\ldots,4$



$$\boxed{x_2 \leq x_1,\ x_4 \leq x_3}$$

Dominance breaking constraints

Dominance Breaking

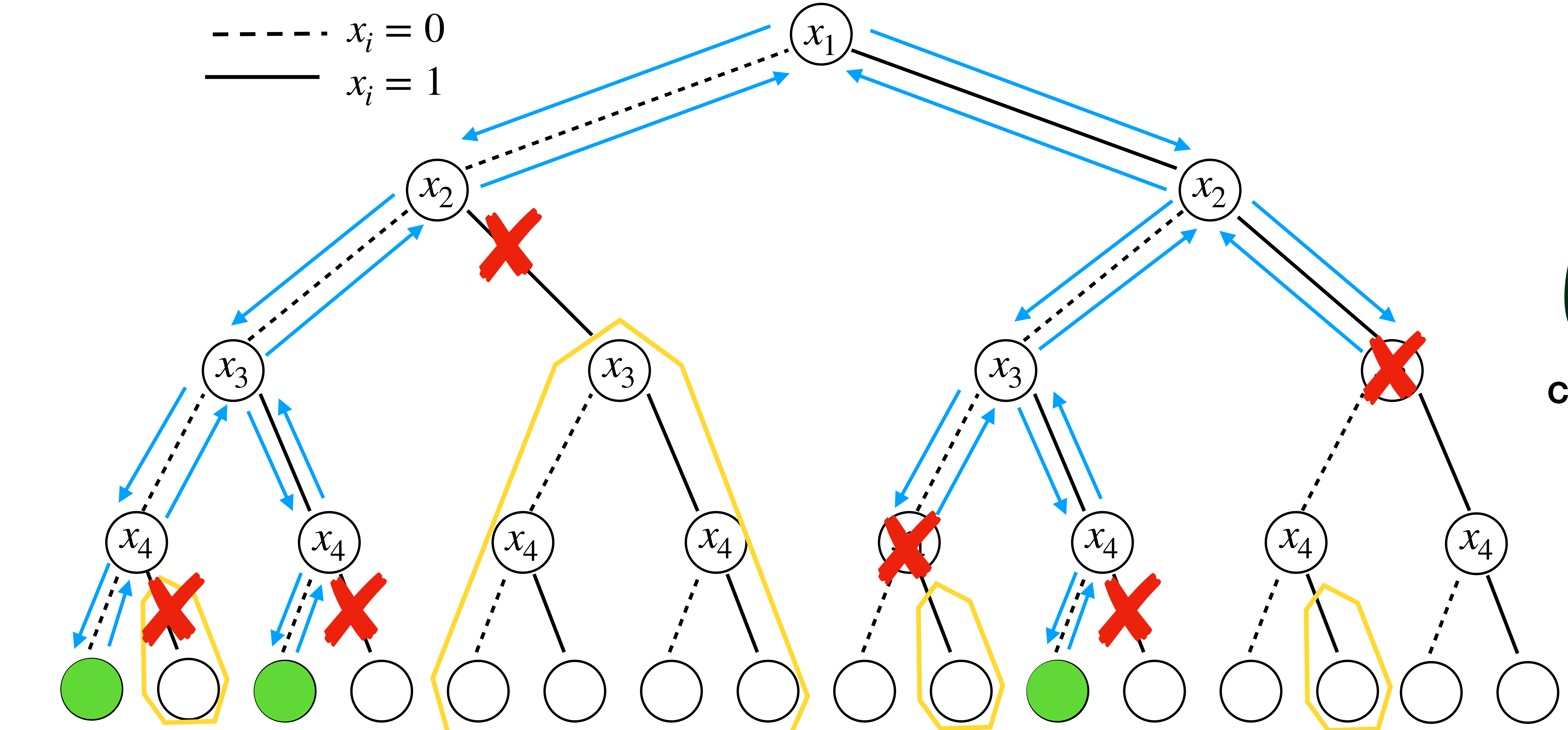
- Branch and bound:

