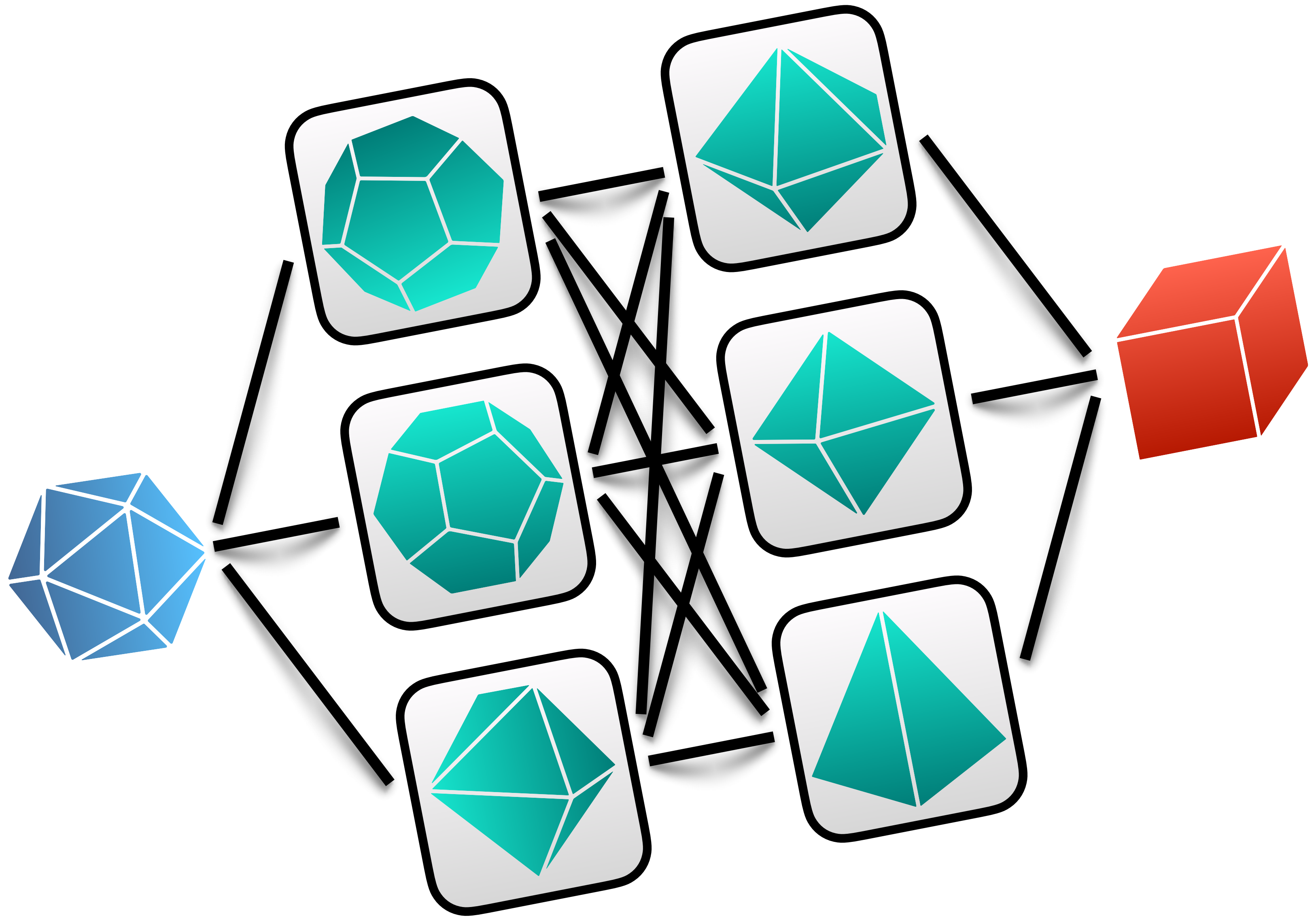
Did I forget to hit record? Please remind me!



