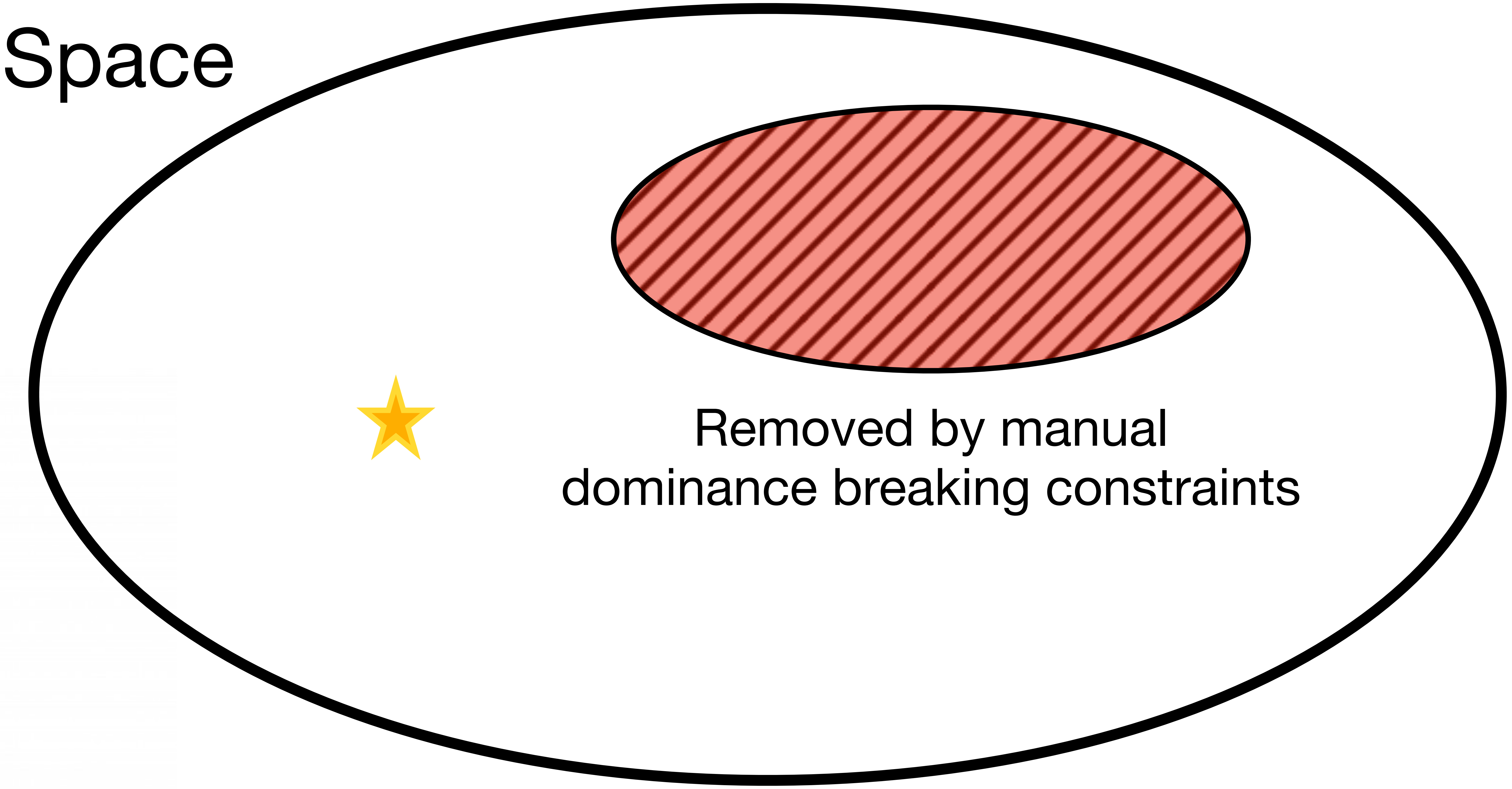
# Towards More Practical and Efficient Automatic Dominance Breaking

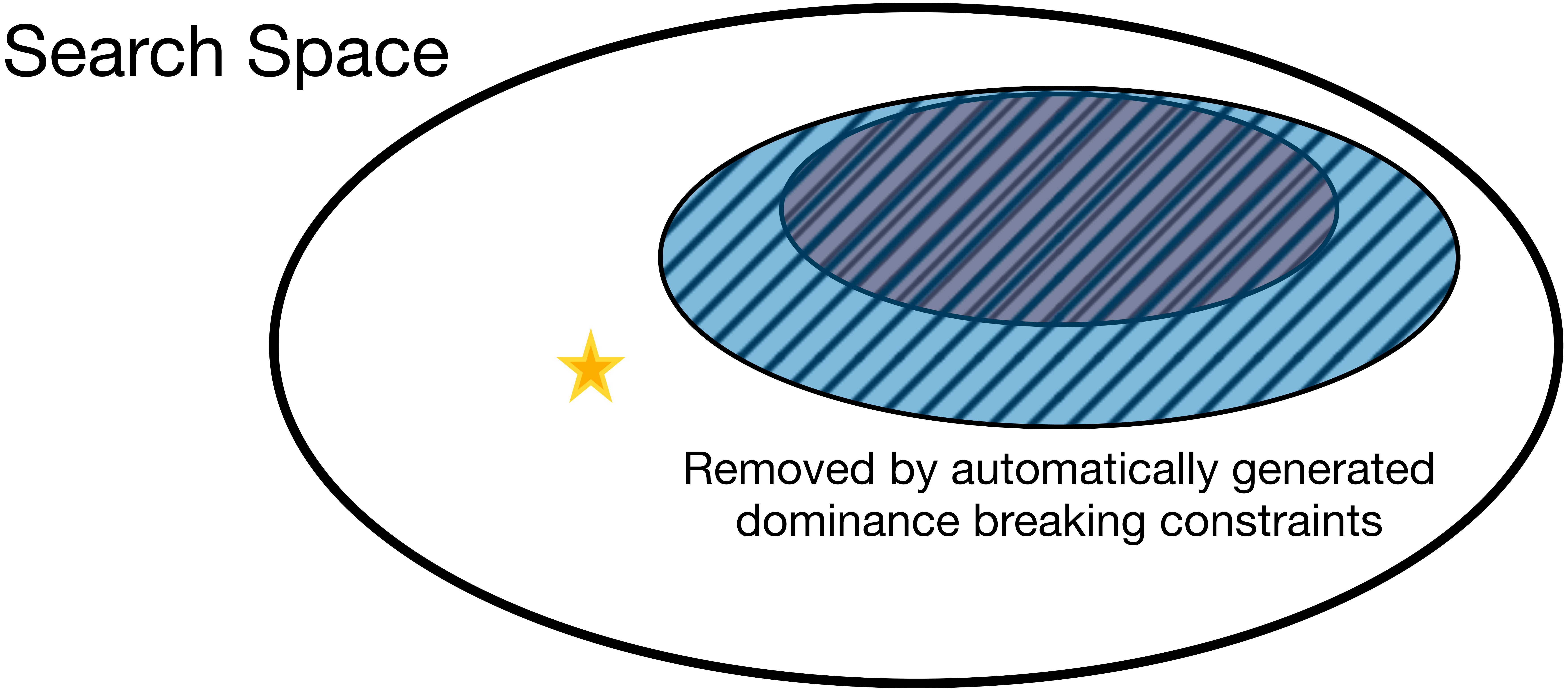
Jimmy H.M. Lee and Allen Z. Zhong
Department of Computer Science and Engineering
The Chinese University of Hong Kong

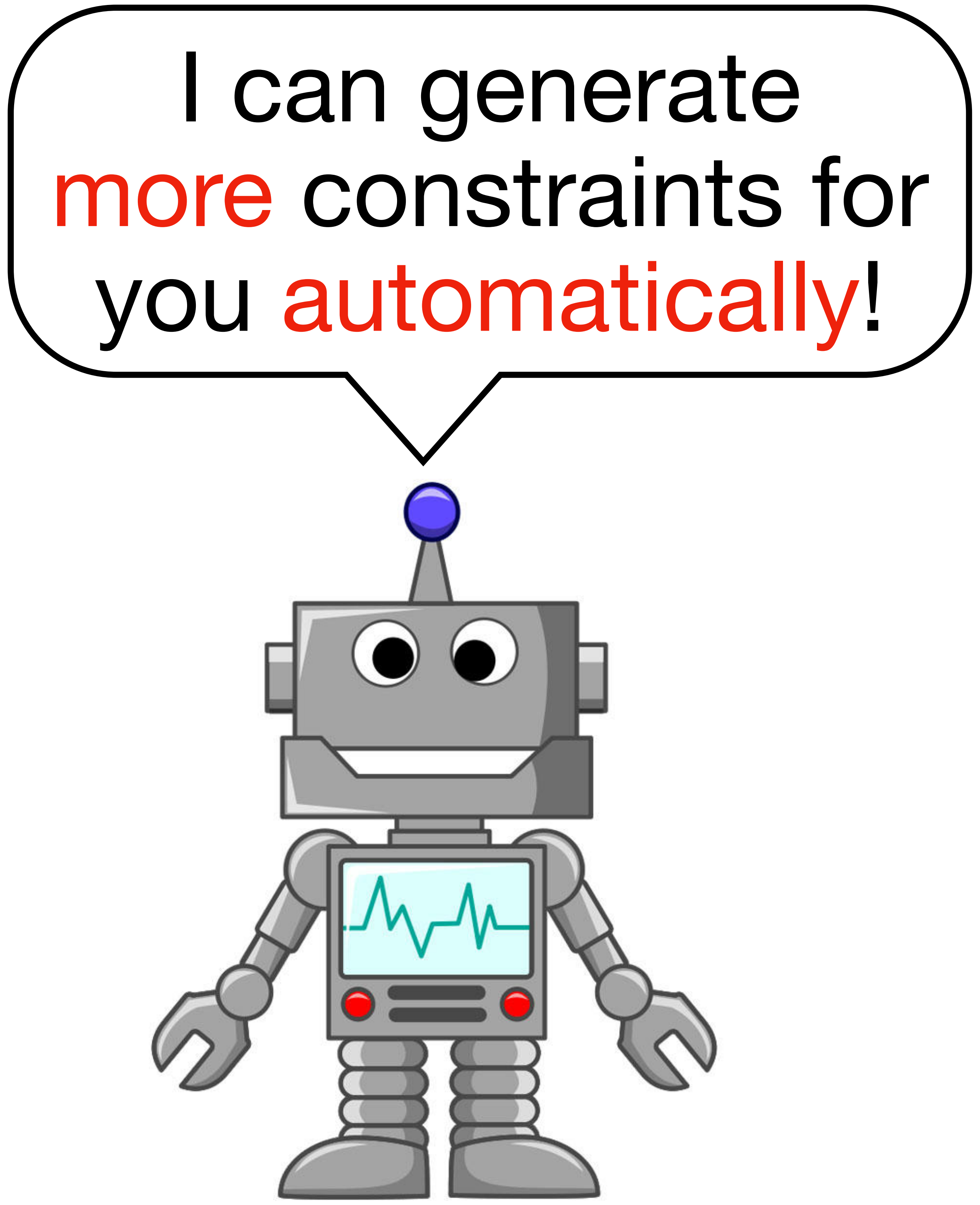
# Dominance Breaking

Search Space




# Dominance Breaking



# Contributions

- Solving larger class of constraint optimization problems
- More efficient generation of dominance breaking constraints