$$\text{maximize } 3x_1 + x_2 + 6x_3 + 4x_4$$

$$\text{s.t. } x_1 + 2x_2 + 3x_3 + 4x_4 \le 5$$

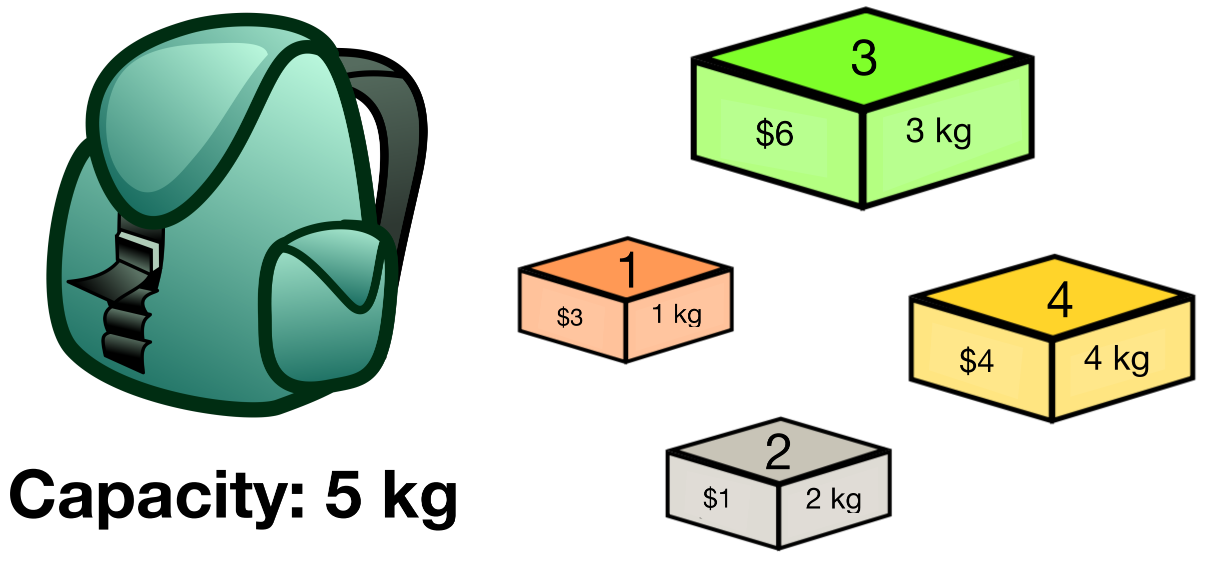
$$x_2 \le x_1,\ x_4 \le x_3$$

$$x_i \in \{0,1\} \text{ for } i = 1,\ldots,4$$



$3x_1 + x_2 + 6x_3 + 4x_4 > 0$ $3x_1 + x_2 + 6x_3 + 4x_4 > 6$

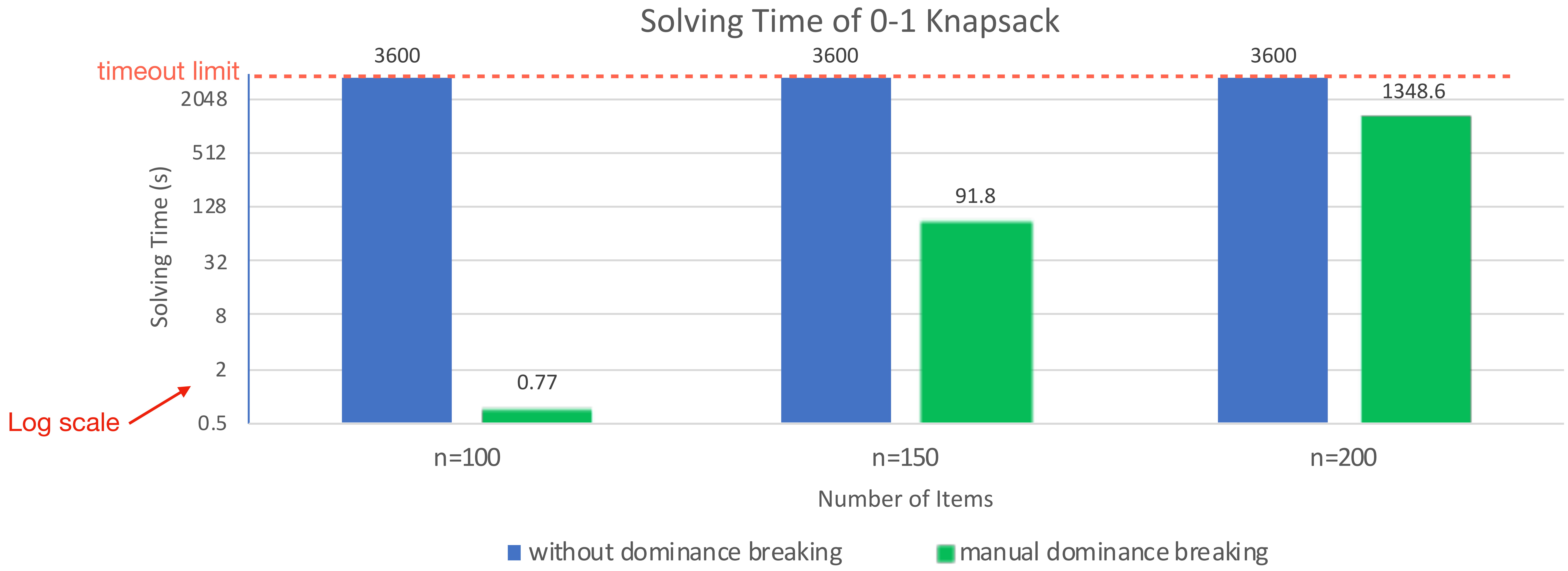
$3x_1 + x_2 + 6x_3 + 4x_4 > 9$



Capacity: 5 kg

Dominance Breaking

- Preliminarily results using the Chuffed solver

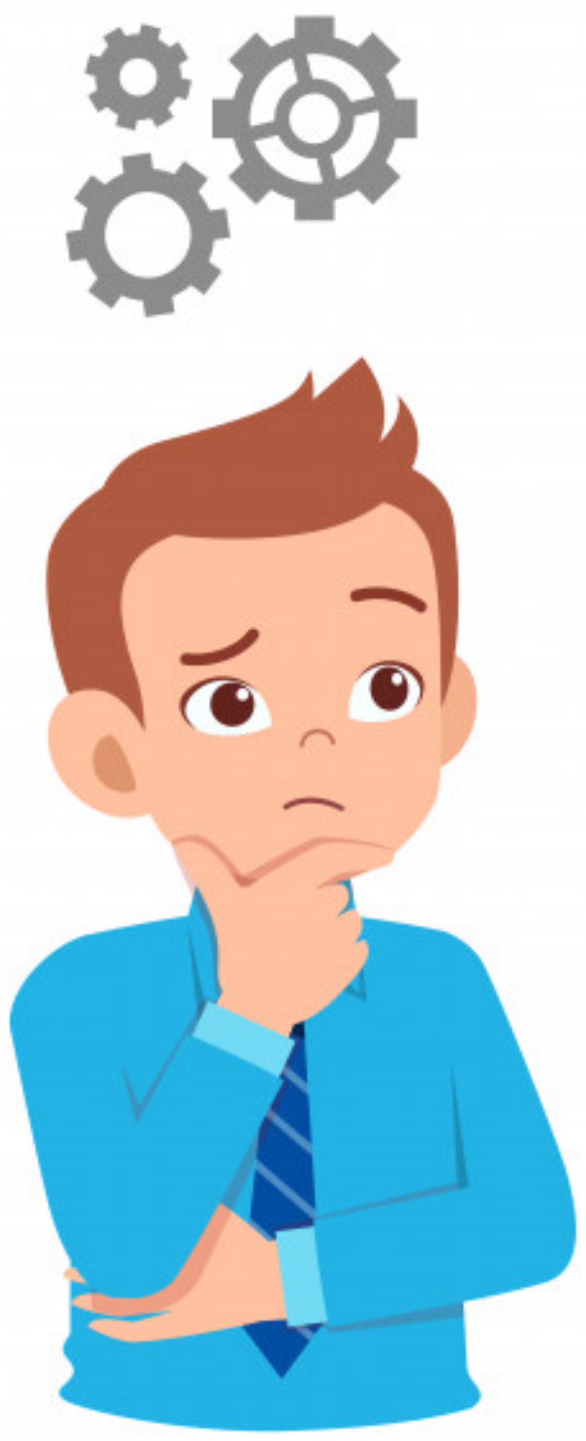


Data source: https://people.eng.unimelb.edu.au/pstuckey/dominance/

Dominance Breaking

- Dominance breaking has been applied successfully in solving many COPs
  - Knapsack problem (Poirriez et al. 2009)
  - Packing problems: rectangle packing (Korf 2004), multicontainer packing (Fukunaga and Korf 2007)
  - Sequencing problems: talent scheduling (Qin et al. 2016, Garcia de la Banda et al. 2011), travelling salesman with time window (Baldacci et al. 2012), minimisation of open stack (Chu et al. 2009)
  - Scheduling problems: balanced academic curriculum problem (Monette et al. 2007), engineer service delivery (Ilankaikone et al. 2021)

Motivation



Problem Model → Model with dominance breaking

Different problems,
Different dominance breaking constraints

Our Approach

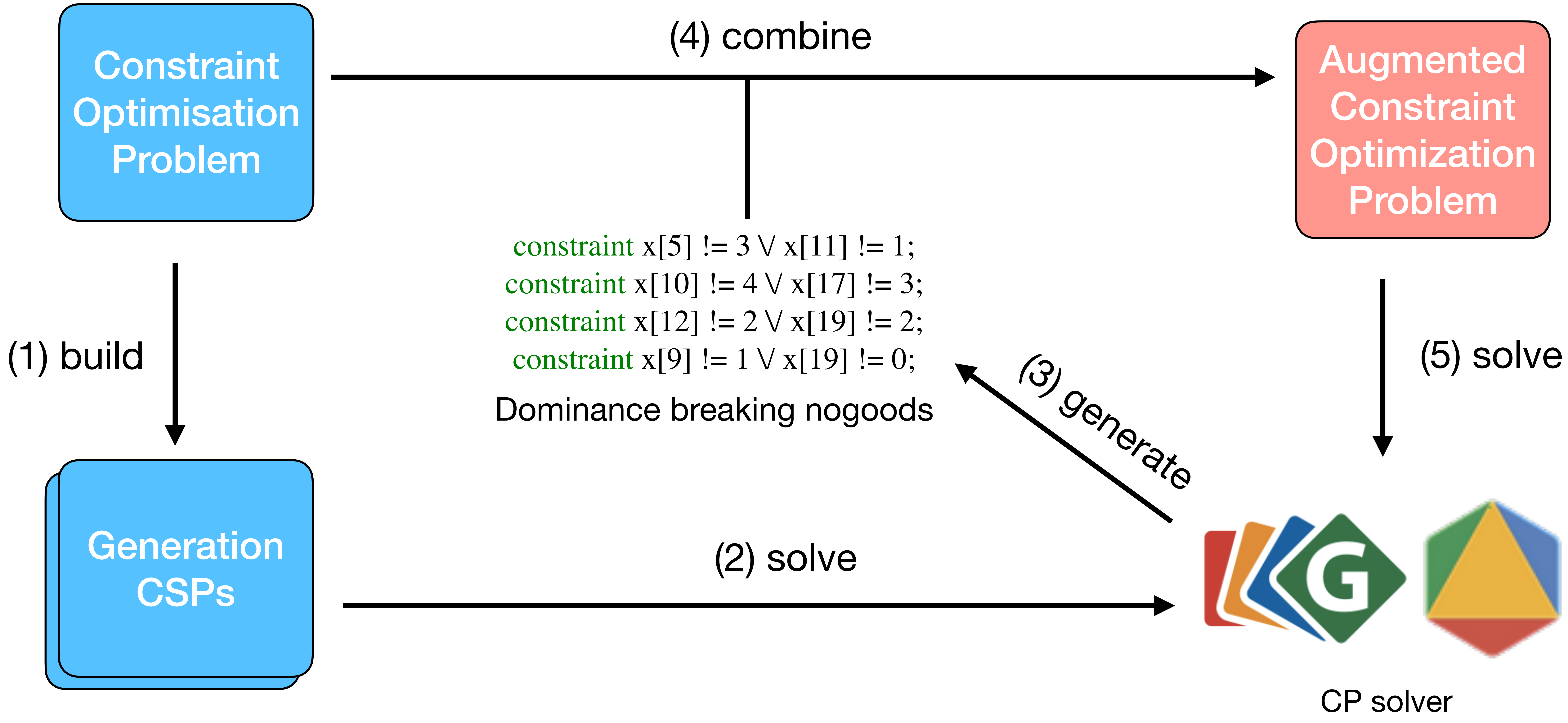
- Focus on nogood constraints
- Full automation
- Solver independence
- More dominance breaking constraints than human
- More efficient than manual methods

Automatic Dominance Breaking

- A formal framework for automatic dominance breaking for a class of constraint optimisation problems
    - Automatic Dominance Breaking for a Class of Constraint Optimization Problems. Jimmy H.M. Lee and **Allen Z. Zhong**, IJCAI-PRICAI 2020
- More efficient generation of dominance breaking nogoods
    - Towards More Practical and Efficient Automatic Dominance Breaking. Jimmy H.M. Lee and **Allen Z. Zhong**, AAAI 2021
- Handling more complex and flexible problems
    - Exploiting Functional Constraints in Generating Dominance Breaking Nogoods for Constraint Optimization. Jimmy H.M. Lee and **Allen Z. Zhong**, CP 2022

Automation Pipeline

(Lee and Zhong 2020)

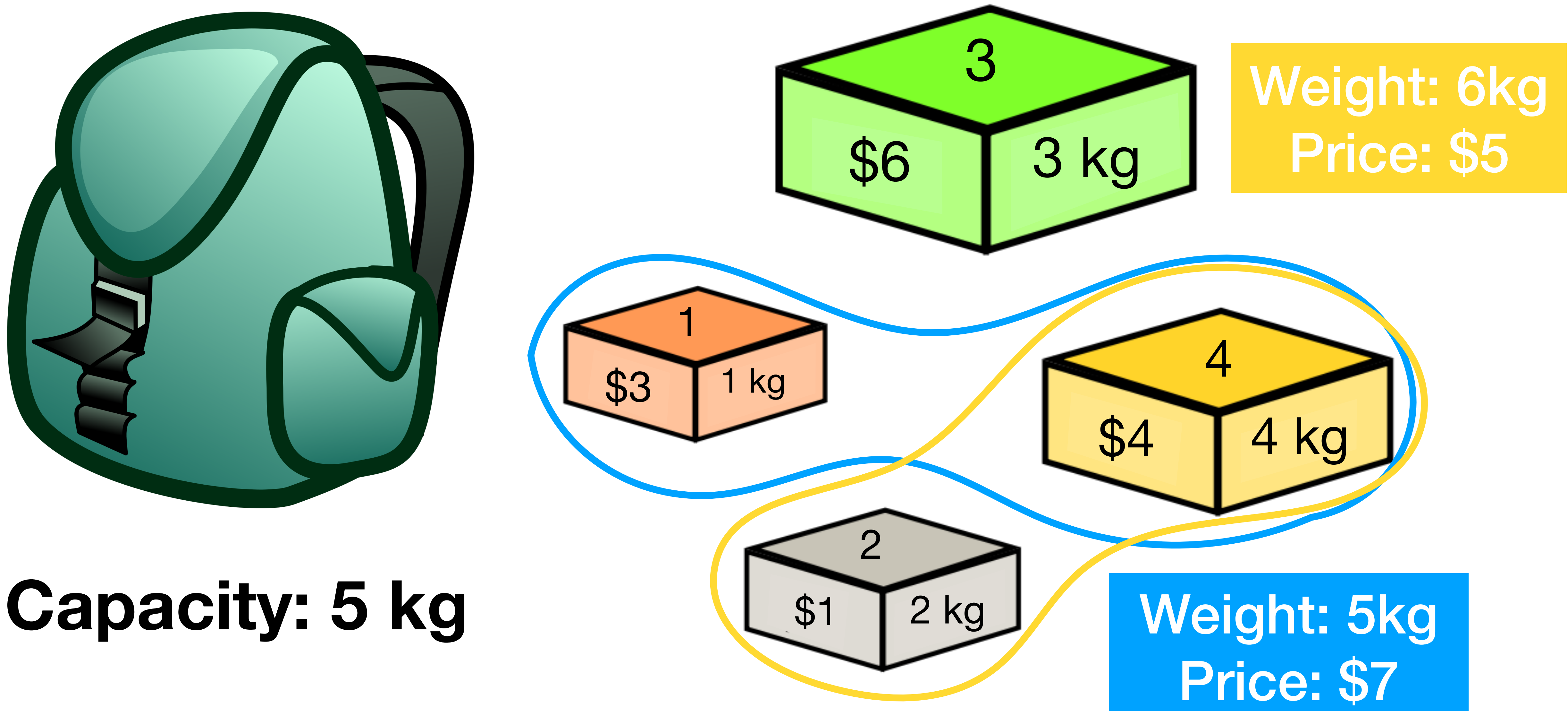


Dominance Relations Over Full Assignments

(Chu and Stuckey 2012)

$\bar{\theta}$ dominates $\bar{\theta}'$ ($\bar{\theta} \prec \bar{\theta}'$):