Course Overview

MIE1666: Machine Learning for Mathematical Optimization

Elias B. Khalil — 13/09/21

UNIVERSITY OF TORONTO

Humans learn to **design** **algorithms**.

Humans learn to **design** **algorithms**.

Can **algorithms** "learn" to **design** **algorithms**?

Humans learn to **design** **algorithms**.

Machine Learning

Can **algorithms** “learn” to **design** **algorithms**?

Humans learn to **design** **algorithms**.

Machine Learning

Can **algorithms** "learn" to **design** **algorithms**?

Discrete Optimization

Data Center Resource Management

Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits

Data Center Resource Management

Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits

Data Center Resource Management

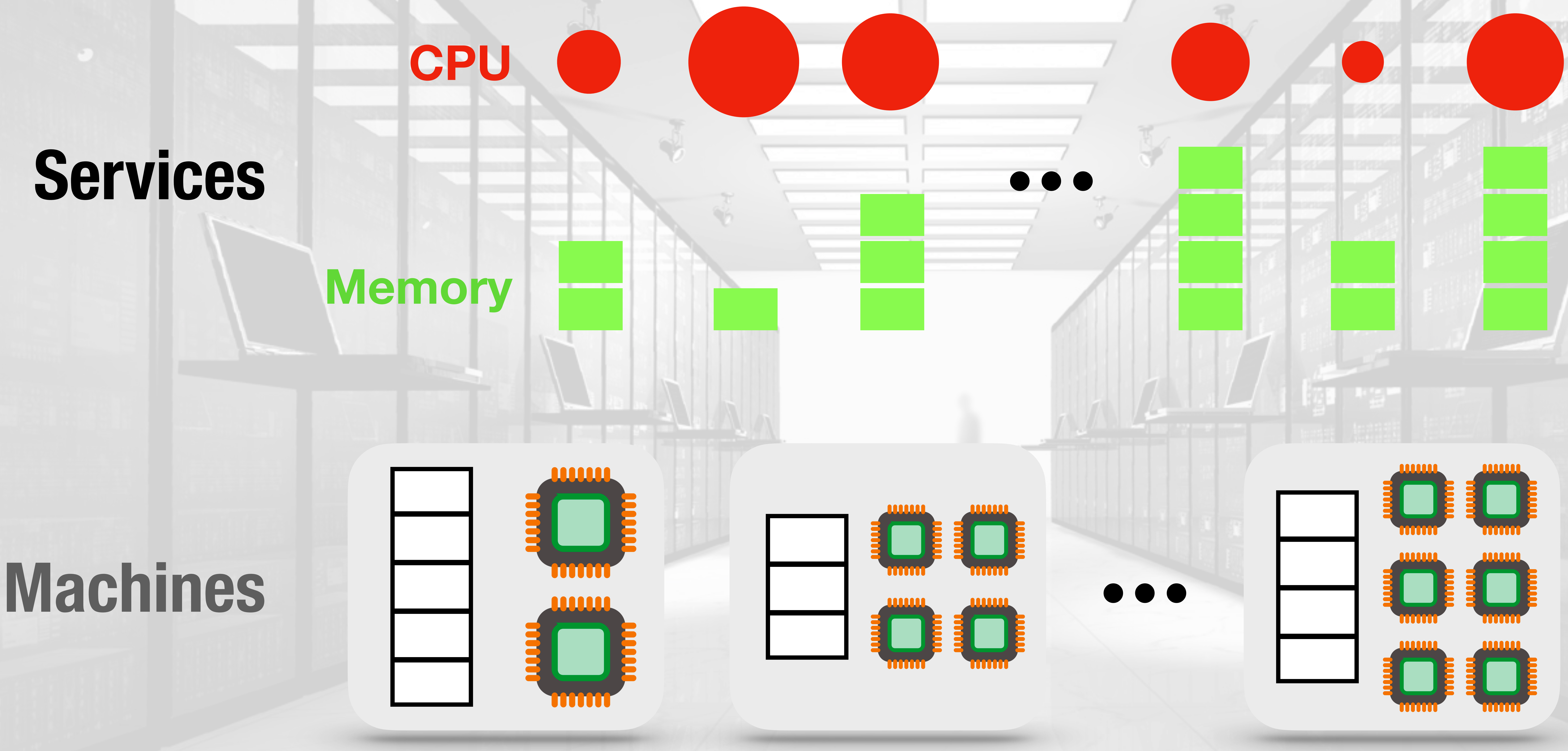


 Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits

Data Center Resource Management

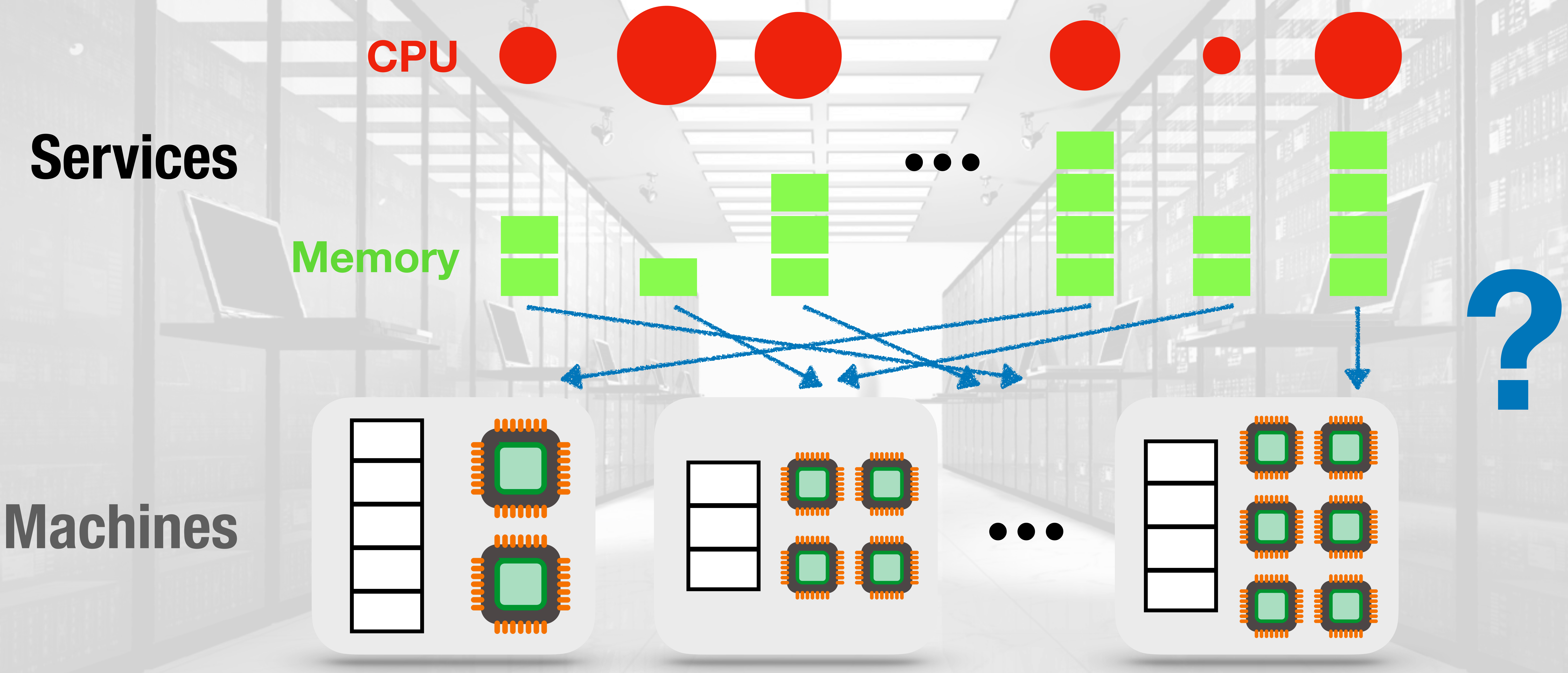
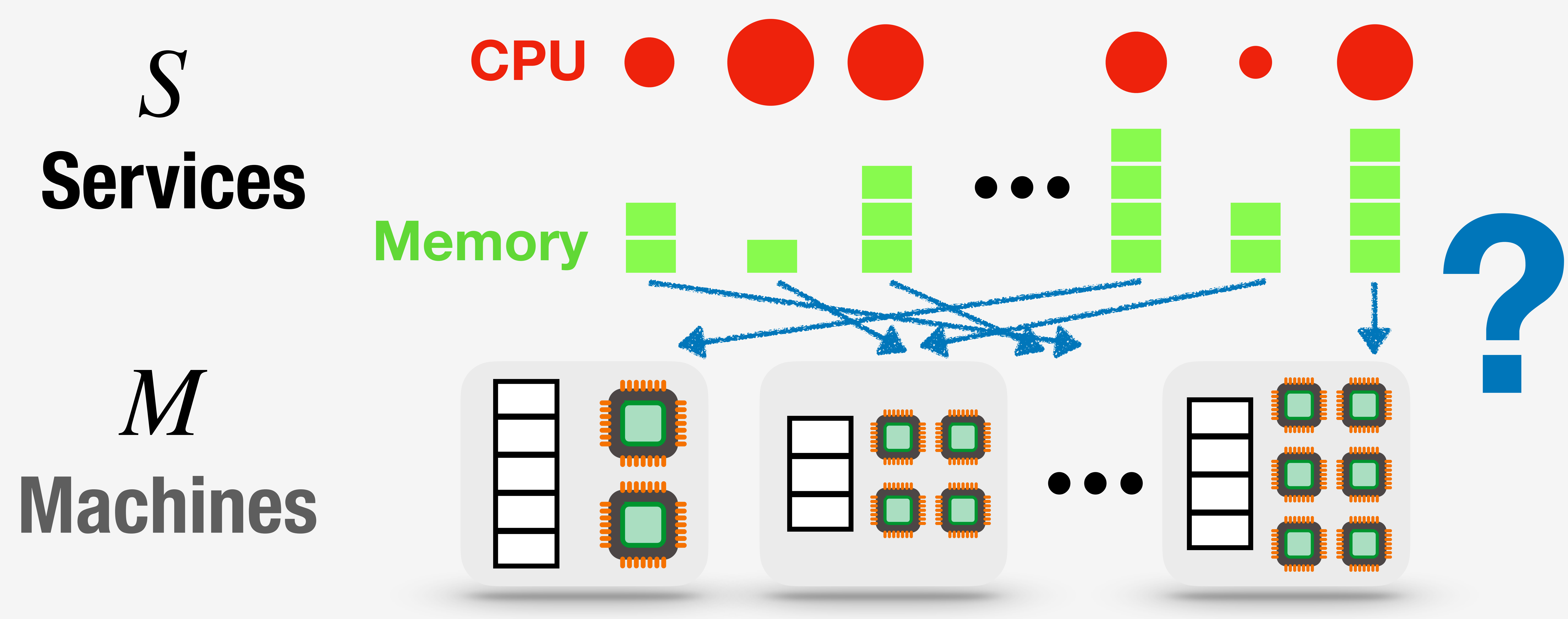