No dominance breaking

Manual dominance breaking

Our previous work

This paper

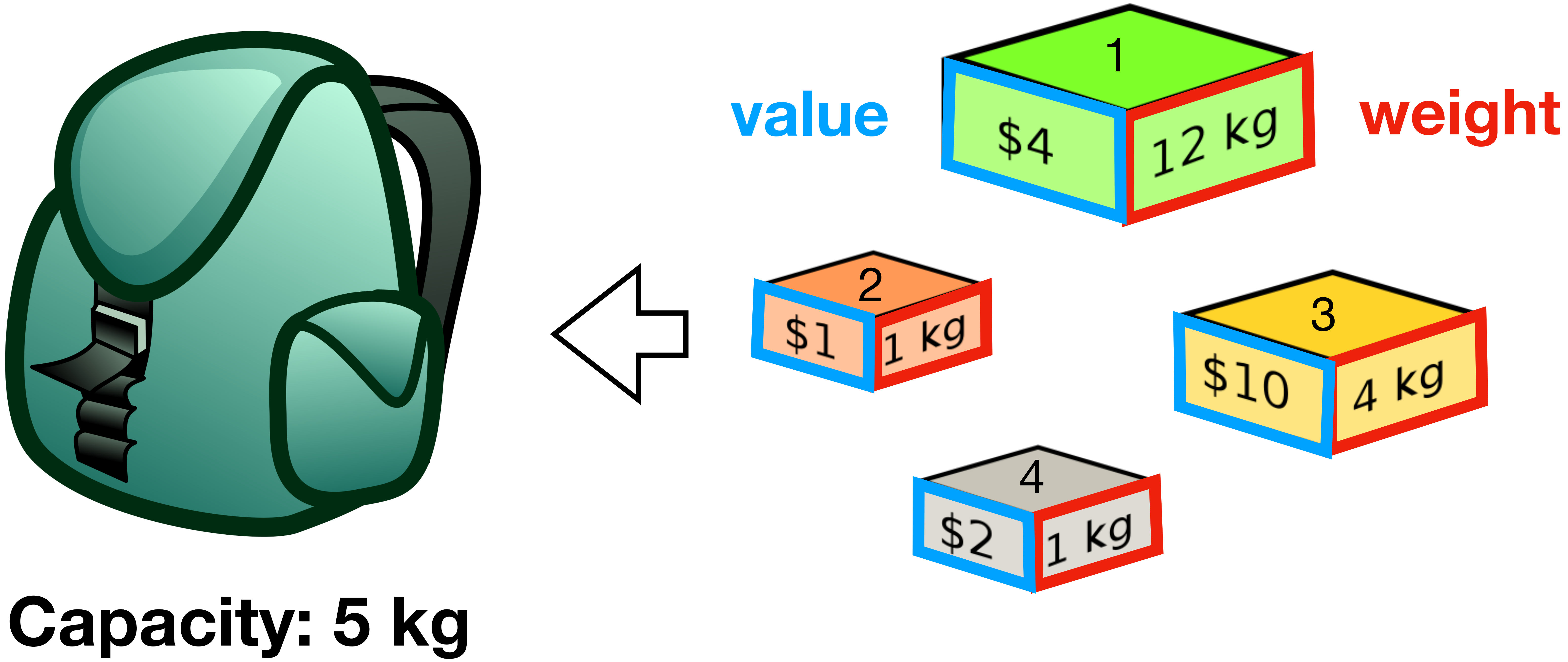
Image source: VectorStock.com/22395696

# Outline

- Automatic Dominance Breaking
- Non-efficiently Checkable Constraints
- Common Assignment Elimination
- Experimental Results

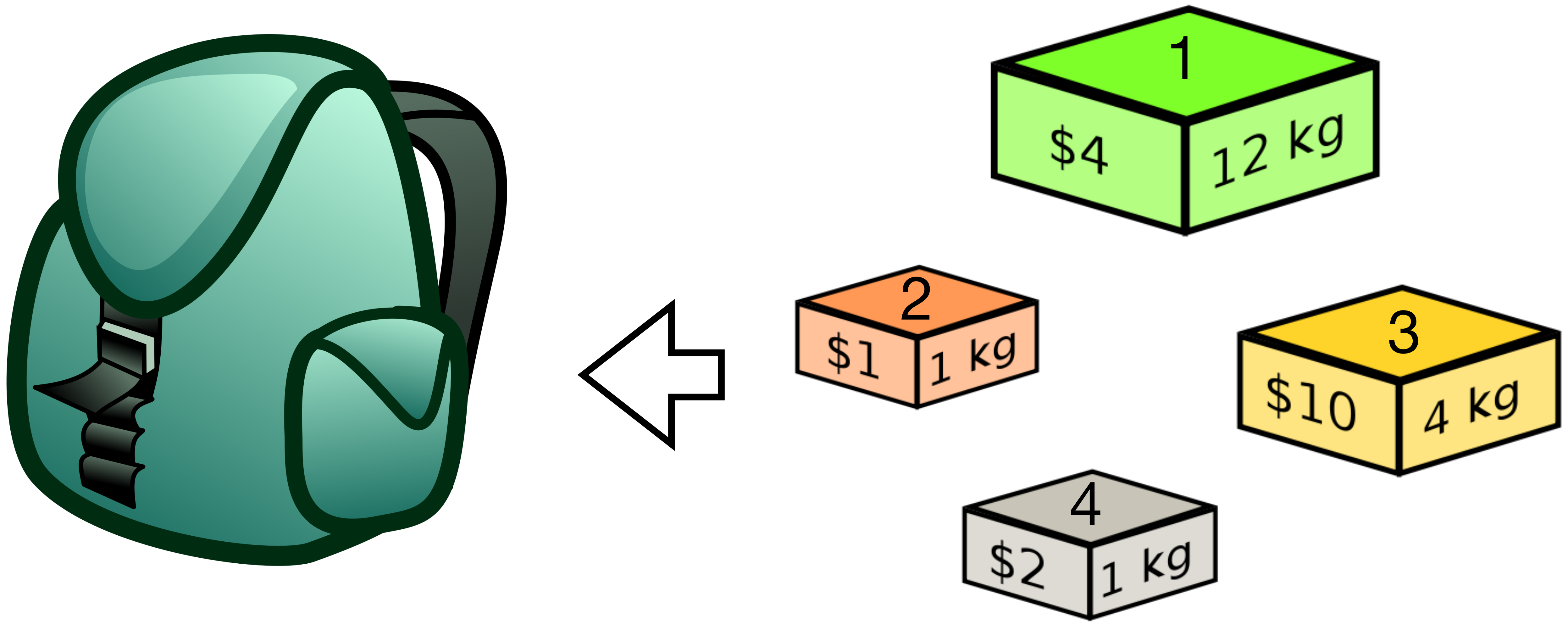
# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.



# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.




**Capacity: 5 kg**

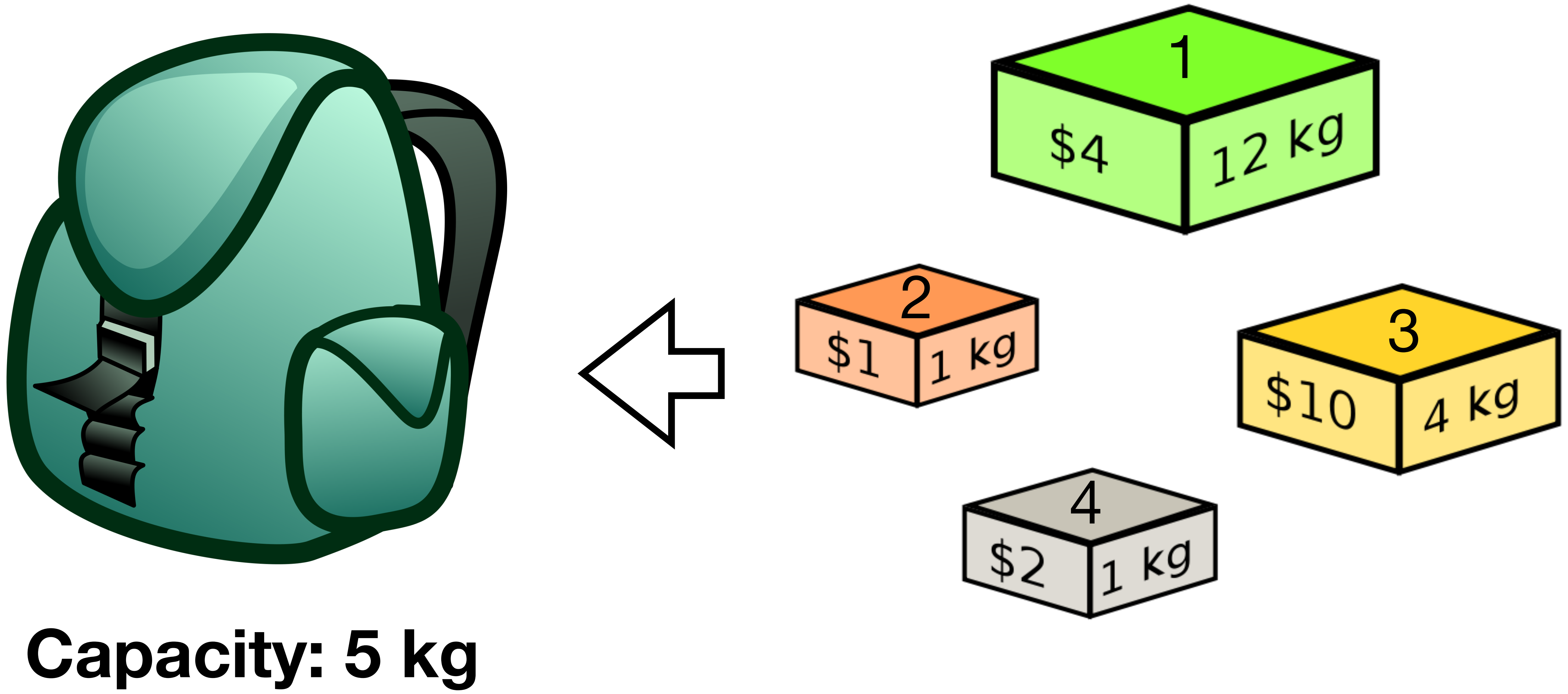
$$\max \; 4x_1 + x_2 + 2x_3 + 10x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \le 5$$

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

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.