- $\bar{\theta}$ solution, $\bar{\theta}'$ non-solution



$$\boxed{\bar{\theta} = \{x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 1\}} \prec \boxed{\bar{\theta}' = \{x_1 = 0,\ x_2 = 0,\ x_3 = 1,\ x_4 = 1\}}$$

solution

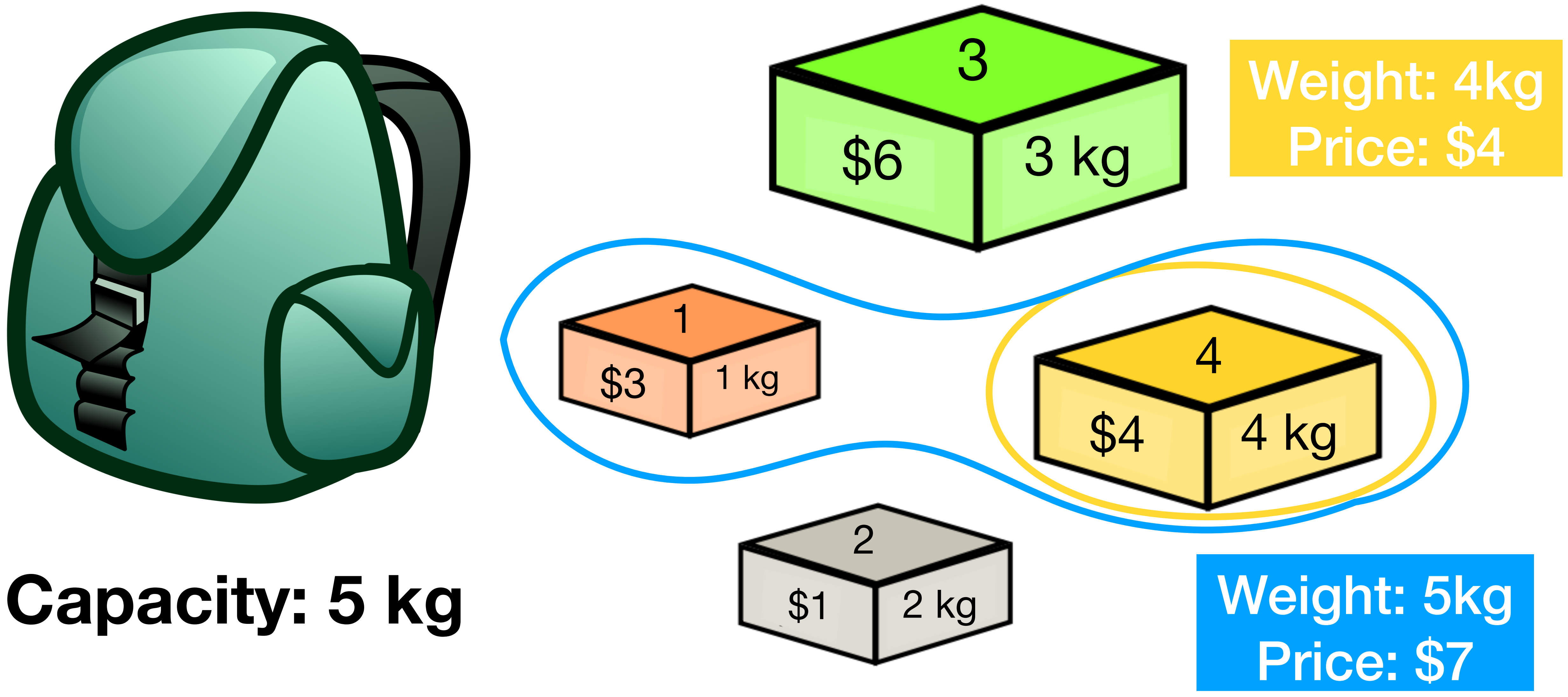
non-solution

Dominance Relations Over Full Assignments

(Chu and Stuckey 2012)

$\bar{\theta}$ dominates $\bar{\theta}'$ ($\bar{\theta} \prec \bar{\theta}'$):

- $\bar{\theta}$ solution, $\bar{\theta}'$ non-solution
- both solutions/non-solutions, $f(\bar{\theta})$ is better than $f(\bar{\theta}')$



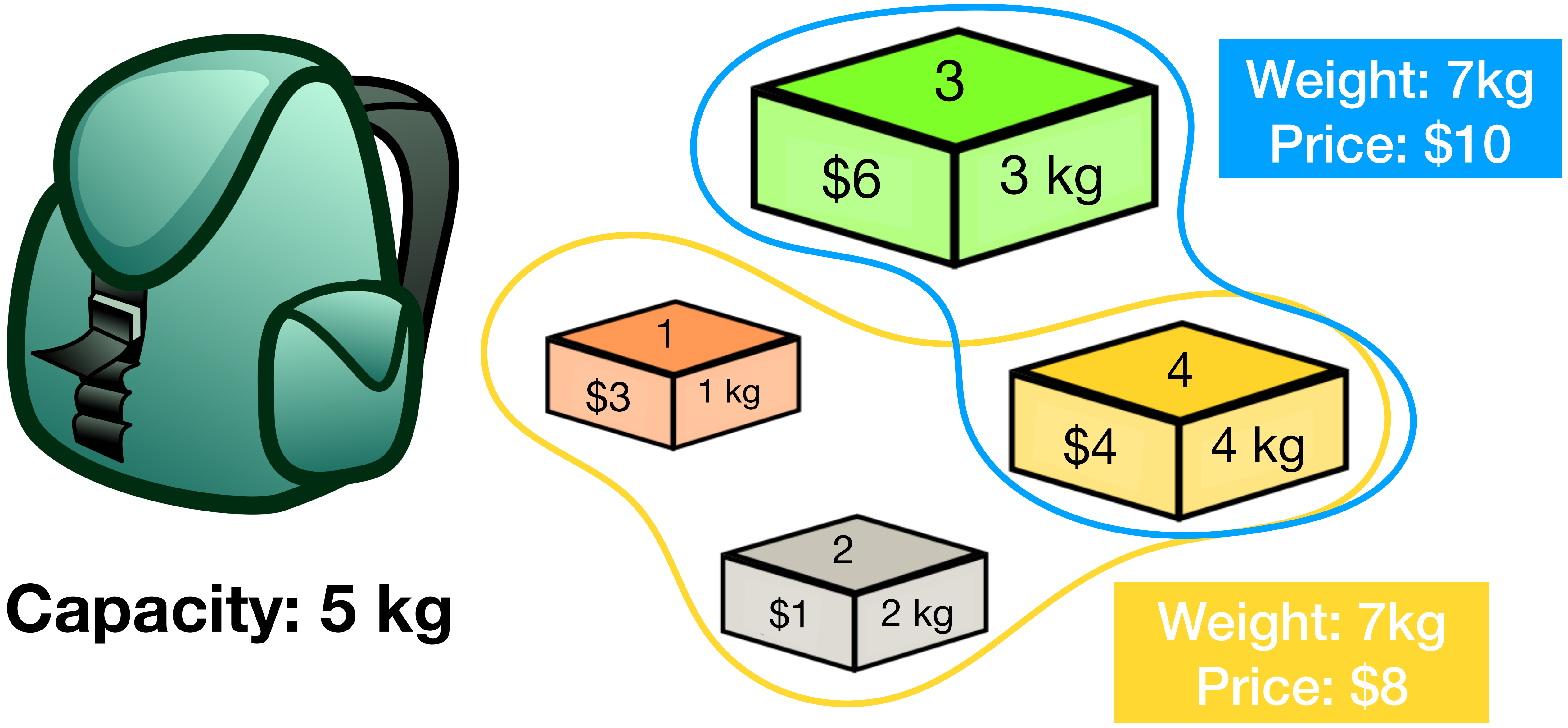
$$\boxed{\bar{\theta} = \{x_1 = 1,\, x_2 = 0,\, x_3 = 0,\, x_4 = 1\}} \prec \boxed{\bar{\theta}' = \{x_1 = 0,\, x_2 = 0,\, x_3 = 0,\, x_4 = 1\}}$$

solution $f(\bar{\theta}) = 7$ $f(\bar{\theta}') = 4$ solution

Dominance Relations Over Full Assignments

(Chu and Stuckey 2012)

if $\bar{\theta}'$ is dominated (by some $\bar{\theta}$), we can safely remove $\bar{\theta}'$



$\bar{\theta} = \{x_1 = 0,\, x_2 = 0,\, x_3 = 1,\, x_4 = 1\}$ $\prec$ $\bar{\theta}' = \{x_1 = 1,\, x_2 = 1,\, x_3 = 0,\, x_4 = 1\}$

non-solution $f(\bar{\theta}) = 10$ $f(\bar{\theta}') = 8$ non-solution

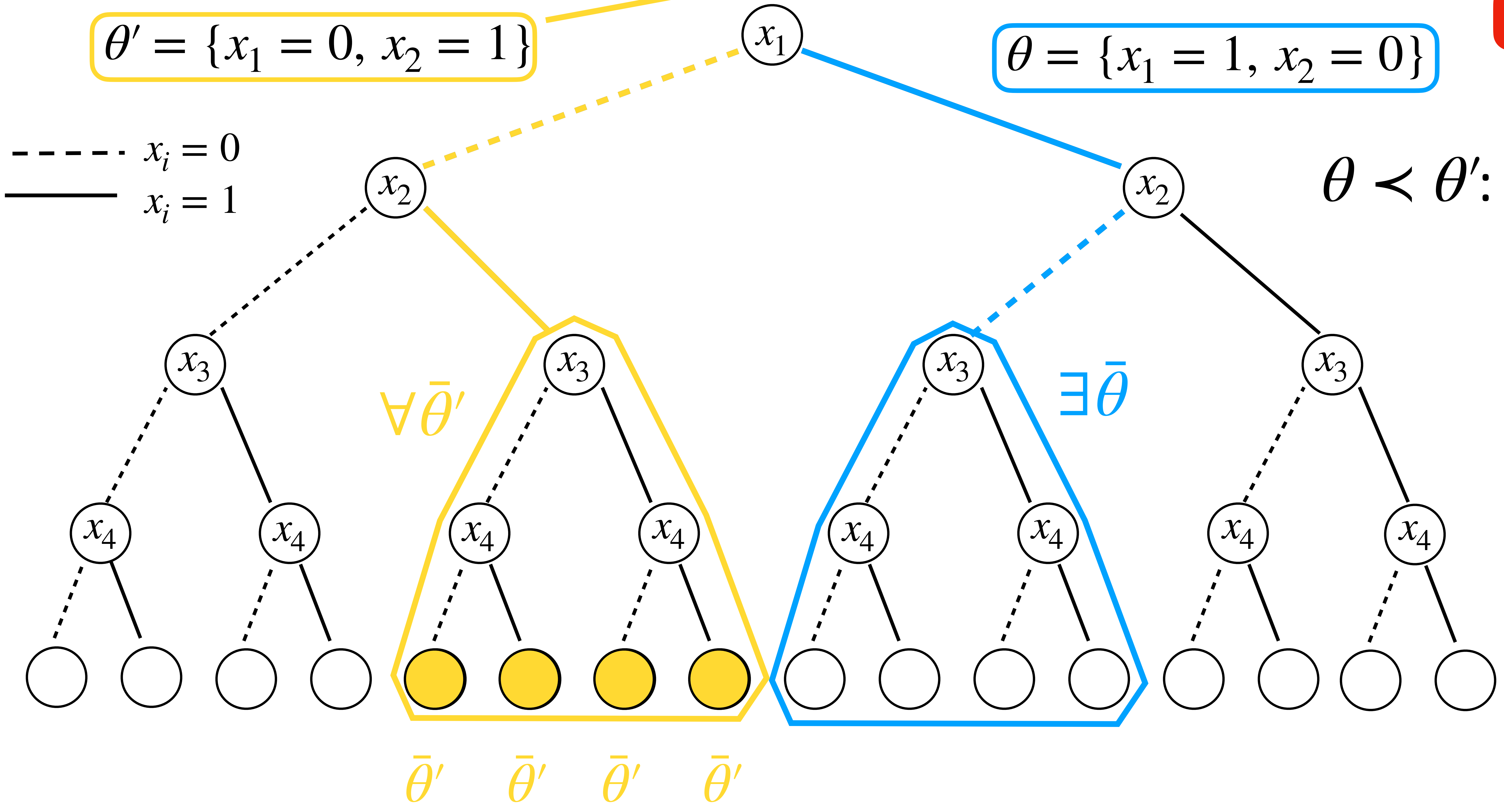
Dominance Relations Over Partial Assignments