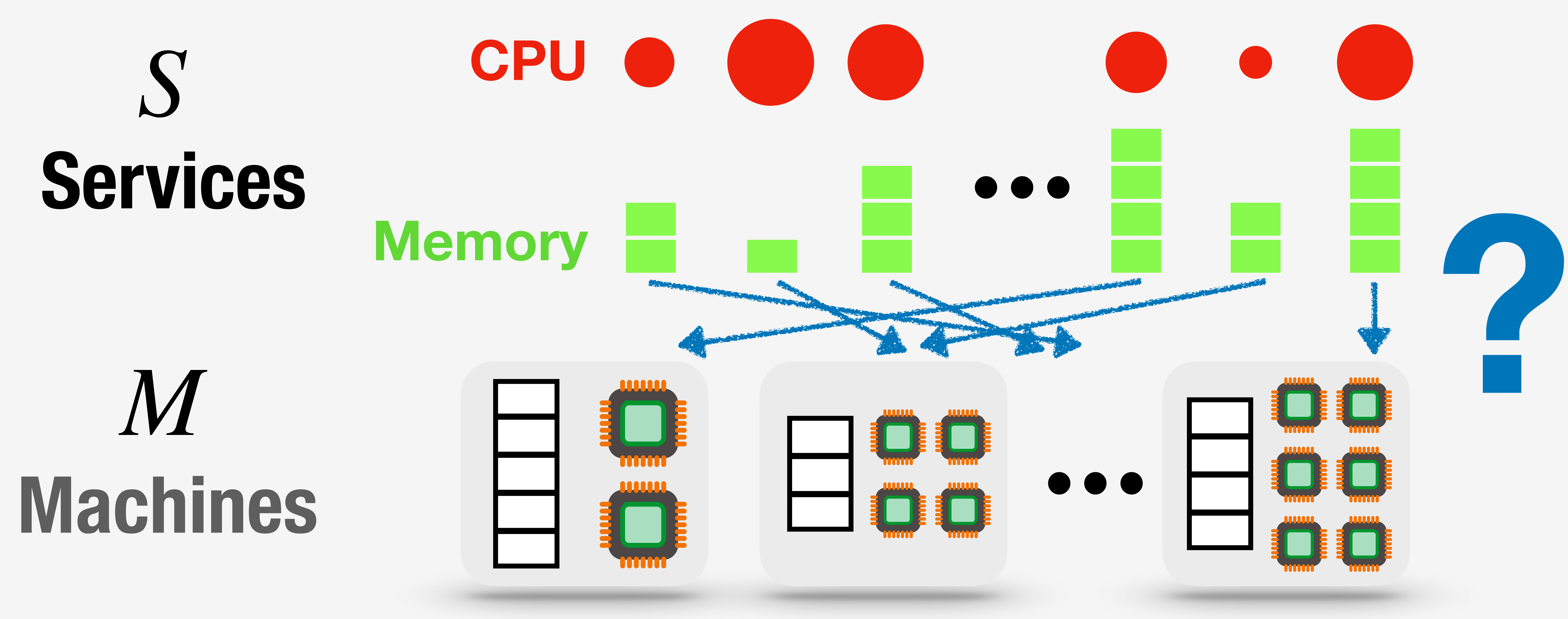
Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits