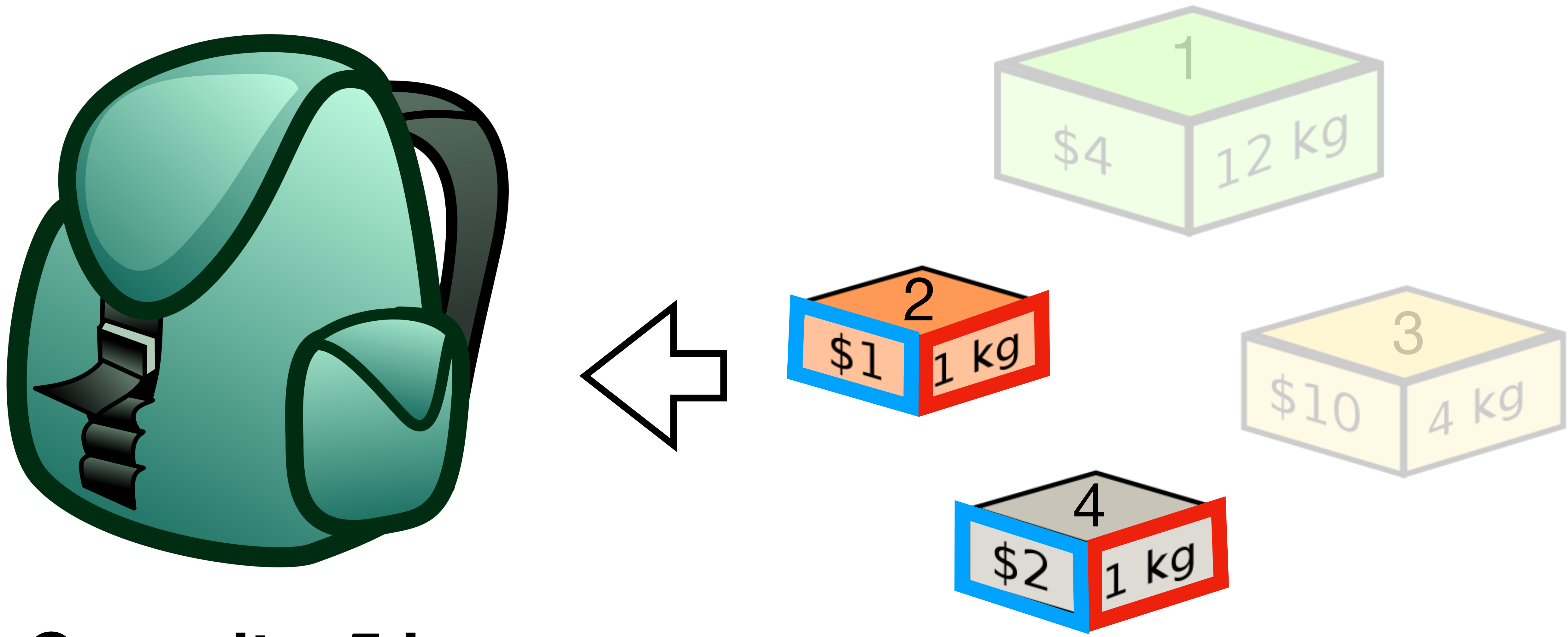

$$\max \ 4x_1 + x_2 + 2x_3 + 10x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

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

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.




**Capacity: 5 kg**

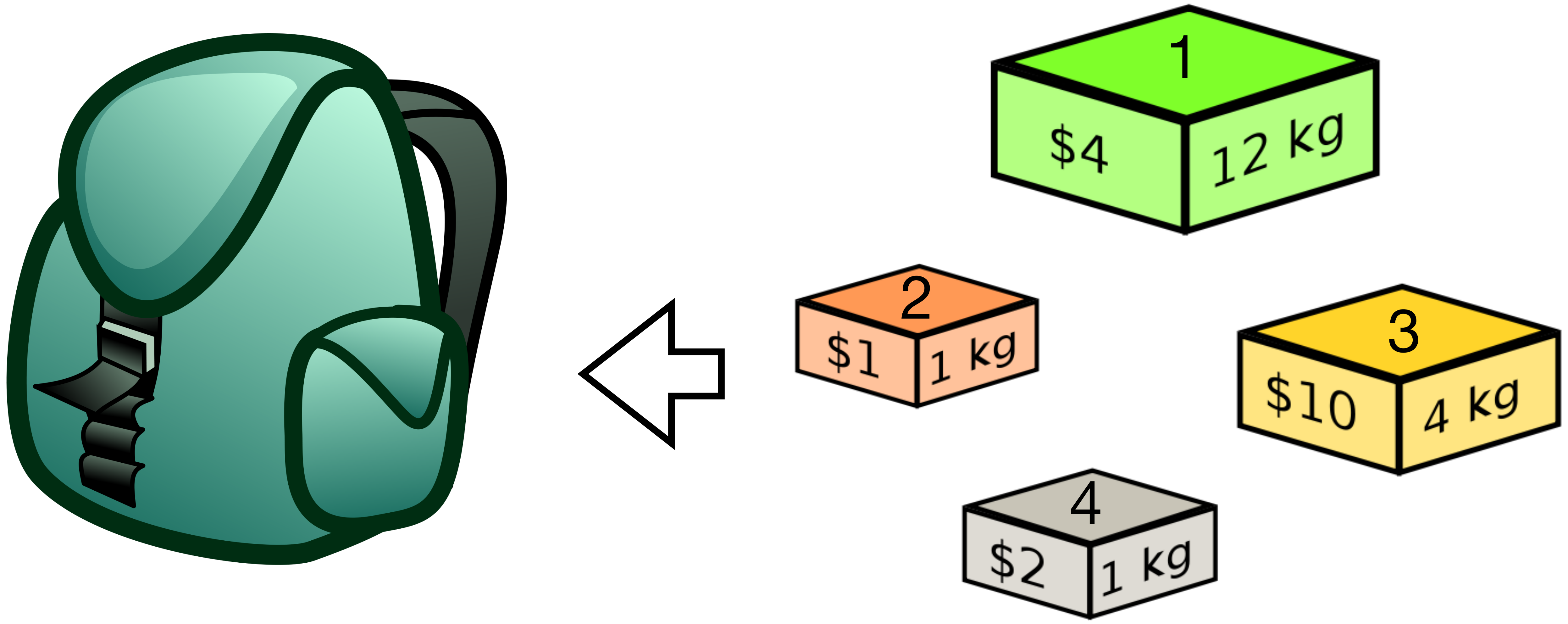
$$\max \ 4x_1 + x_2 + 10x_3 + 2x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

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

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.




**Capacity: 5 kg**

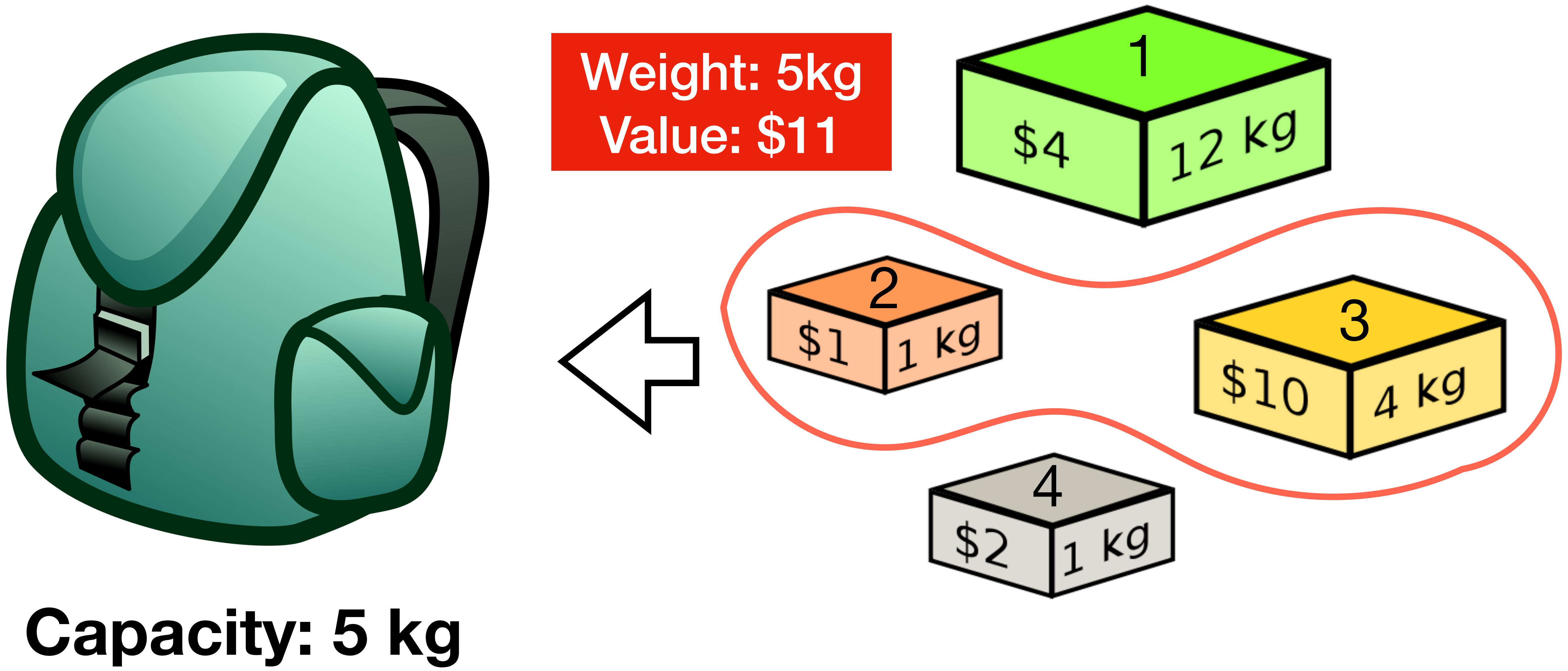
$$\max \; 4x_1 + x_2 + 10x_3 + 2x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

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

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.