(Lee and Zhong 2020)

$\theta' = \{x_1 = 0, x_2 = 1\}$ $\xrightarrow{\text{equivalent}}$ $(x_1 = 0 \land x_2 = 1)$ $\xrightarrow{\text{negation}}$ $\boxed{\neg\theta' \equiv x_1 \neq 0 \lor x_2 \neq 1}$

Dominance Breaking Nogood

$\theta = \{x_1 = 1, x_2 = 0\}$



$\theta \prec \theta'$: $\forall\bar{\theta}'$ extending θ'

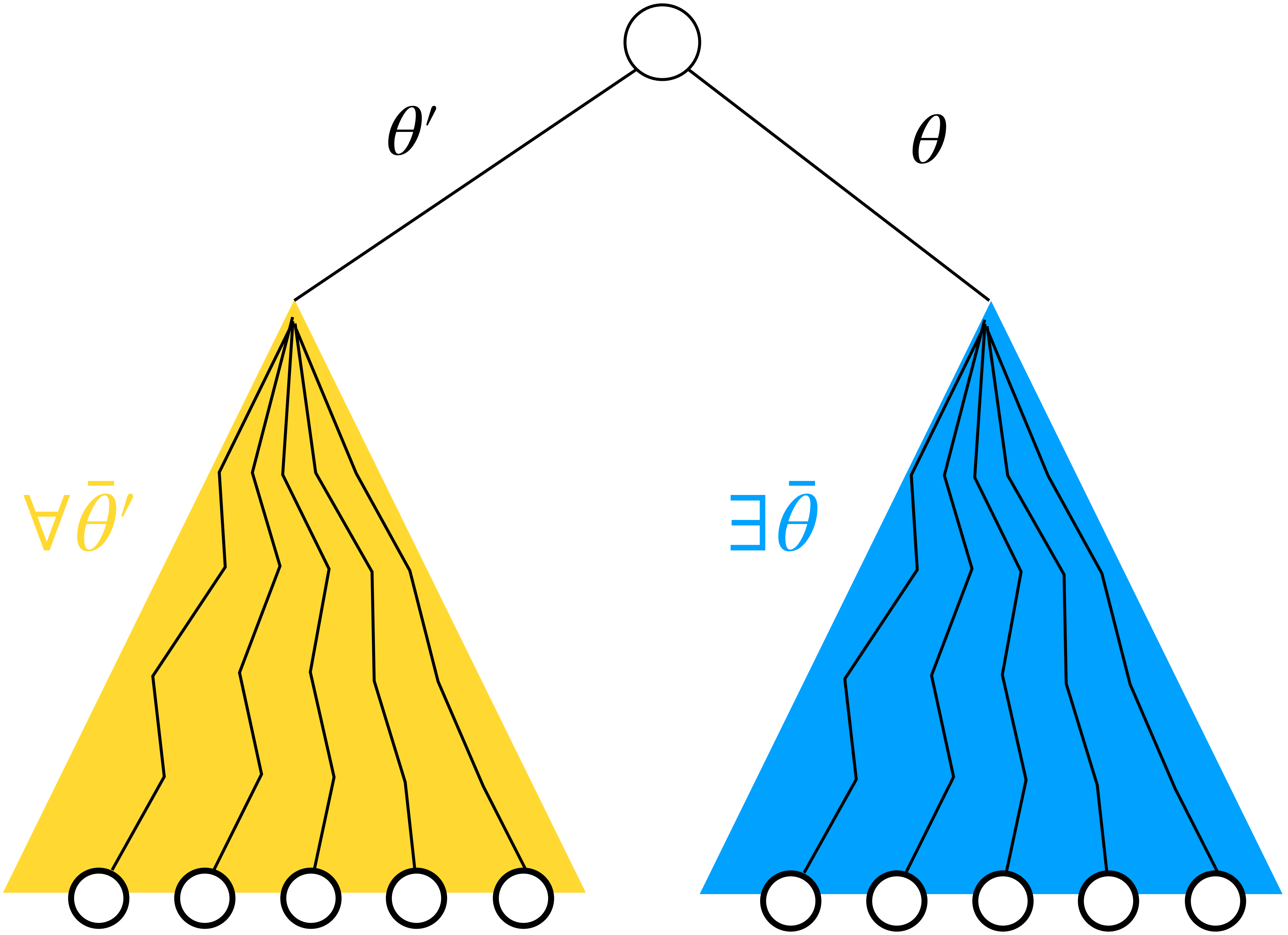
$\exists\bar{\theta}$ extending θ

s.t. $\bar{\theta} \prec \bar{\theta}'$

Theorem:
if θ dominates θ', we can add $\neg\theta'$ to COP

Automatic Dominance Breaking

(Lee and Zhong 2020)



Generation Problems:

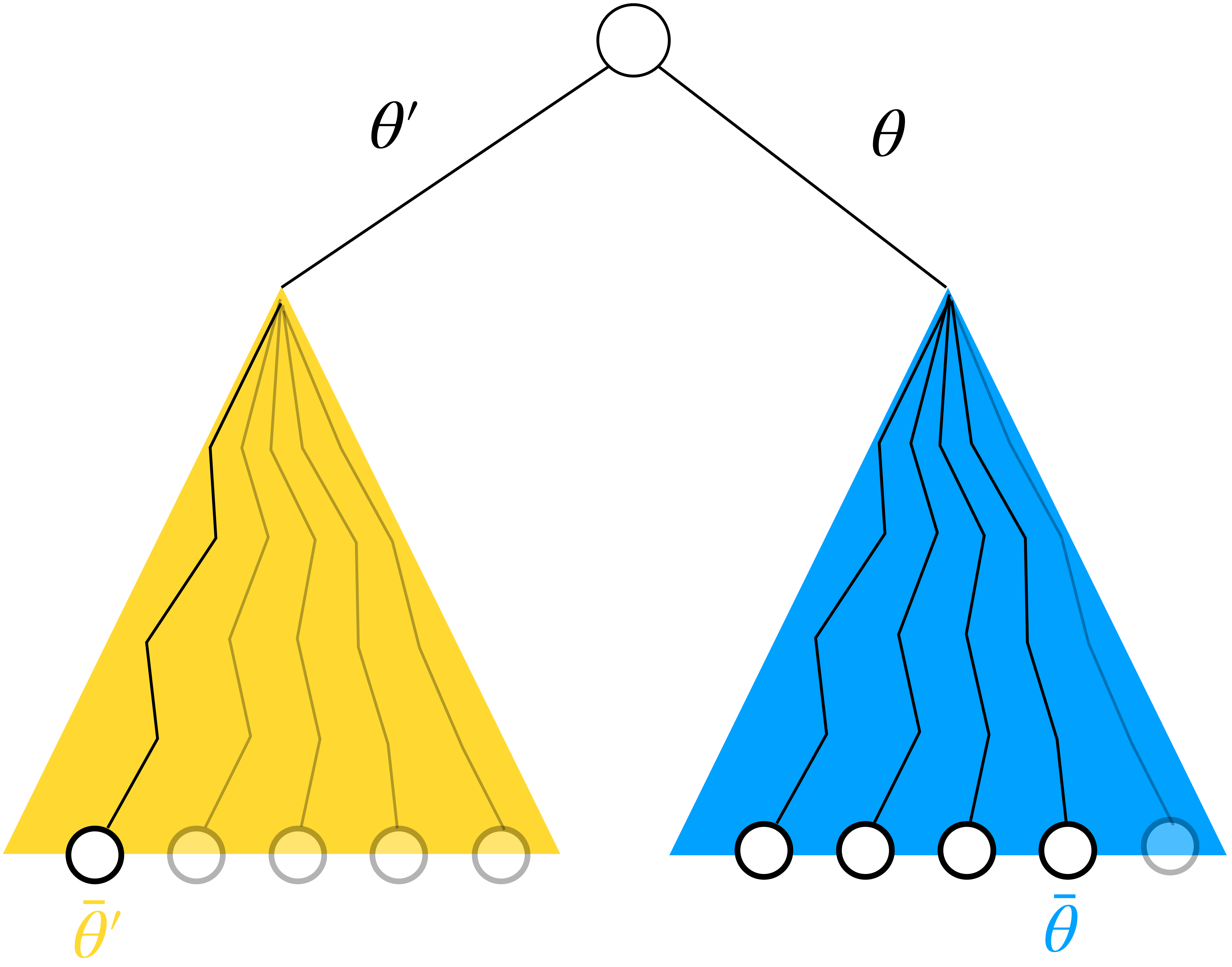
find pairs (θ, θ') such that:

(1) $\theta \prec \theta'$

Want to show $\forall\bar{\theta}'\ \exists\bar{\theta}$ s.t. $\bar{\theta} \prec \bar{\theta}'$

Automatic Dominance Breaking

(Lee and Zhong 2020)



Generation Problems:

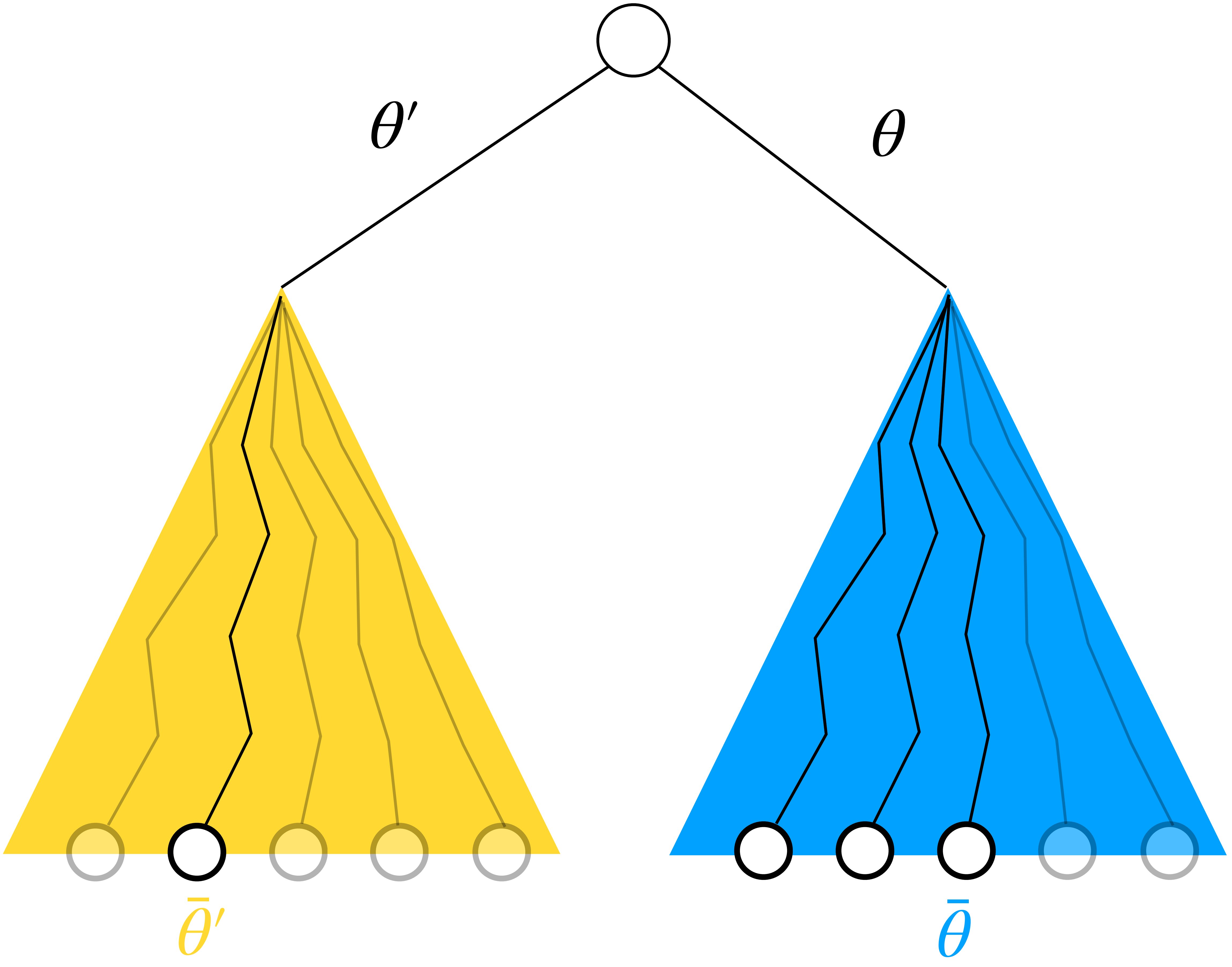
find pairs (θ, θ') such that:

(1) $\theta \prec \theta'$

Want to show $\forall \bar{\theta}' \; \exists \bar{\theta}$ s.t. $\bar{\theta} \prec \bar{\theta}'$

Automatic Dominance Breaking

(Lee and Zhong 2020)



Generation Problems:

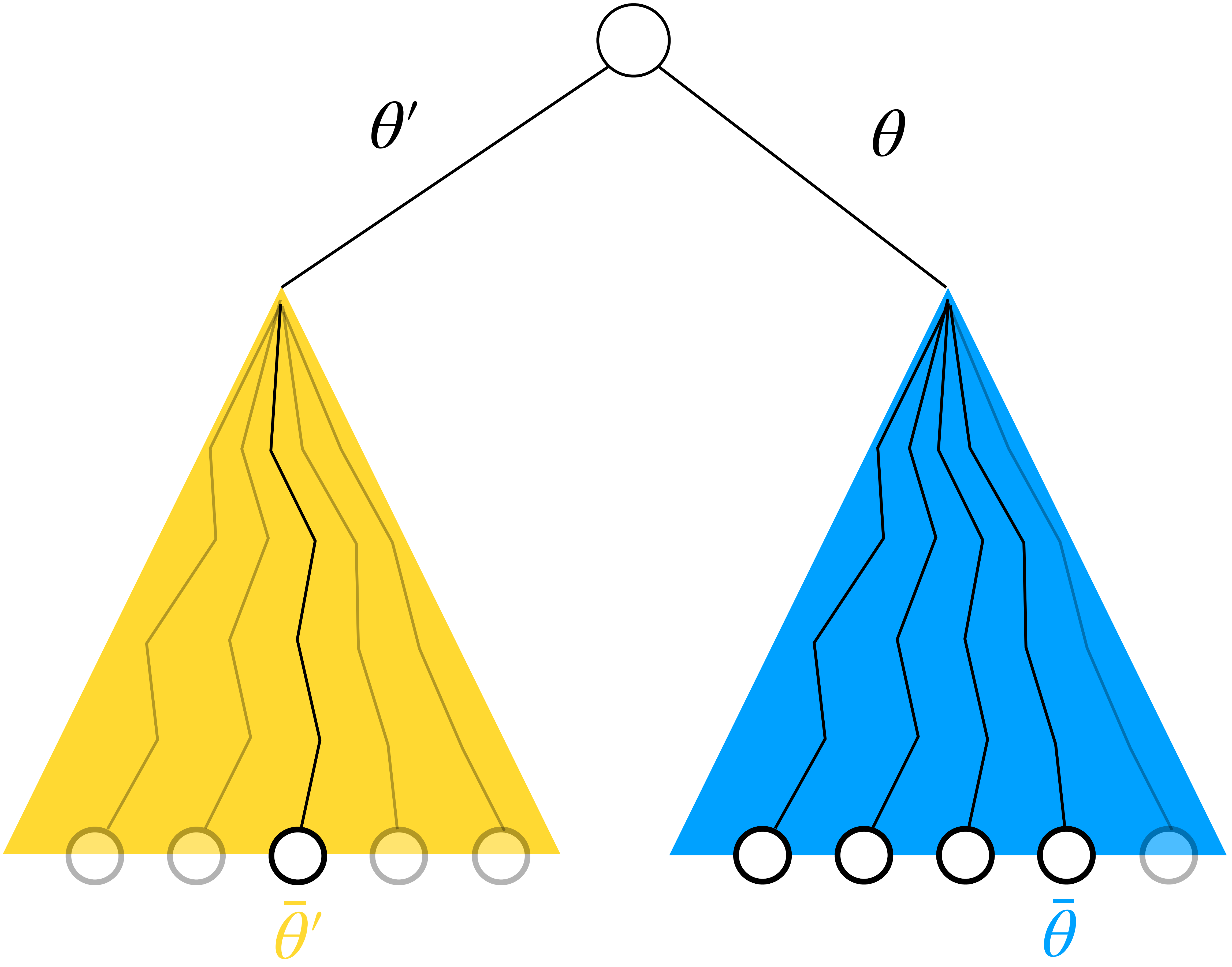
find pairs (θ, θ') such that:

(1) $\theta \prec \theta'$

Want to show $\forall \bar{\theta}'\ \exists \bar{\theta}$ s.t. $\bar{\theta} \prec \bar{\theta}'$

Automatic Dominance Breaking

(Lee and Zhong 2020)



Generation Problems:

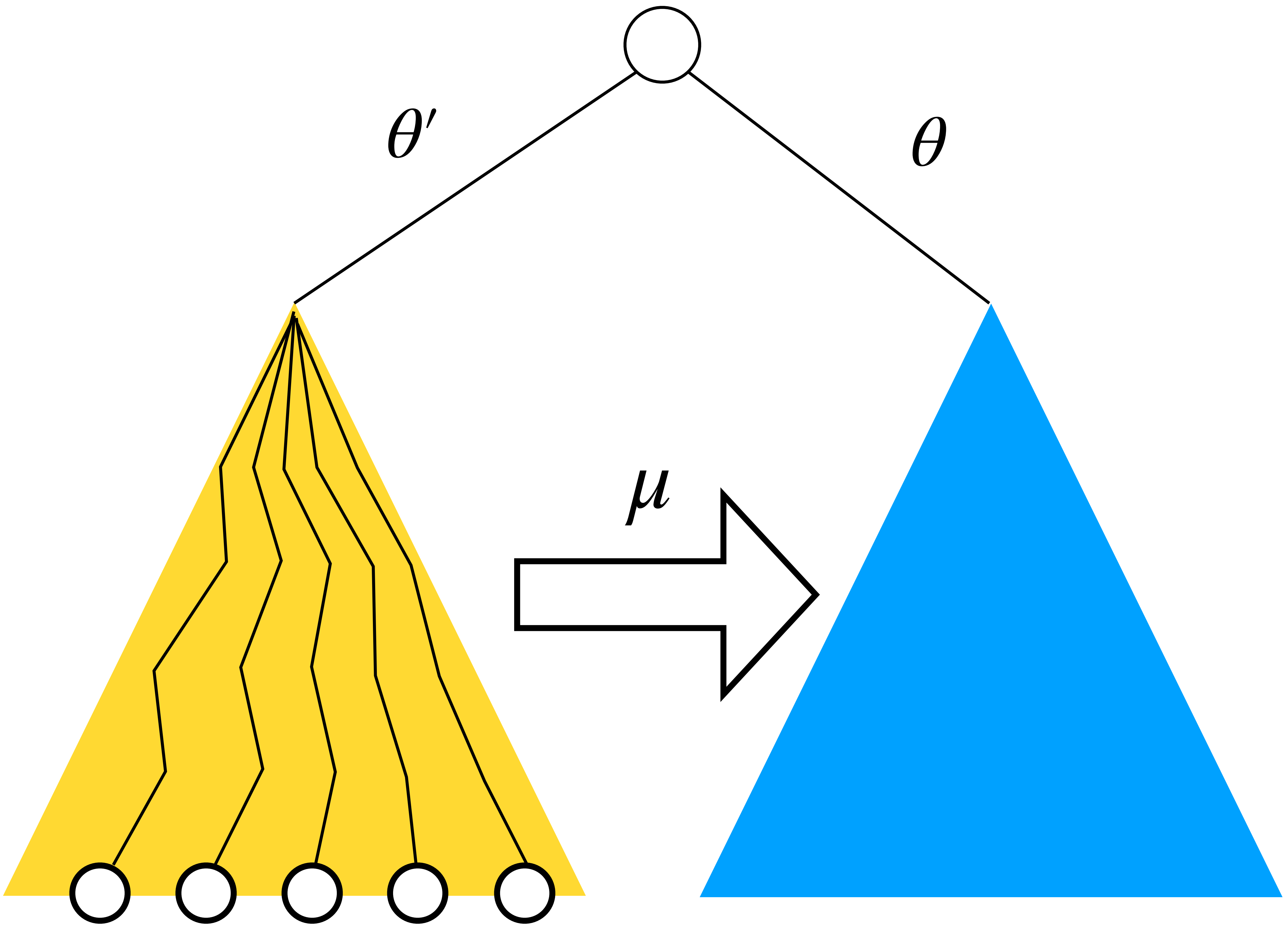
find pairs (θ, θ') such that:

(1) $\theta \prec \theta'$

Want to show $\forall \bar{\theta}'\ \exists \bar{\theta}$ s.t. $\bar{\theta} \prec \bar{\theta}'$

Automatic Dominance Breaking

(Lee and Zhong 2020)



Generation Problems:

find pairs (θ, θ') such that:

(1) $\theta \prec \theta'$

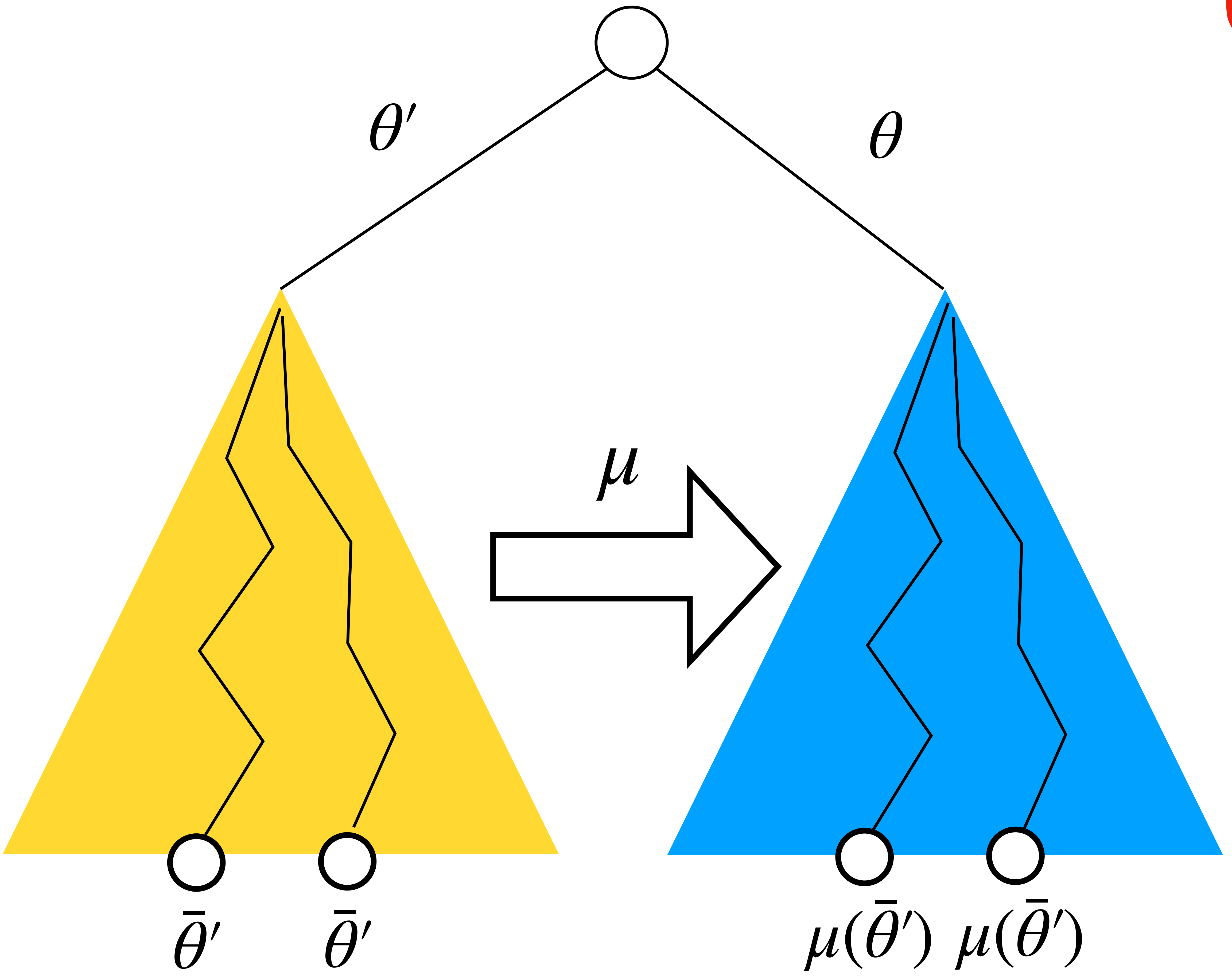
(2) $var(\theta) = var(\theta')$

Want to show $\forall \bar{\theta}'\ \exists \bar{\theta}$ s.t. $\bar{\theta} \prec \bar{\theta}'$

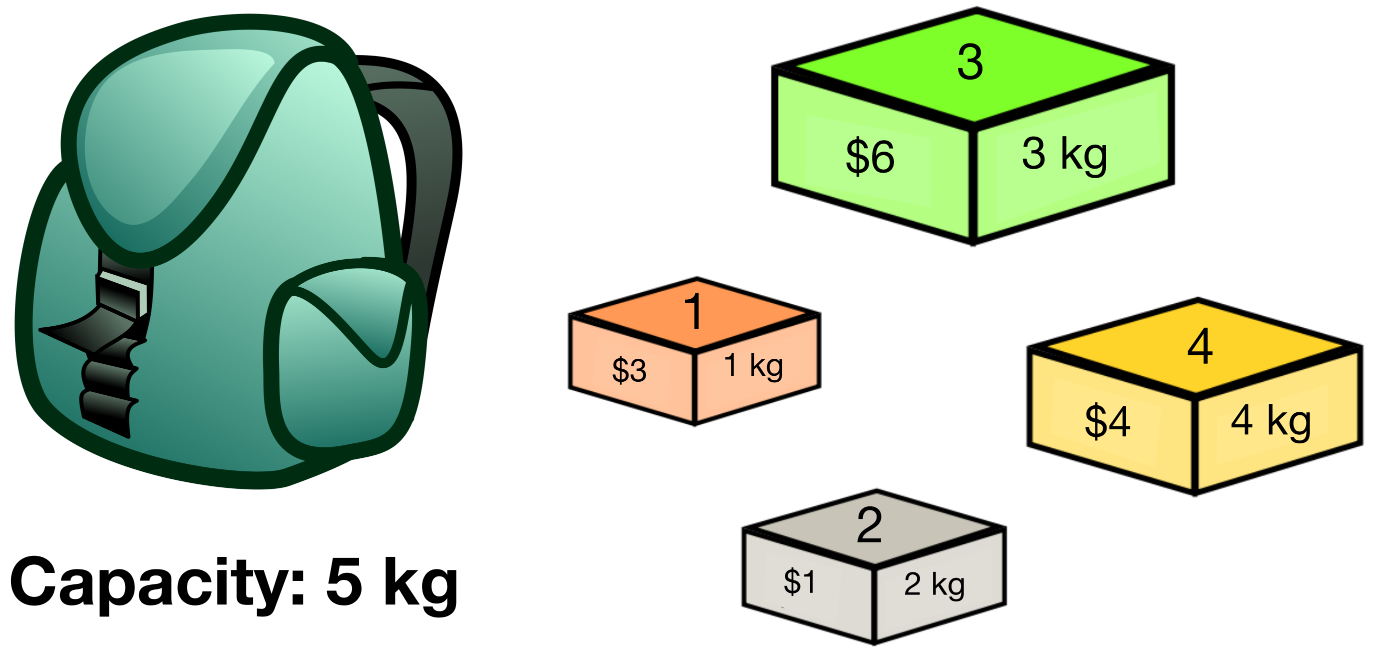
Suffice to show $\forall \bar{\theta}'$, $\mu(\bar{\theta}') \prec \bar{\theta}'$

Mutation Mapping

(Lee and Zhong 2020)



Mutation mapping: $\mu(\bar{\theta}')$ extends θ **in the same way** $\bar{\theta}'$ extends θ'