S
Services
CPU
Memory
M
Machines
?

$y_m = 1$ if machine m is used

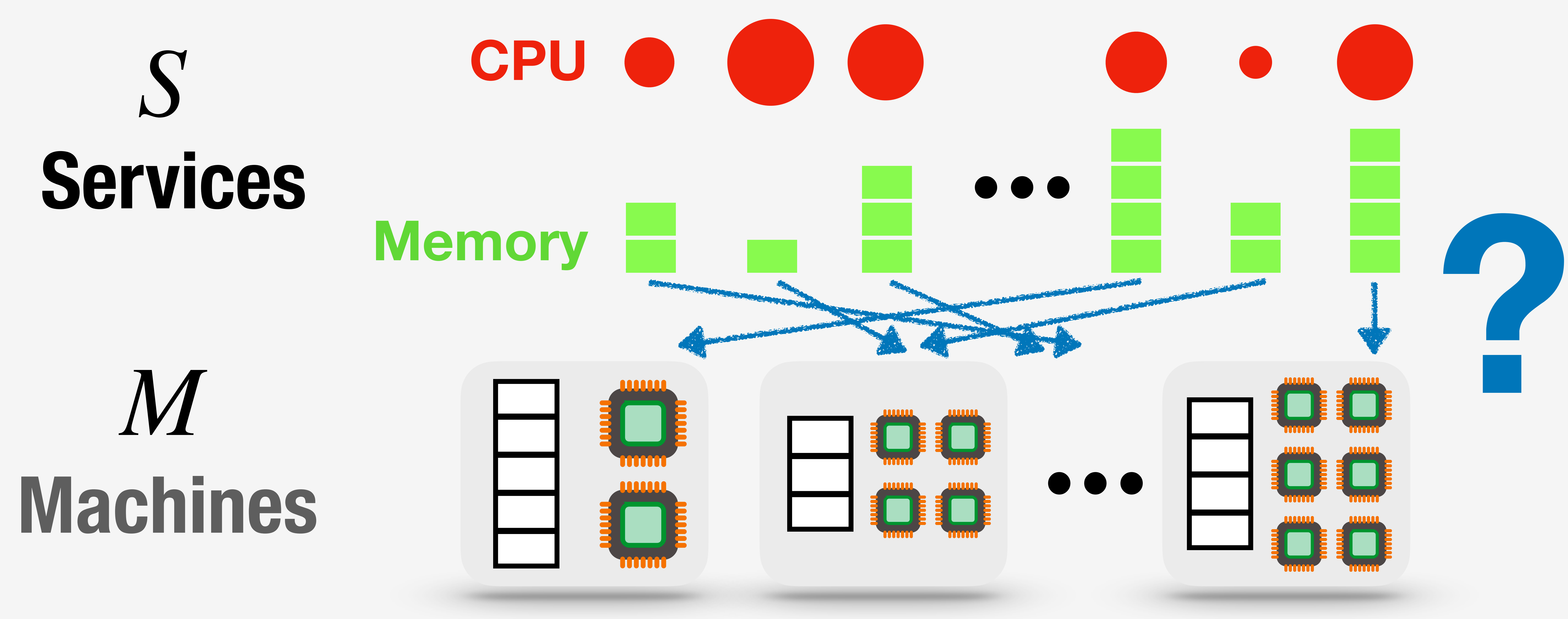
$x_{s,m} = 1$ if service s runs on m



$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}, y \in \{0,1\}^M$

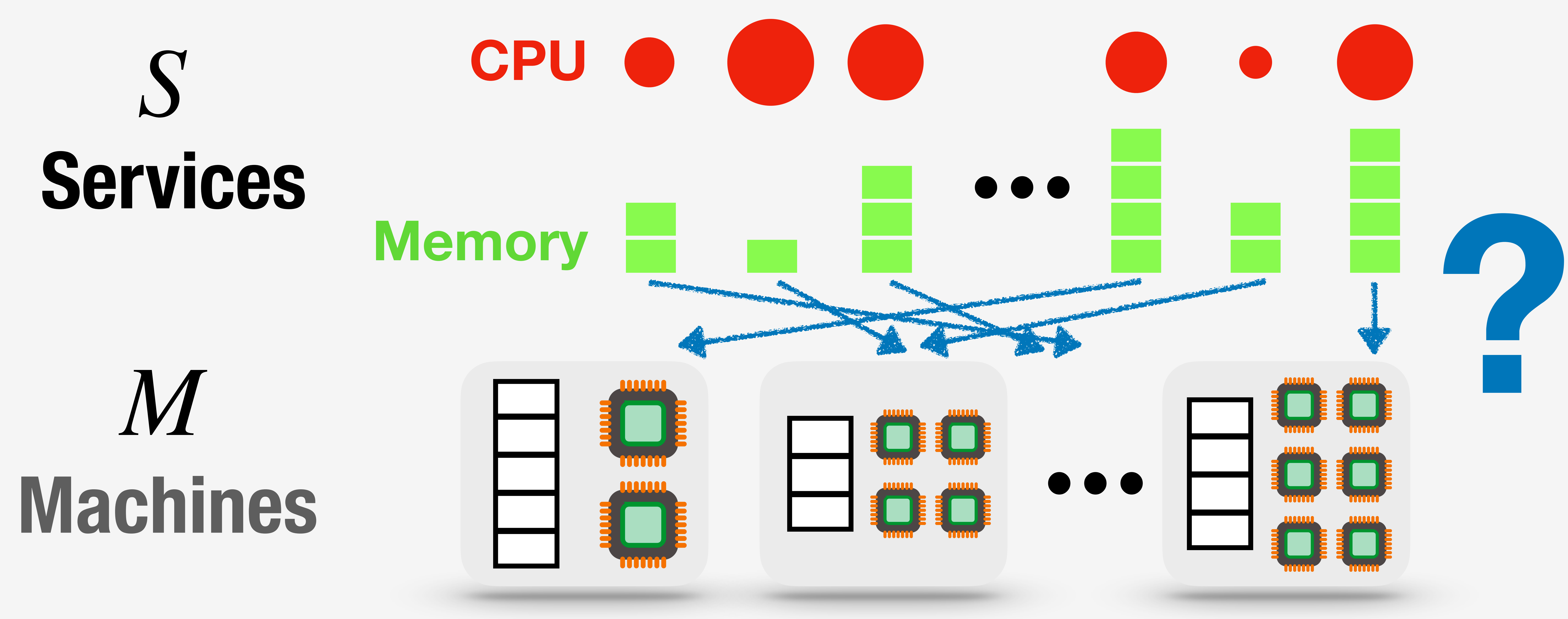