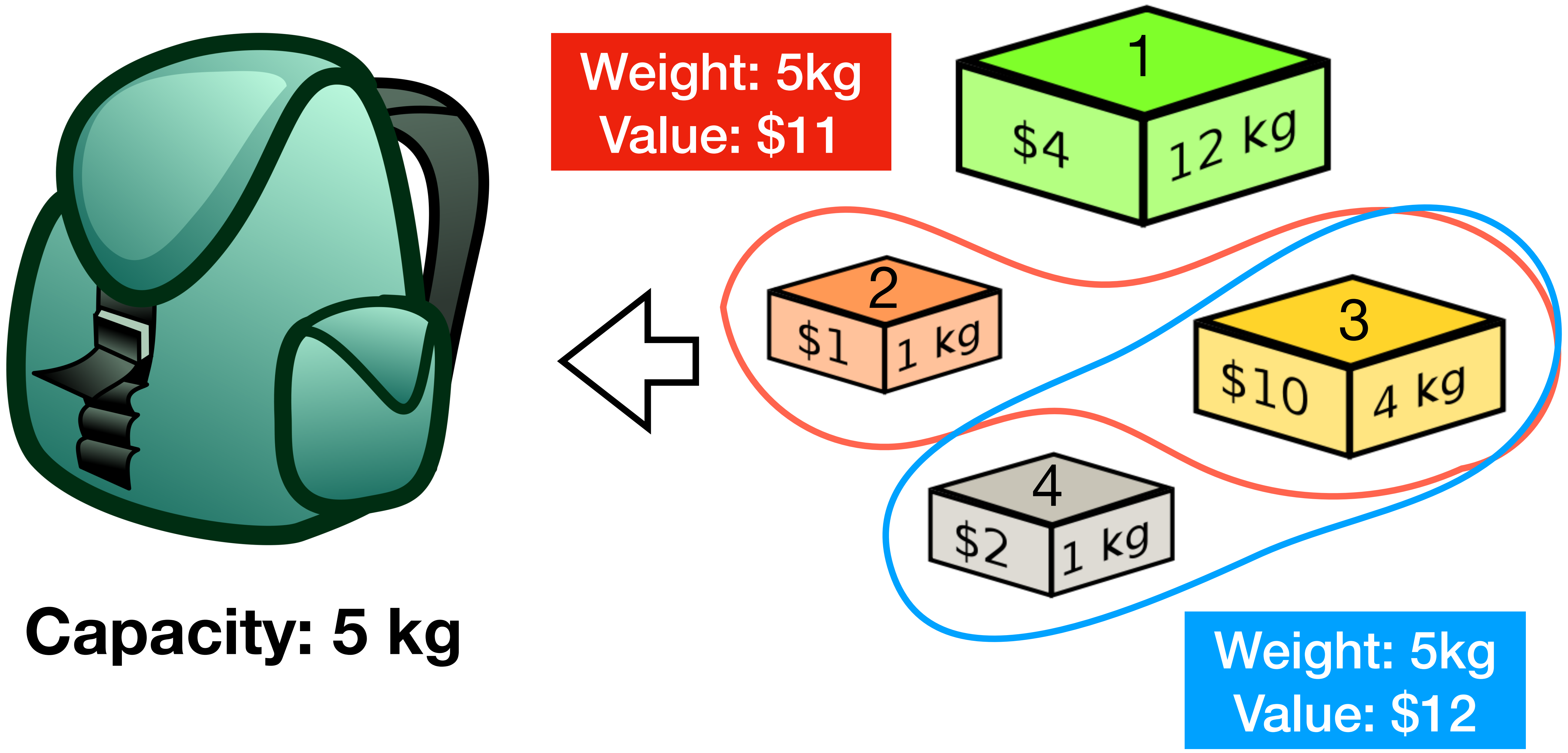

$$\max \ 4x_1 + x_2 + 10x_3 + 2x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

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

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.



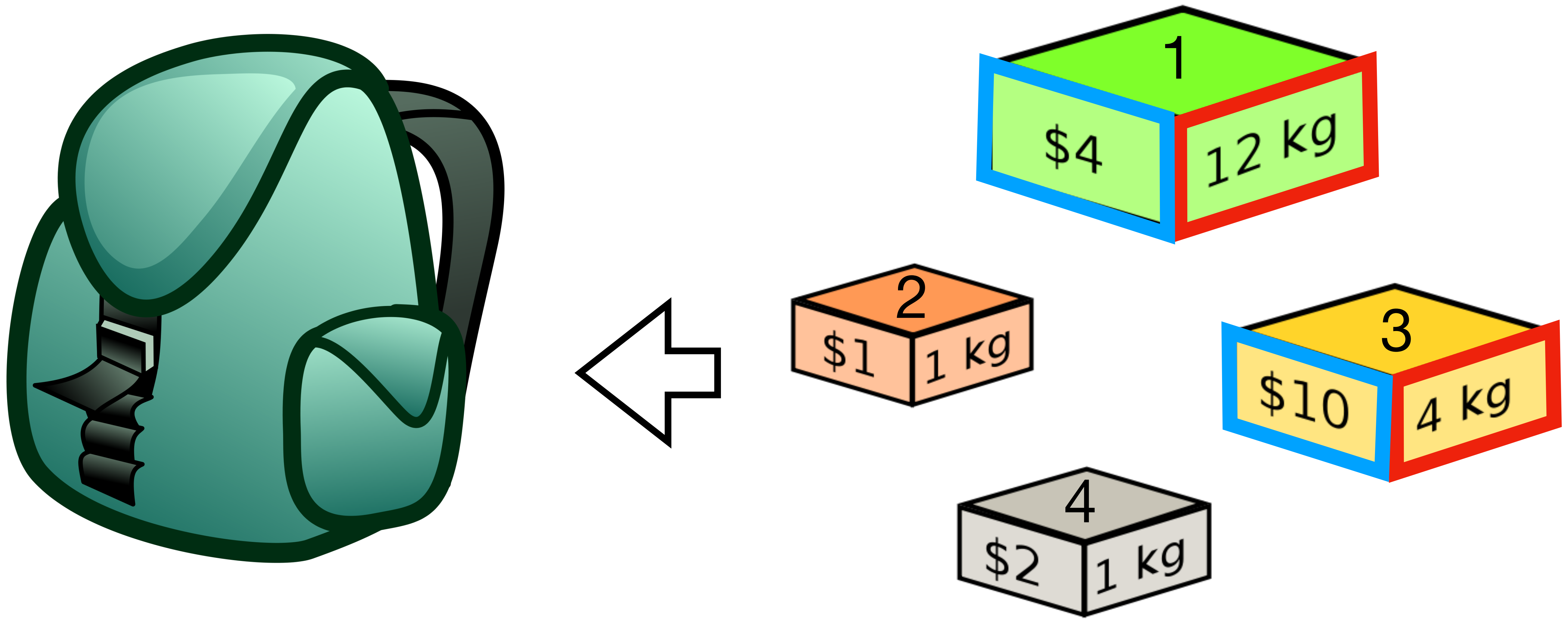

$$\max \ 4x_1 + x_2 + 10x_3 + 2x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

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

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.




**Capacity: 5 kg**

$$\max \; 4x_1 + x_2 + 10x_3 + 2x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

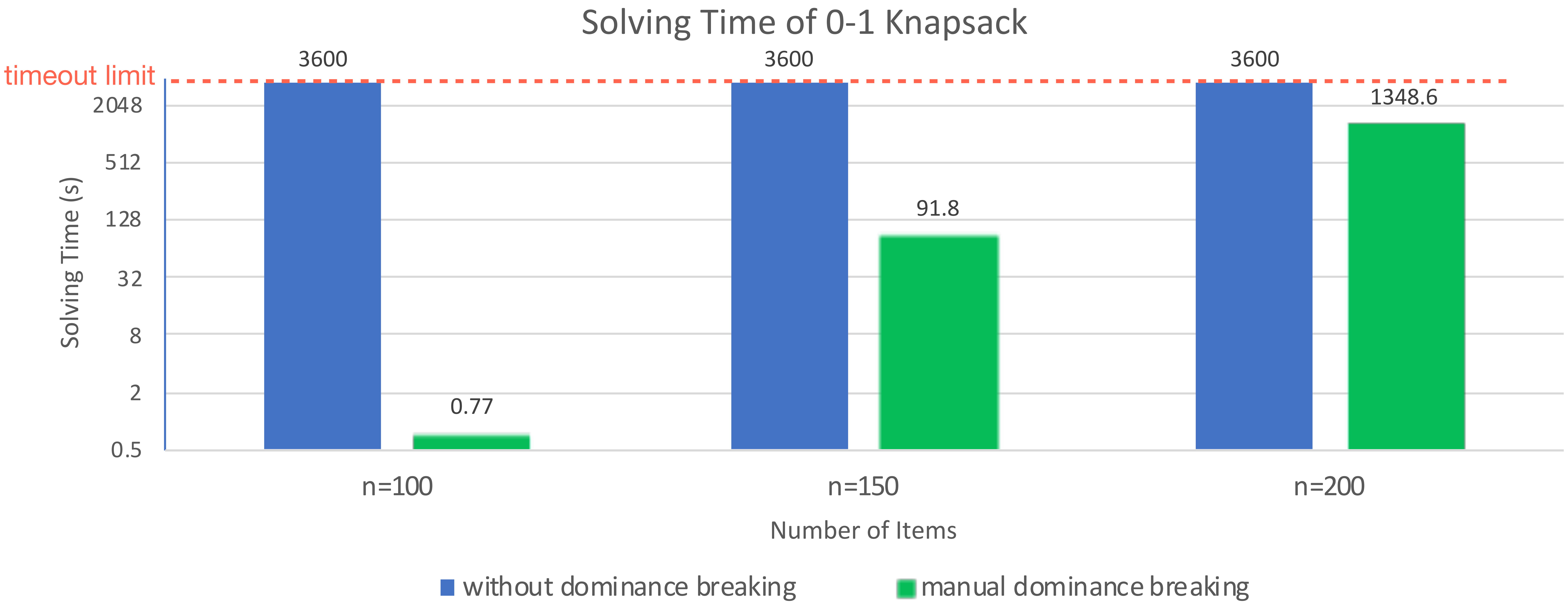
$$x_i \in \{0,1\} \text{ for } i = 1,\ldots,4$$

$$x_2 \leq x_4,\; x_1 \leq x_3$$

Dominance breaking constraints

# Dominance Breaking

- Dominance Breaking is a useful technique for solving COPs.