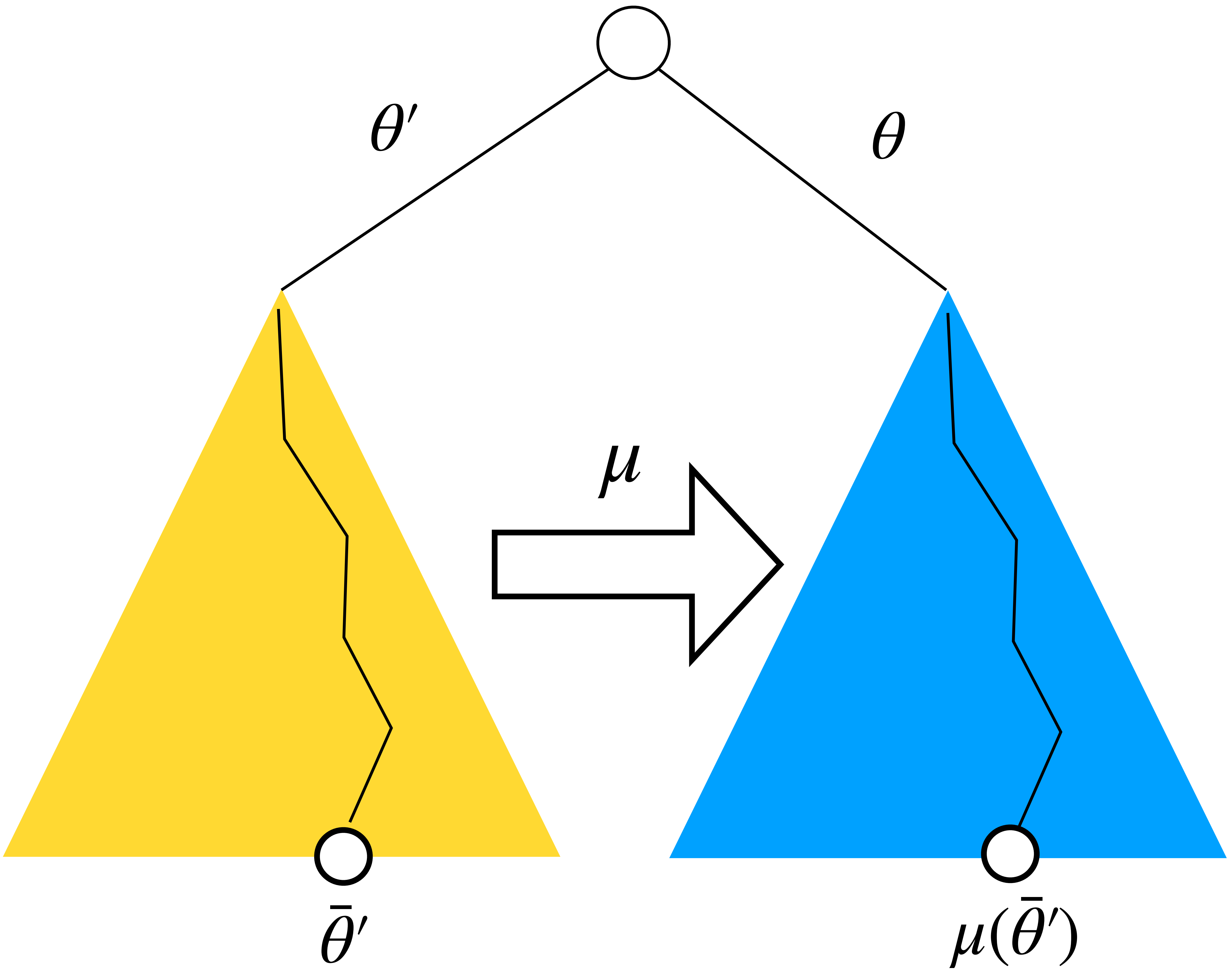
Example: Suppose $a, b, c, d \in \{0,1\}$

$\theta' = \{x_1 = a,\ x_2 = b\}$ $\qquad$ $\theta = \{x_1 = c,\ x_2 = d\}$

x1	x2	x3	x4	μ	x1	x2	x3	x4
a	b	0	0	$\mapsto$	c	d	0	0
a	b	0	1	$\mapsto$	c	d	0	1
a	b	1	0	$\mapsto$	c	d	1	0
a	b	1	1	$\mapsto$	c	d	1	1

Automatic Dominance Breaking

(Lee and Zhong 2020)



Mutation mapping: $\mu(\bar{\theta}')$ extends θ **<u>in the same way</u>** $\bar{\theta}'$ extends θ'

Theorem: $\theta \prec \theta'$ if

- Not equal: $\theta \neq \theta'$
- Betterment:
  $\forall \bar{\theta}', f(\mu(\bar{\theta}'))$ is better than $f(\bar{\theta}')$
- Implied satisfaction:
  $\forall \bar{\theta}', \bar{\theta}'$ solution $\Rightarrow \mu(\bar{\theta}')$ solution

Constraints over (θ, θ') !

Automatic Dominance Breaking

(Lee and Zhong 2020)

- Automatic dominance breaking is enabled for a class of COPs.

Efficiently Checkable Objectives	**Efficiently Checkable Constraints**
• Separable objectives • Submodular set objectives	• Domain constraints • Boolean disjunction constraints • Linear inequality constraints • Counting family constraints

Modelling

(Lee and Zhong 2020)

```
int: n;       % number of items
int: W;       % knapsack capacity
array [1..n] of int: w;  % weight of each item
array [1..n] of int: v;  % value of each item

array [1..n] of var 0..1: x;

% constraint
constraint sum (i in 1..n) (w[i]*x[i]) <= W;

% objective
solve maximize sum (i in 1..n) (v[i]*x[i]);
```

COP MiniZinc Model

```
int: n;       % number of items
int: W;       % knapsack capacity
array [1..n] of int: w;  % weight of each item
array [1..n] of int: v;  % value of each item
int: k; % size of partial assignments
```

Generation CSP MiniZinc Model

Modelling

(Lee and Zhong 2020)

```
int: n;       % number of items
int: W;       % knapsack capacity
array [1..n] of int: w;  % weight of each item
array [1..n] of int: v;  % value of each item

array [1..n] of var 0..1: x;

% constraint
constraint sum (i in 1..n) (w[i]*x[i]) <= W;

% objective
solve maximize sum (i in 1..n) (v[i]*x[i]);
```

COP MiniZinc Model

```
int: n;       % number of items
int: W;       % knapsack capacity
array [1..n] of int: w;  % weight of each item
array [1..n] of int: v;  % value of each item
int: k; % size of partial assignments

array [1..k] of var 1..n: F; % indices for fixed variable
array [1..k] of var 0..1: v1; % fixed value for \theta
array [1..k] of var 0..1: v2; % fixed value for \theta'
constraint increasing(F); % symmetry breaking
constraint lex_less(v1,v2); % compatibility and not equal
```

Generation CSP MiniZinc Model

Modelling

(Lee and Zhong 2020)

```
int: n;         % number of items
int: W;         % knapsack capacity
array [1..n] of int: w;  % weight of each item
array [1..n] of int: v;  % value of each item

array [1..n] of var 0..1: x;

% constraint
constraint sum (i in 1..n) (w[i]*x[i]) <= W;

% objective
solve maximize sum (i in 1..n) (v[i]*x[i]);
```

COP MiniZinc Model

```
int: n;         % number of items
int: W;         % knapsack capacity
array [1..n] of int: w;   % weight of each item
array [1..n] of int: v;   % value of each item
int: k; % size of partial assignments

array [1..k] of var 1..n: F; % indices for fixed variable
array [1..k] of var 0..1: v1; % fixed value for \theta
array [1..k] of var 0..1: v2; % fixed value for \theta'
constraint increasing(F); % symmetry breaking
constraint lex_less(v1,v2); % compatibility and not equal

% constraint for implied satisfaction
constraint sum(t in 1..k)( w[F[t]] * v1[t] )
           <=  sum(t in 1..k)( w[F[t]] * v2[t] );
```

Generation CSP MiniZinc Model

Modelling

(Lee and Zhong 2020)

```
int: n;        % number of items
int: W;        % knapsack capacity
array [1..n] of int: w;  % weight of each item
array [1..n] of int: v;  % value of each item

array [1..n] of var 0..1: x;

% constraint
constraint sum (i in 1..n) (w[i]*x[i]) <= W;

% objective
solve maximize sum (i in 1..n) (v[i]*x[i]);
```

COP MiniZinc Model

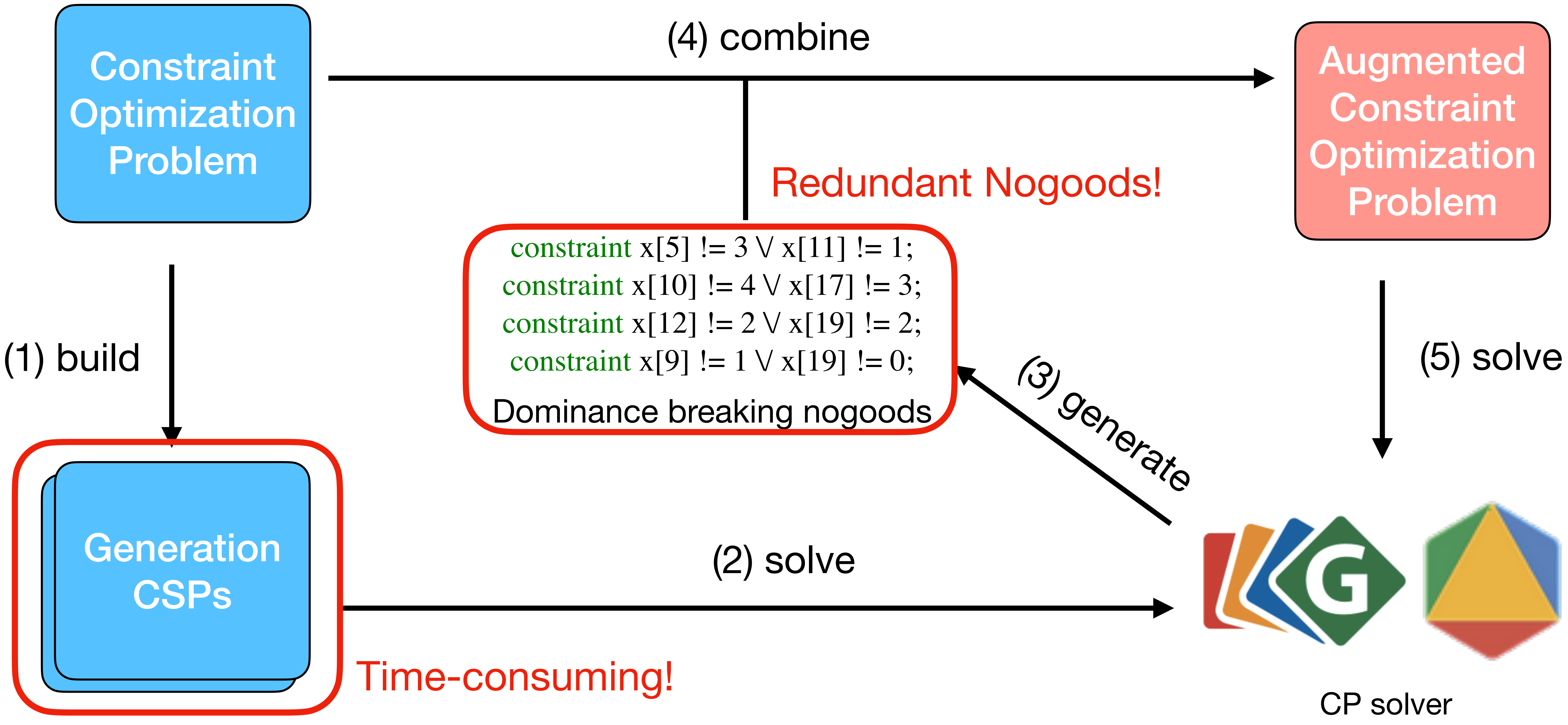
```
int: n;        % number of items
int: W;        % knapsack capacity
array [1..n] of int: w;   % weight of each item
array [1..n] of int: v;   % value of each item
int: k; % size of partial assignments

array [1..k] of var 1..n: F; % indices for fixed variable
array [1..k] of var 0..1: v1; % fixed value for \theta
array [1..k] of var 0..1: v2; % fixed value for \theta'
constraint increasing(F); % symmetry breaking
constraint lex_less(v1,v2); % compatibility and not equal

% constraint for implied satisfaction
constraint sum(t in 1..k)( w[F[t]] * v1[t] )
             <=  sum(t in 1..k)( w[F[t]] * v2[t] );

% constraint for betterment
constraint sum(t in 1..k)( v[F[t]] * v1[t] )
             >= sum(t in 1..k)( v[F[t]] * v2[t] );
```