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

# Dominance Breaking

Problem Model

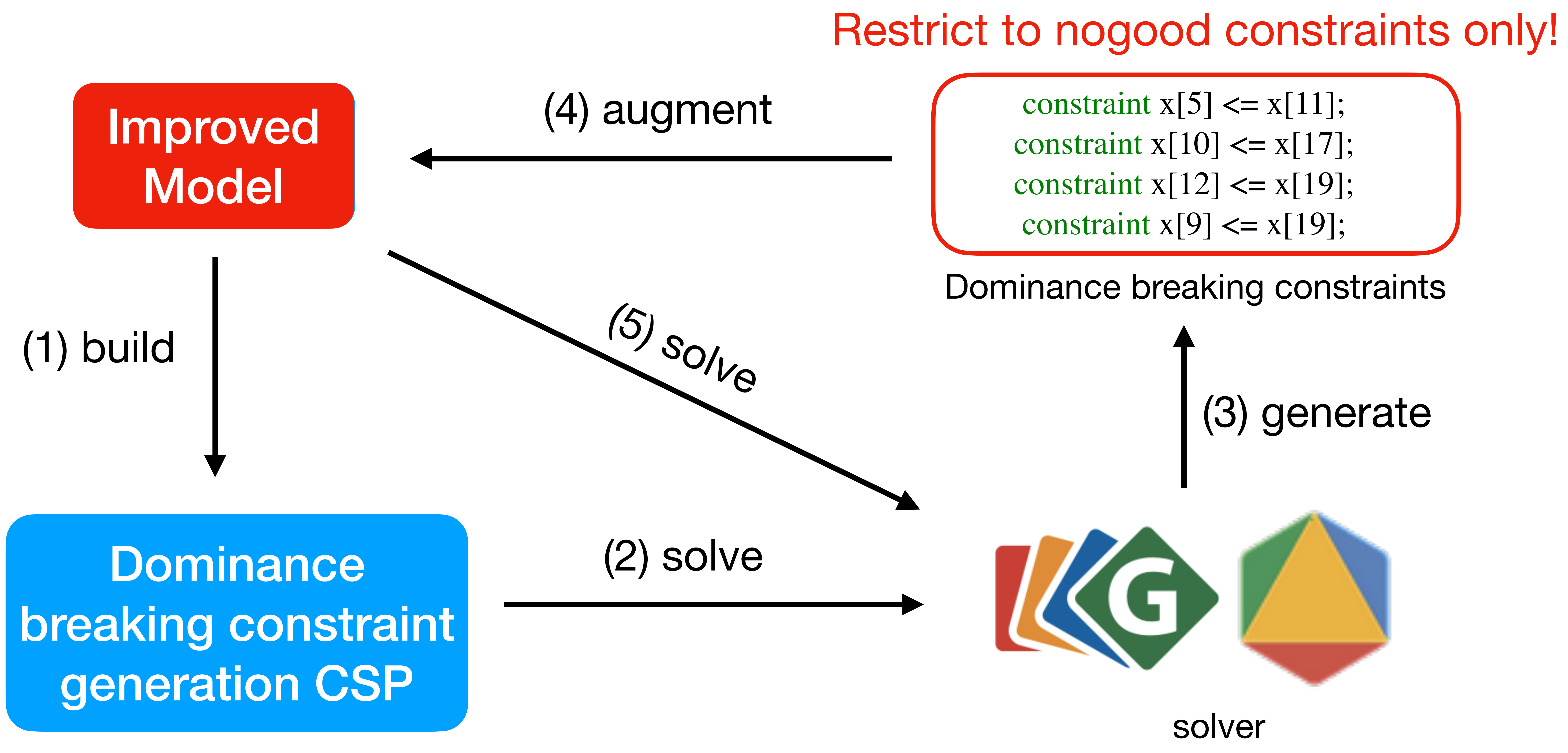
Image source: https://www.freepik.com/vectors/people

# Dominance Breaking

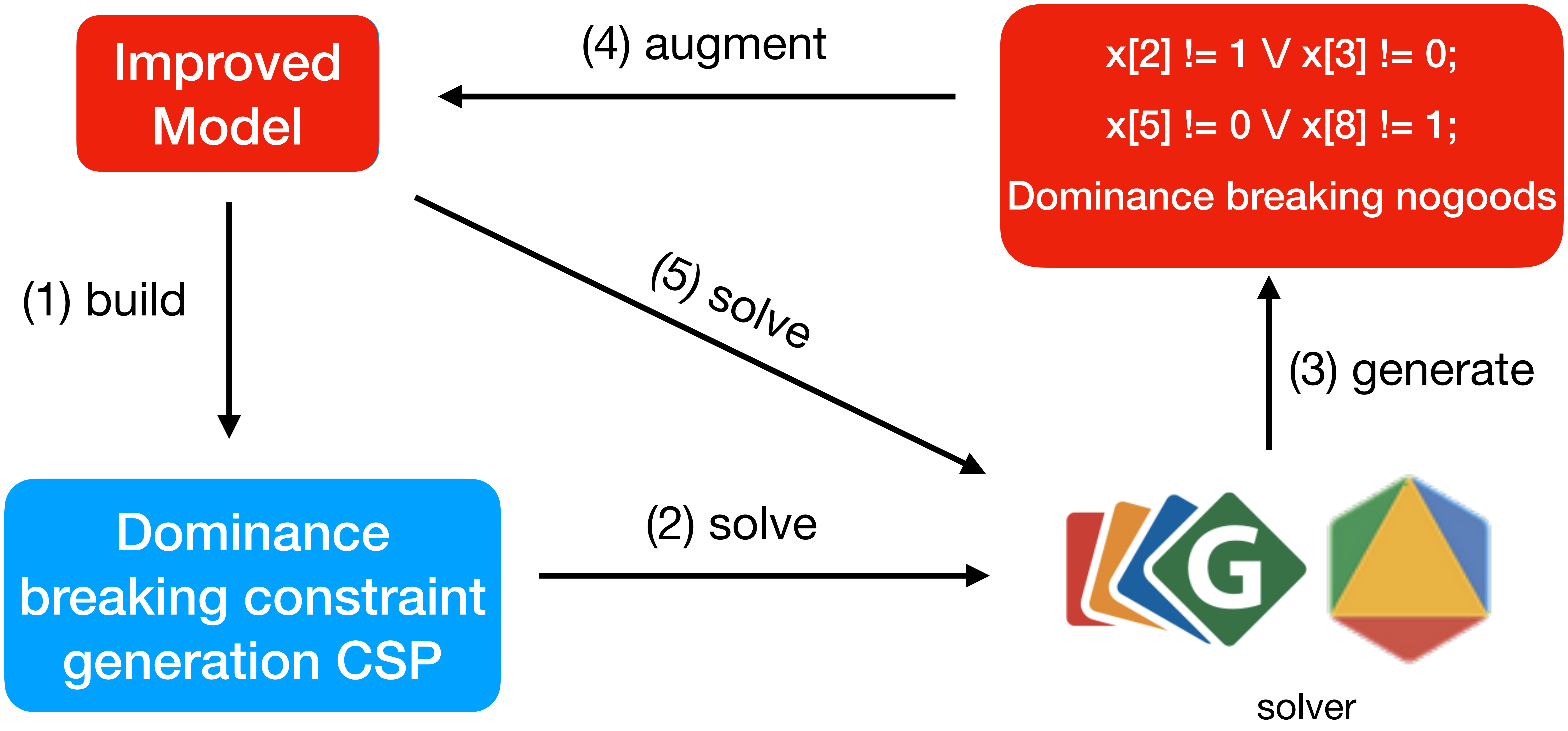
Problem Model → Model with dominance breaking

Image source: https://www.freepik.com/vectors/people

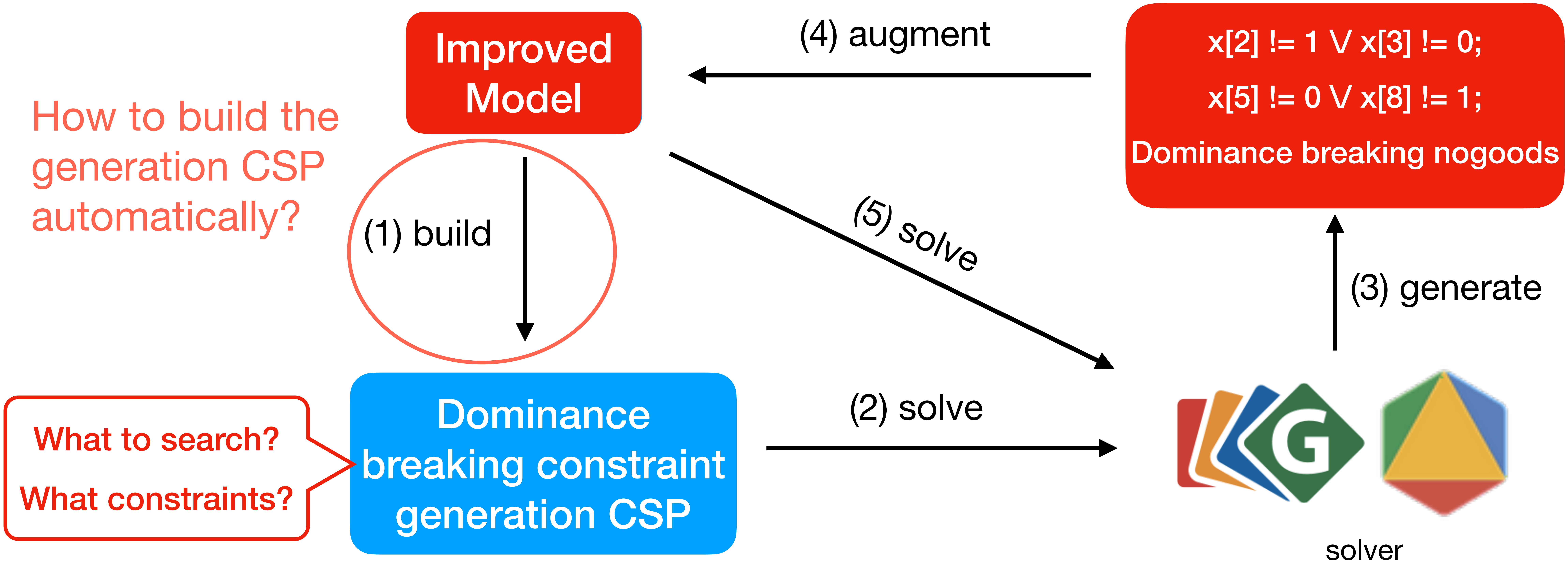
# Automatic Dominance Breaking



# Automatic Dominance Breaking



# Automatic Dominance Breaking

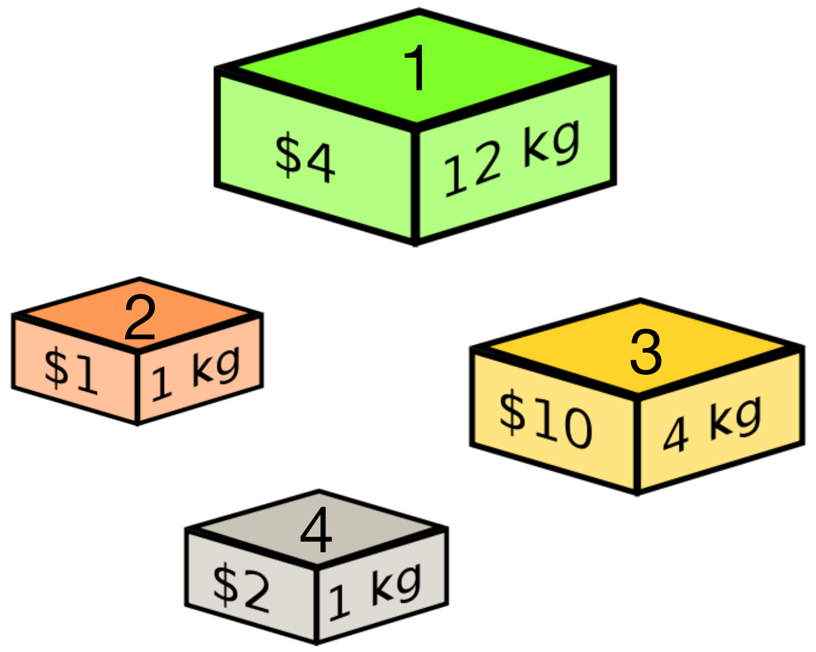


# Automatic Dominance Breaking

- What to search?
  - Pairs of partial assignments
- What constraints?

$$\max \; 4x_1 + x_2 + 10x_3 + 2x_4$$
$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$
$$x_i \in \{0,1\} \text{ for } i = 1,\ldots,4$$

Capacity: 5 kg




$\theta = \{x_2 = 0,\ x_4 = 1\}$

$\theta' = \{x_2 = 1,\ x_4 = 0\}$

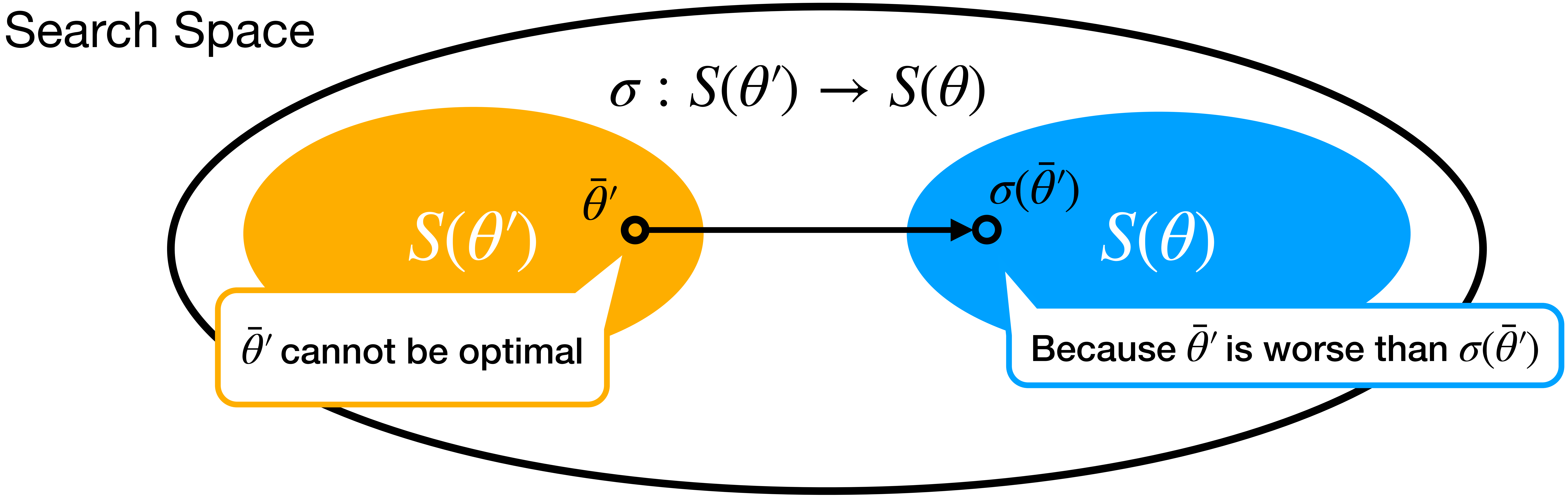
$S(\theta)$

| x2 | x4 | x1 | x3 |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |

$S(\theta')$

| x2 | x4 | x1 | x3 |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 |

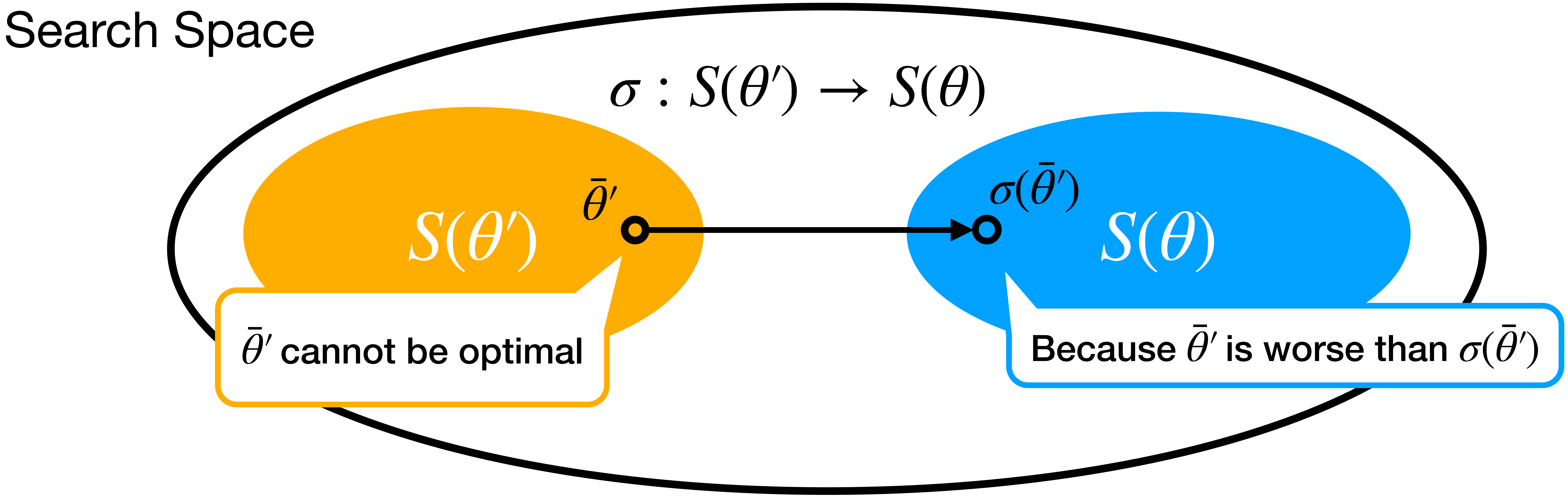
# Automatic Dominance Breaking




$\bar{\theta}'$ is worse than $\sigma(\bar{\theta}')$ if:

- $\bar{\theta}'$ solution $\Rightarrow \sigma(\bar{\theta}')$ solution
- $f(\sigma(\bar{\theta}'))$ is better than $f(\bar{\theta}')$

# Automatic Dominance Breaking




$\bar{\theta}'$ is worse than $\sigma(\bar{\theta}')$ if:

- Implied satisfaction
- Betterment

⇒ $\sigma(\bar{\theta}')$ dominates $\bar{\theta}'$ ( $\sigma(\bar{\theta}') \prec \bar{\theta}'$ )

# Automatic Dominance Breaking

- What objects?
  - Pairs of partial assignments
- What constraints?

$$\theta' = \{x_2 = 1,\ x_4 = 0\}$$

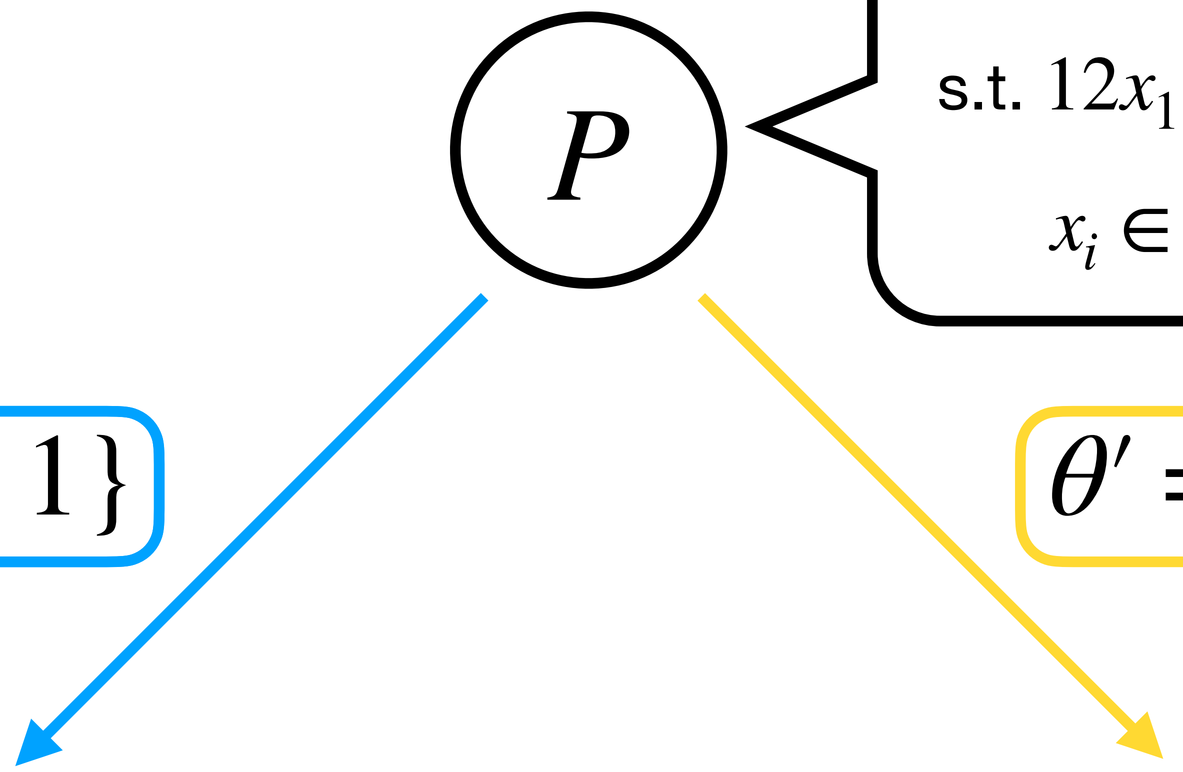
equivalent

$$\theta' \equiv x_2 = 1 \wedge x_4 = 0$$

negation

$$\neg\theta' \equiv x_2 \neq 1 \vee x_4 \neq 0$$

**Dominance Breaking Nogood**




$$\max\ 4x_1 + x_2 + 10x_3 + 2x_4$$
$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$
$$x_i \in \{0,1\} \text{ for } i = 1,\ldots,4$$