Generation CSP MiniZinc Model

Common Assignment Elimination

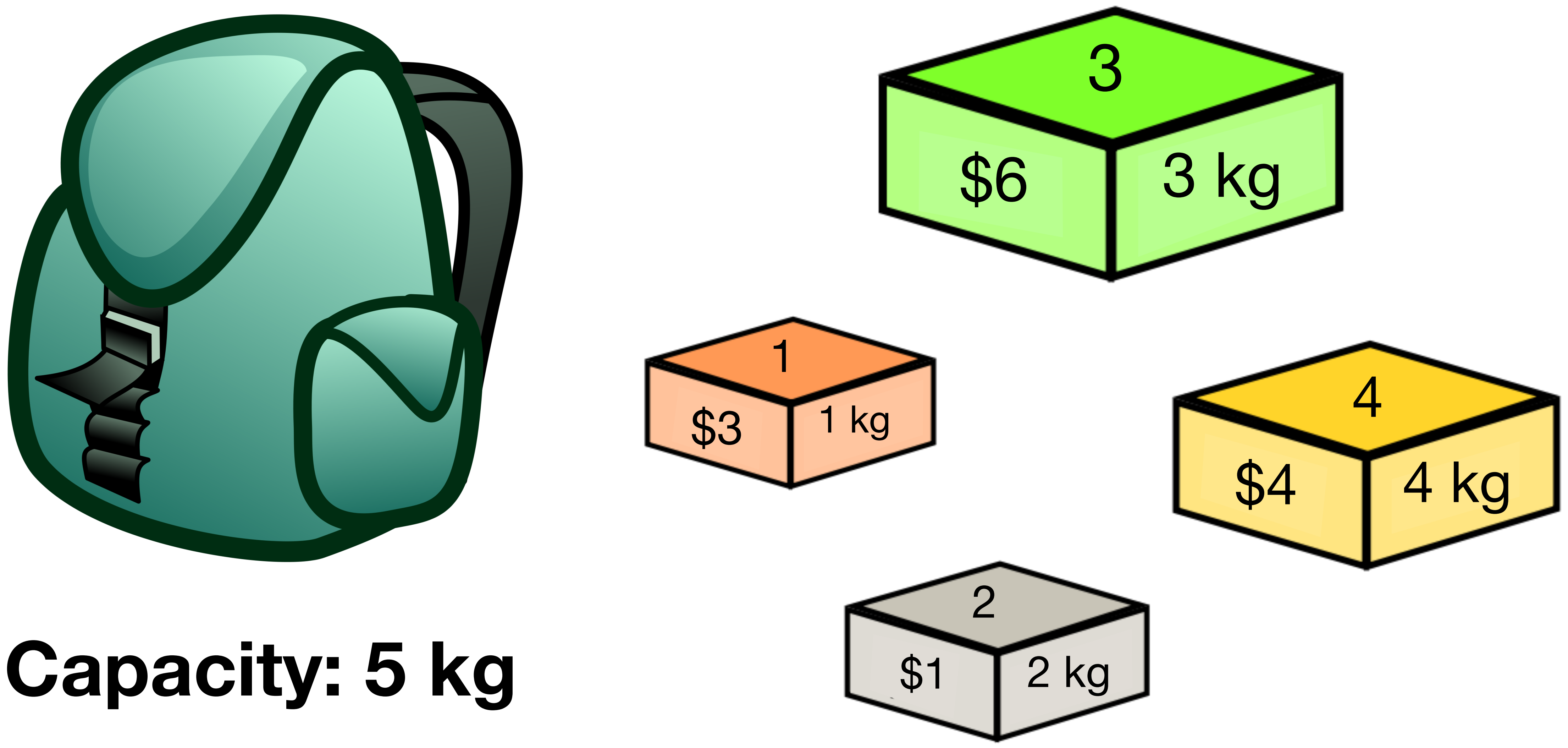
(Lee and Zhong 2021)

- Automatic dominance breaking is not efficient enough.



Common Assignment Elimination

(Lee and Zhong 2021)



Capacity: 5 kg

maximize $x_1 + 2x_2 + 4x_3 + 10x_4$

s.t. $x_1 + 2x_2 + 3x_3 + 4x_4 \leq 5$

$x_i \in \{0,1\}$ for $i = 1,\ldots,4$

$\cancel{c_1 \equiv (\boxed{x_2 \neq 1 \vee x_4 \neq 0} \vee x_3 \neq 1)}$

$c_2 \equiv (\boxed{x_2 \neq 1 \vee x_4 \neq 0})$

Avoid generating c_1 by common assignment elimination

$c_2 \Rightarrow c_1$

c_1 is also propagation redundant!

Exploiting Functional Constraints

(Lee and Zhong 2022)

- Automatic dominance breaking is enabled only for a class of COPs.

Efficiently Checkable Objectives	Efficiently Checkable Constraints
• Separable objectives • Submodular set objectives	• Domain constraints • Boolean disjunction constraints • Linear inequality constraints • Counting family constraints

- Impractical restriction on efficiently checkable objectives and constraints.

Exploiting Functional Constraints

(Lee and Zhong 2022)

- Automatic dominance breaking is enabled only for a class of COPs.
- Constraint programming provides a flexible modelling language which can form various objectives and constraints.
  - Example:

$$\text{minimize } \boxed{\max(x_1, x_2)} + 4x_3$$

$$\text{s.t. } 2x_1 - \boxed{3x_2 * x_3} \leq 5$$

$$x_i \in \{0,1\} \text{ for } i = 1,2,3$$

Not efficiently checkable!

- Unknown constraints for betterment and implied satisfaction in generation CSPs

Exploiting Functional Constraints

(Lee and Zhong 2022)

- COPs specified in a modelling language are normalised/flattened into a form with only standard constraints from the underlying solver.
  - Example:

minimize $\max(x_1, x_2) + 4x_3$

s.t. $2x_1 - 3x_2 * x_3 \leq 5$

$x_i \in \{0,1\}$ for $i = 1,2,3$

normalised $\longrightarrow$

Functional constraints

minimize obj

s.t. $\boxed{obj = y_1 + 4x_3,\ y_1 = \max(x_1, x_2)}$

$y_2 \leq 5, \boxed{y_2 = 2x_1 - 3y_3,\ y_3 = x_2 * x_3}$

$x_i \in \{0,1\}$ for $i = 1,2,3$

Auxiliary variables $\boxed{y_1, y_2, y_3, obj \in \mathbb{Z}}$

- Motivation: exploit standard functional constraints from CP solvers

Experimental Settings

- Modify the compiler for the MiniZinc modelling language
  - Available online: https://github.com/AllenZzw/auto-dom
- Experiment on talent scheduling, maximum coverage, sensor placement
  - Chuffed for problem-solving, Geas for nogood generation
  - 20 random instances for each problem size
  - 2 hours total timeout; reserve 1 hour for nogood generation.

Experimental Evaluation

- Geometric mean of time (seconds) for problem of different sizes:

Problem	Basic	2-dom		3-dom		4-dom	
		Solving	Total	Solving	Total	Solving	Total
Team-6-5	24.48	10.57	**12.49**	9.70	32.00	8.88	427.73
Team-7-5	276.84	138.88	**146.15**	130.71	225.19	150.83	1745.96
Team-8-5	1983.53	819.58	**829.05**	767.52	1024.43	724.63	5191.70
MaxCover-45	75.91	53.47	53.79	5.07	**9.96**	0.27	83.93
MaxCover-50	615.04	464.81	465.53	26.31	**34.92**	1.12	134.99
MaxCover-55	3576.98	2859.60	2860.27	78.37	**91.53**	2.54	199.11
Sensor-50	156.84	138.65	139.44	94.05	**108.99**	57.34	297.18
Sensor-60	595.46	404.27	405.52	269.61	**296.56**	172.43	709.37
Sensor-70	1615.18	1144.17	1145.83	810.01	**854.61**	651.72	1724.70

Concluding Remarks

- Automatic dominance breaking
  - Generating dominance breaking nogoods as constraint satisfaction
  - Automatically derive sufficient conditions in generation CSPs
- Future work
  - Nogood generation from constraint models alone
  - Dynamic generation of dominance breaking nogoods

Thanks!

Experimental Evaluation

- Geometric mean of time (seconds) for problem of different sizes:

Problem	Basic	Manual	2-dom		3-dom		4-dom	
			Solving	Total	Solving	Total	Solving	Total
Talent-16	187.79	5929.75	189.95	192.16	130.78	**148.91**	256.46	1988.75
Talent-18	1575.51	7200.0	1565.89	1568.29	672.26	**713.55**	1864.68	5760.68
Talent-20	5013.10	7200.0	4936.18	4960.54	2856.33	**2960.09**	3268.72	7006.10
Warehouse-35	7200.0	N/A	10.29	**52.11**	8.53	2442.71	8.51	3619.87
Warehouse-40	7200.0	N/A	46.08	**111.43**	32.93	3652.15	32.55	3657.33
Warehouse-45	7200.0	N/A	69.41	**140.92**	45.45	3690.84	46.19	3694.63
Team-6-5	24.48	N/A	10.57	**12.49**	9.70	32.00	8.88	427.73
Team-7-5	276.84	N/A	138.88	**146.15**	130.71	225.19	150.83	1745.96
Team-8-5	1983.53	N/A	819.58	**829.05**	767.52	1024.43	724.63	5191.70

Experimental Evaluation

- Geometric mean of time (seconds) for problem of different sizes:

Problem	Basic	Manual	2-dom		3-dom		4-dom	
			Solving	Total	Solving	Total	Solving	Total
MaxCover-45	75.91	N/A	53.47	53.79	5.07	**9.96**	0.27	83.93
MaxCover-50	615.04	N/A	464.81	465.53	26.31	**34.92**	1.12	134.99
MaxCover-55	3576.98	N/A	2859.60	2860.27	78.37	**91.53**	2.54	199.11
PartialCover-45	2383.2	N/A	366.17	368.03	59.44	**70.64**	2.49	90.25
PartialCover-50	3769.26	N/A	780.80	781.73	74.86	**88.45**	6.86	153.90
PartialCover-55	4640.06	N/A	1769.31	1770.42	211.83	**234.41**	15.23	240.68
Sensor-50	156.84	N/A	138.65	139.44	94.05	**108.99**	57.34	297.18
Sensor-60	595.46	N/A	404.27	405.52	269.61	**296.56**	172.43	709.37
Sensor-70	1615.18	N/A	1144.17	1145.83	810.01	**854.61**	651.72	1724.70