$$\theta = \{x_2 = 0,\ x_4 = 1\}$$

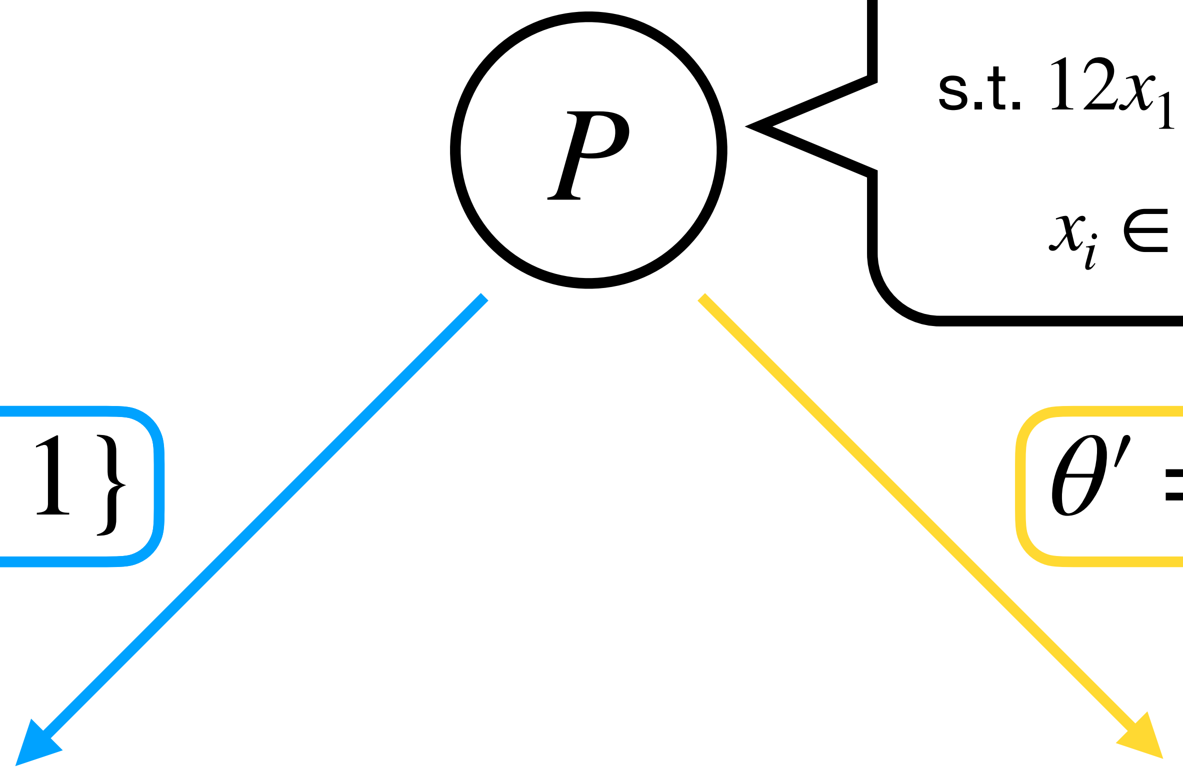
$$\theta' = \{x_2 = 1,\ x_4 = 0\}$$

| x2 | x4 | x1 | x3 |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |

$S(\theta)$

# Automatic Dominance Breaking

- What objects?
  - Pairs of partial assignments
- What constraints?




$$\max\ 4x_1 + x_2 + 10x_3 + 2x_4$$
$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \le 5$$
$$x_i \in \{0,1\} \text{ for } i = 1,\ldots,4$$

$\theta = \{x_2 = 0,\ x_4 = 1\}$

$\theta' = \{x_2 = 1,\ x_4 = 0\}$

**Theorem:**
if $\forall \bar{\theta}' \in S(\theta')$ s.t. $\sigma(\bar{\theta}') \prec \bar{\theta}'$,
then we can add $\neg\theta'$ to P

| x2 | x4 | x1 | x3 |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |

$S(\theta)$

# Automatic Dominance Breaking

- What objects?
  - Pairs of partial assignments
- What constraints?

Want to show : $\forall \bar{\theta}' \in S(\theta')$,

- Implied satisfaction:

  $\bar{\theta}'$ solution $\Rightarrow \sigma(\bar{\theta}')$ solution

- Betterment:

  $f(\sigma(\bar{\theta}'))$ is better than $f(\bar{\theta}')$

- Implied satisfaction concerns each constraint in the problem model
- Betterment concerns the objective in the problem model
- Sufficient conditions for efficiently checkable (EC) objectives and constraints

Constraints for $(\theta, \theta')$ !

# EC Objectives and Constraints

| Objectives | Constraints |
| --- | --- |
| • Separable objectives<br>• Submodular set objectives<br>• Maximum objectives | • Domain constraints<br>• Boolean disjunction constraints<br>• Linear Inequality constraints<br>• Alldifferent constraints<br>• Circuit constraints<br>• Counting family constraints |

# Modelling

```
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]);
```

Problem 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; % length of nogoods

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 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 Model

# Modelling

```
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; % length of nogoods

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'
```

## Three Orders of magnitude improvement!

```
% 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 Model

```
int: n;        % number of items
int: W;        % knapsack capacity
array [1..n] of int: w;   % weight of each item
array
array
% constraint
constraint sum (i in 1..n) (w[i]*x[i]) <= W;

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

Problem Model

# Outline

- Automatic Dominance Relations
- Non-Efficiently Checkable Constraints
- Common Assignment Elimination
- Experimental Results

# Non-Efficiently Checkable Constraints

```
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;

constraint x[1] = 1 Λ x[3] = 0 -> false;

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

Problem Model

**All constraints are efficiently checkable**

A non-EC constraint

# Non-Efficiently Checkable Constraints

```
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;

constraint x[1] = 1 ∧ x[3] = 0 -> false;

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

→ $var(c) = \{x_1, x_3\}$

Problem Model

# Non-Efficiently Checkable Constraints

```
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;

constraint x[1] = 1 Λ x[3] = 0 -> false;

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

Problem Model

$C_{nec}$ : the set of non-EC constraints

Constraints in the generation CSP

- Sufficient conditions for EC constraints and objectives
- $\forall c \in C_{nec}$ s.t. $var(\theta) \cap var(c) = \emptyset$

# Non-Efficiently Checkable Constraints

```
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;

constraint x[1] = 1 ∧ x[3] = 0 -> false;

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

Problem Model

$C_{nec}$ : the set of non-EC constraints

Constraints in the generation CSP

- Sufficient conditions for EC constraints and objectives

**Theorem:**
**if $var(\theta) \cap var(c) = \emptyset$, then $(\theta, \theta')$ satisfies implied satisfaction for $c$**

# Non-Efficiently Checkable Constraints

```
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;

constraint x[1] = 1 ∧ x[3] = 0 -> false;

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

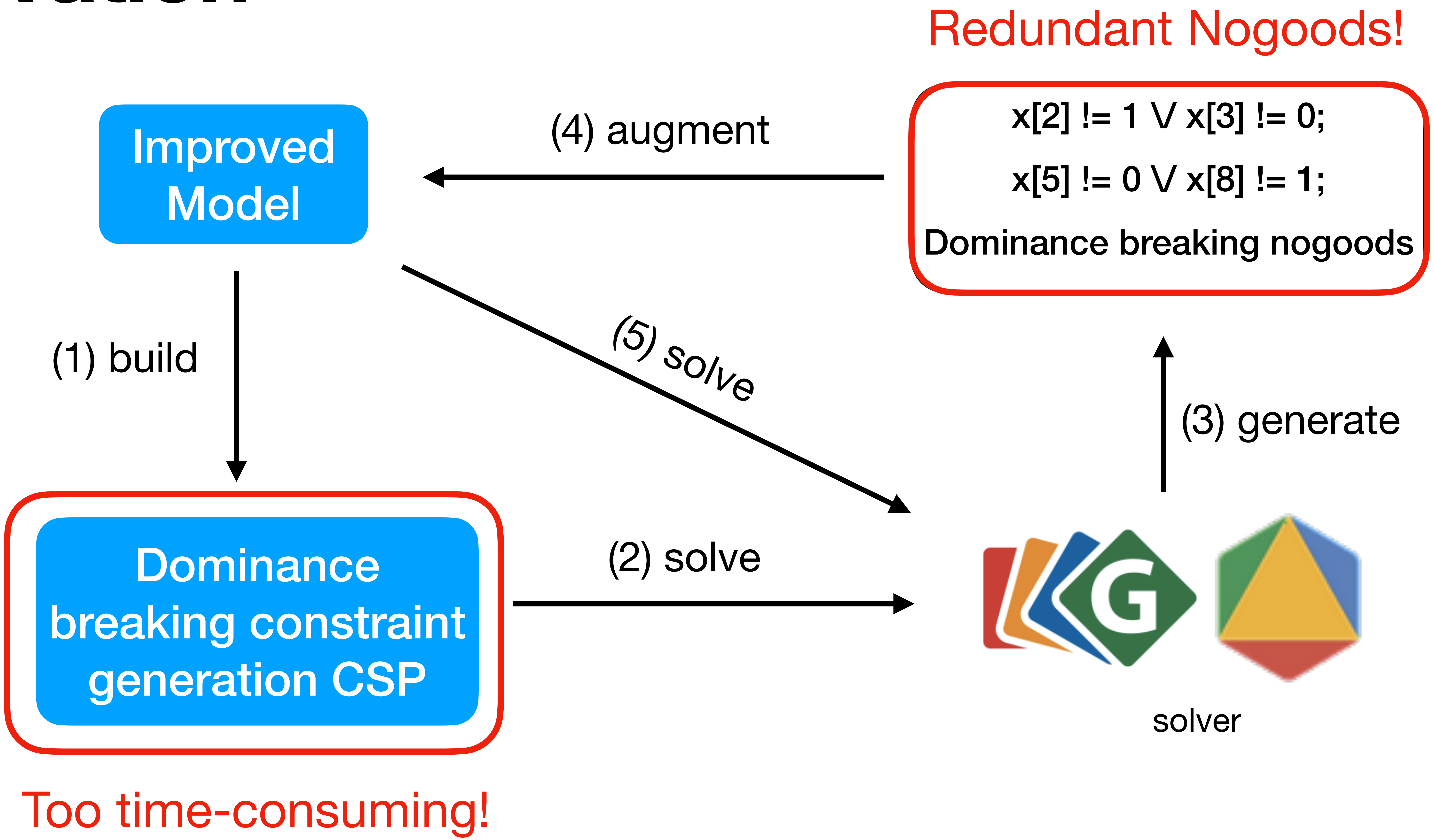
Problem Model

- Useful only if the scope of the non-EC constraint is relatively small
  - Side constraints that involves several variables
- Enable automatic dominance breaking on a larger class of problems

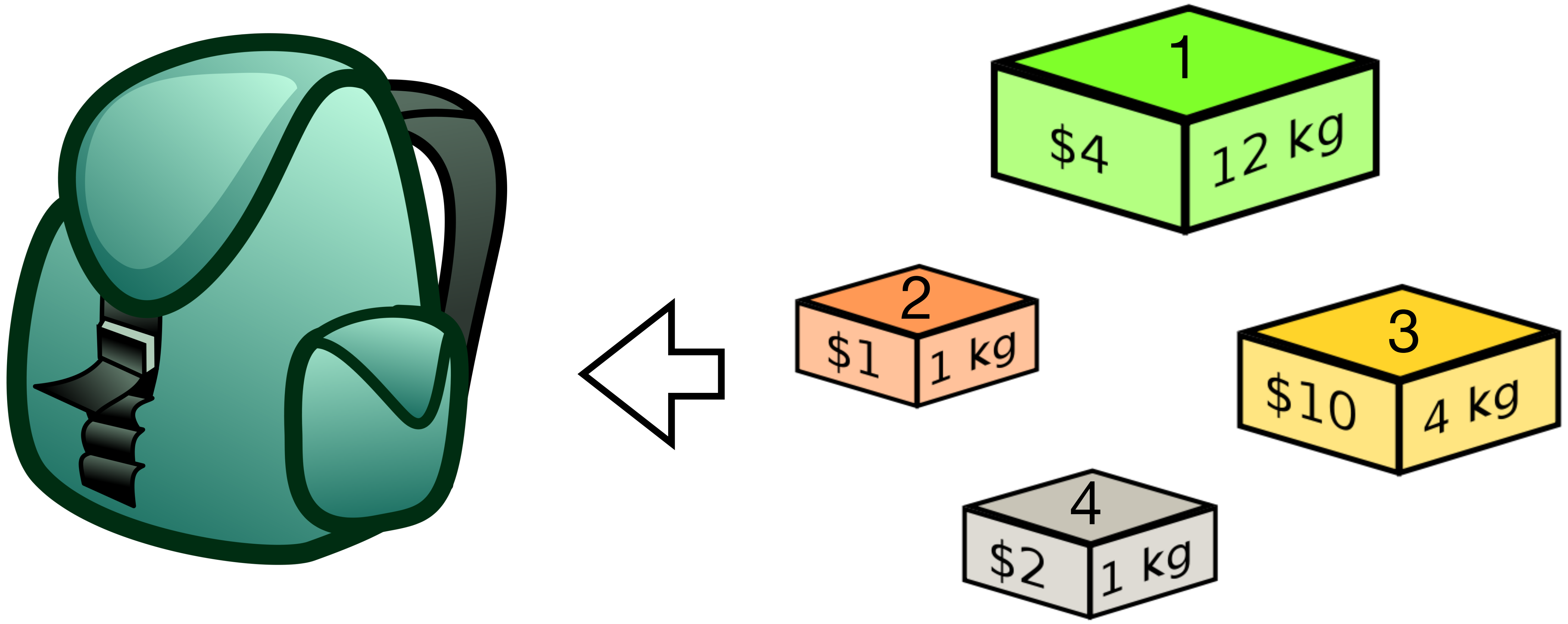
# Outline

- Automatic Dominance Relations
- Non-efficiently Checkable Constraints
- Common Assignment Elimination
- Experimental Results

# Motivation


Redundant Nogoods!
x[2] != 1 V x[3] != 0;
x[5] != 0 V x[8] != 1;
Dominance breaking nogoods
Improved Model
(4) augment
(1) build
(5) solve
(3) generate
Dominance breaking constraint generation CSP
(2) solve
solver
Too time-consuming!

# Common Assignment Elimination




**Capacity: 5 kg**

$$\max \; 4x_1 + x_2 + 10x_3 + 2x_4$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 \leq 5$$

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

~~$c_1 \equiv (\, x_2 \neq 1 \vee x_4 \neq 0 \vee x_3 \neq 1 \,)$~~

$$c_2 \equiv (\, x_2 \neq 1 \vee x_4 \neq 0 \,)$$

$$c_2 \Rightarrow c_1$$

How to avoid generating $c_1$?
Adding more constraints!

# Common Assignment Elimination

**Generation CSP**

- $var(\theta) = var(\theta')$
- $\forall \bar{\theta}' \in S(\theta')$ s.t. $\sigma(\bar{\theta}') \prec \bar{\theta}'$
  - $\bar{\theta}'$ solution $\Rightarrow \sigma(\bar{\theta}')$ solution
  - $f(\sigma(\bar{\theta}'))$ is better than $f(\bar{\theta}')$

Generate

Common assignment

$\theta_1 = \{x_2 = 0, x_4 = 1, x_3 = 1\}$

$\theta_1' = \{x_2 = 1, x_4 = 0, x_3 = 1\}$

Eliminate $(x_3 = 1)$

$\theta_2 = \{x_2 = 0, x_4 = 1\}$

$\theta_2' = \{x_2 = 1, x_4 = 0\}$

$\theta_1 \prec \theta_1'$ Derive

$\theta_2 \prec \theta_2'$ ? Derive

~~$c_1 \equiv \neg\theta_1' \equiv (x_2 \neq 1 \vee x_4 \neq 0 \vee x_3 \neq 1)$~~

Redundant

Implies

$c_2 \equiv \neg\theta_2' \equiv (x_2 \neq 1 \vee x_4 \neq 0)$

# Common Assignment Elimination

**Generation CSP**

- $var(\theta) = var(\theta')$
- $\forall \bar{\theta}' \in S(\theta')$ s.t. $\sigma(\bar{\theta}') \prec \bar{\theta}'$
  - $\bar{\theta}'$ solution $\Rightarrow \sigma(\bar{\theta}')$ solution
  - $f(\sigma(\bar{\theta}'))$ is better than $f(\bar{\theta}')$

$(x_3 = 1) \notin \theta \cap \theta'$

$$\theta_2 = \{x_2 = 0,\ x_4 = 1\}$$
$$\theta_2' = \{x_2 = 1,\ x_4 = 0\}$$

$\theta_2 \prec \theta_2'$ ? Derive

$$c_2 \equiv \neg\theta_2' \equiv (x_2 \neq 1 \vee x_4 \neq 0)$$

# Commonly Eliminable Assignments

| Common Assignments | Objectives / Constraints |
| --- | --- |
| x = 0 | • Submodular set objectives<br>• Boolean disjunction constraint |
| x = v for any value v | • Separable objectives<br>• Domain constraint<br>• Linear Inequality constraint<br>• Alldifferent constraint |

# Modelling

```
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;

% non-EC constraint
constraint x[3] != 1 \/ x[1] != 0;

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

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

Problem Model

```
% problem parameters
…
int: k; % length 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 for implied satisfaction and betterment
…
```

Generation CSP Model

# Modelling

```
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;

% non-EC constraint
constraint x[3] != 1 \/ x[1] != 0;

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

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

Problem Model

```
% problem parameters
…
int: k; % length 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 for implied satisfaction and betterment
…

% handling non-EC constraint
constraint formal (i in 1..k) (F[i] != 3 \/ F[I] != 1);
```

Generation CSP Model

# Modelling

```
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;

% non-EC constraint
constraint x[3] != 1 \/ x[1] != 0;

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

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

Problem Model

```
% problem parameters
...
int: k; % length 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 for implied satisfaction and betterment
...

% handling non-EC constraint
constraint formal (i in 1..k) (F[i] != 3 \/ F[I] != 1);

% common assignment elimination
constraint formal (i in 1..k, v in 0..1) (
        v1[k] != v \/ v2[k] != v
);
```

Generation CSP Model

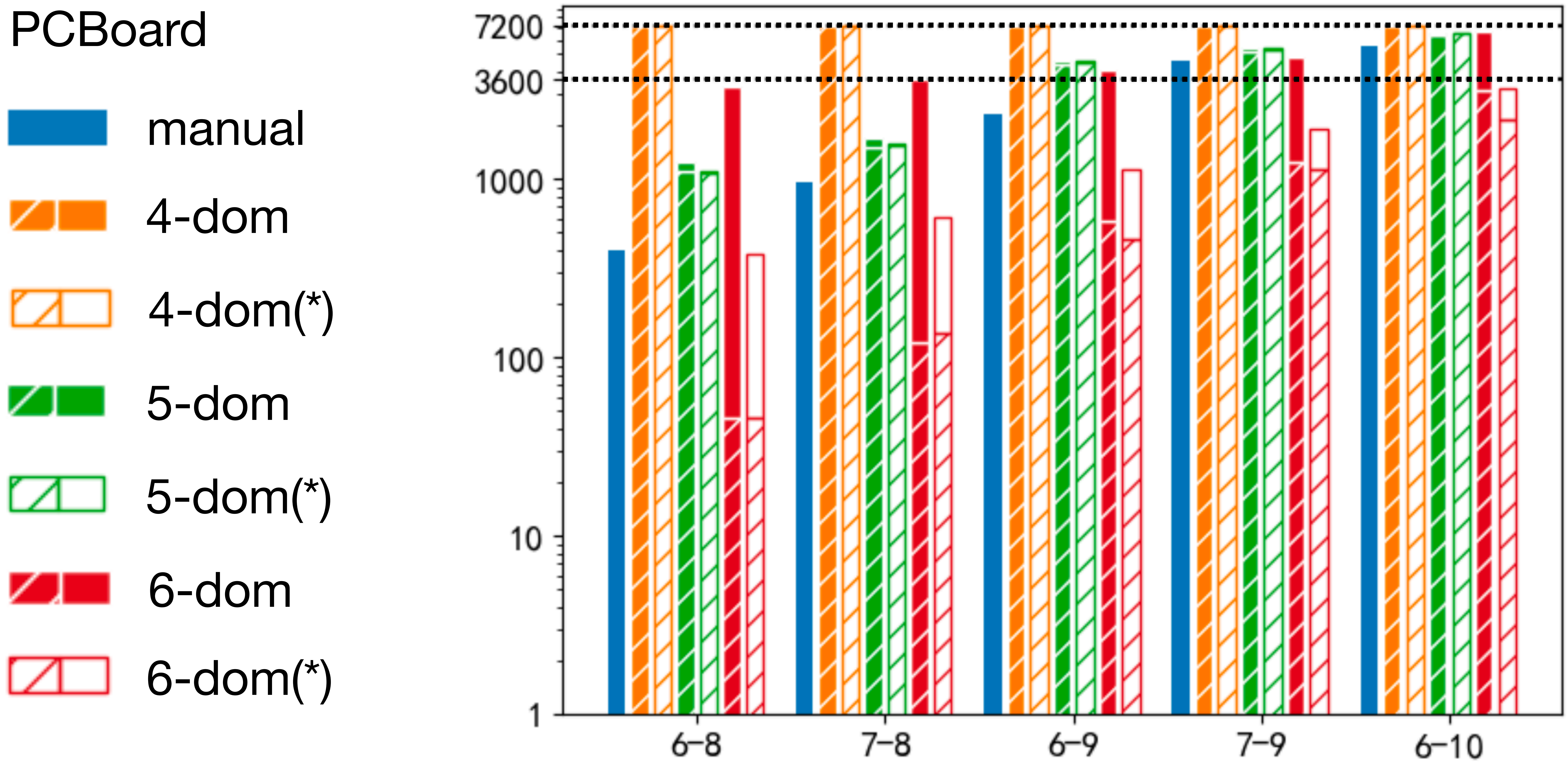
# Outline

- Automatic Dominance Relations
- Non-efficiently Checkable Constraints
- Common Assignment Elimination
- Experimental Results

# Experimental Setup

- **MiniZinc** for modelling, **Chuffed** for solving;
- **Two hours** total timeout; **One hour** timeout for no-good generation;
- 6 benchmarks, 20 random instances for each configuration
  - Existing: **Knapsack, Disjunctively Constrained Knapsack, Concert Hall Scheduling, Maximum Cut**
  - **KnapsackSide: Knapsack** with additional table constraints
  - **PCBoard**: larger overhead of nogood generation

# Experimental Evaluation



# Concluding Remarks

- Dominance breaking is powerful, but applying it is difficult.
- Automatic dominance breaking for a class of problems.
  - Instead of free-form constraints, generate nogoods
  - Some are not discovered by human (yet)
- Handle Non-EC constraints and Common Assignment Elimination

# Thanks for listening!