$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$$x \in \{0,1\}^{S \times M},\ y \in \{0,1\}^M$$

minimize $\sum_{m=1}^{M} y_m$



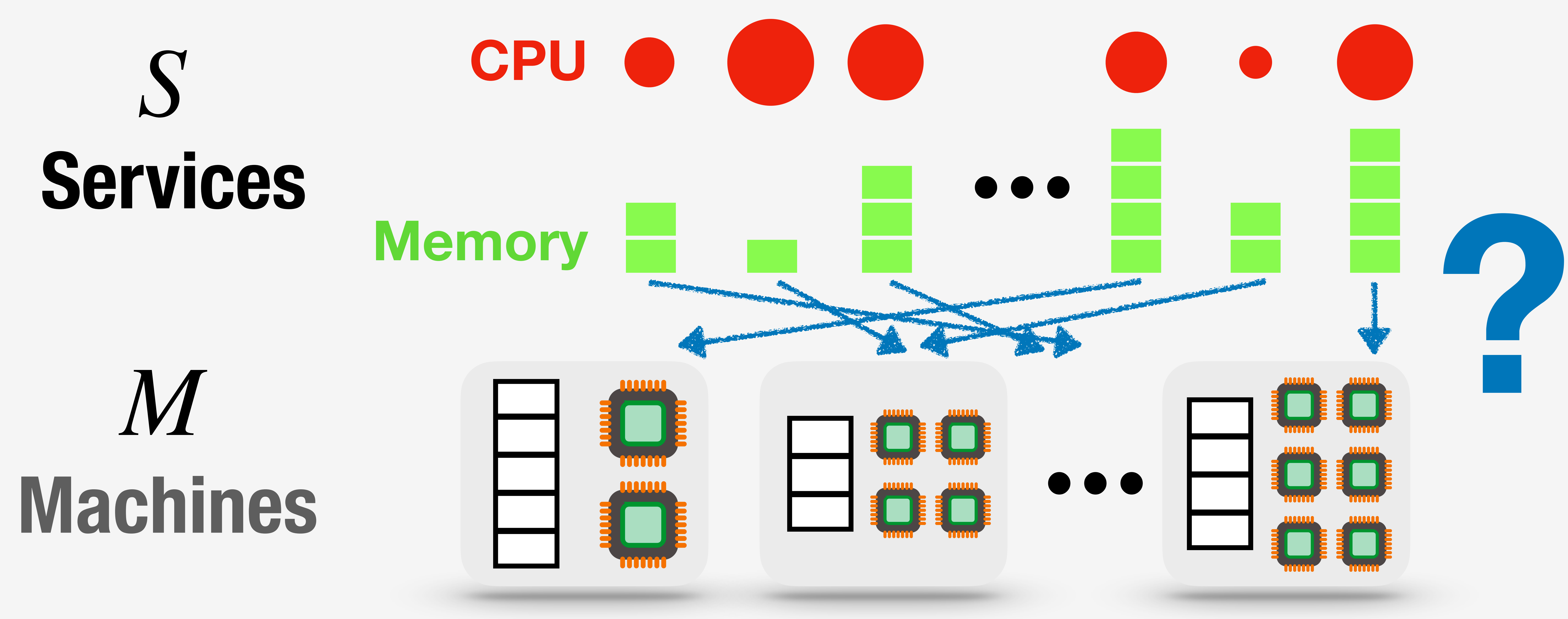
$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$$x \in \{0,1\}^{S \times M}, \; y \in \{0,1\}^M$$

minimize $\sum_{m=1}^{M} y_m$

Constraints:

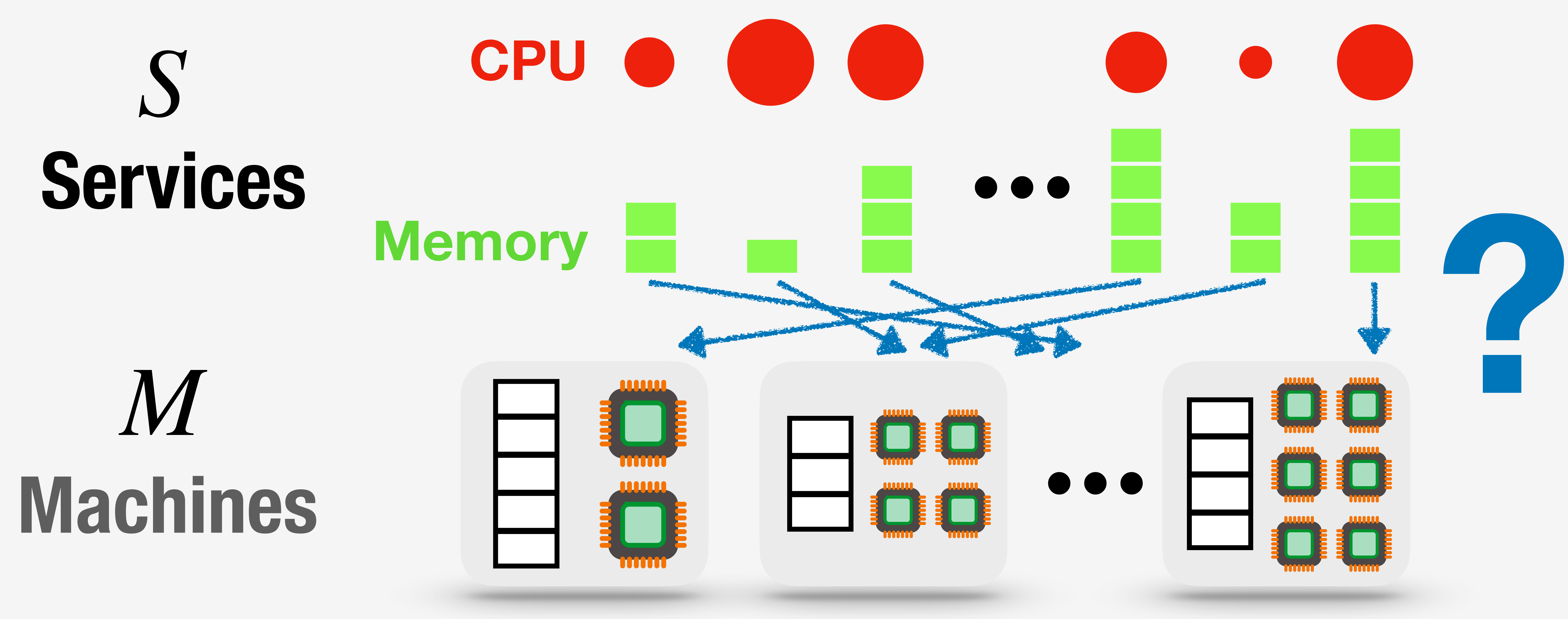


$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$$x \in \{0,1\}^{S\times M},\ y \in \{0,1\}^M$$

$$\textbf{minimize} \sum_{m=1}^{M} y_m$$



Constraints:

Each service on one machine only

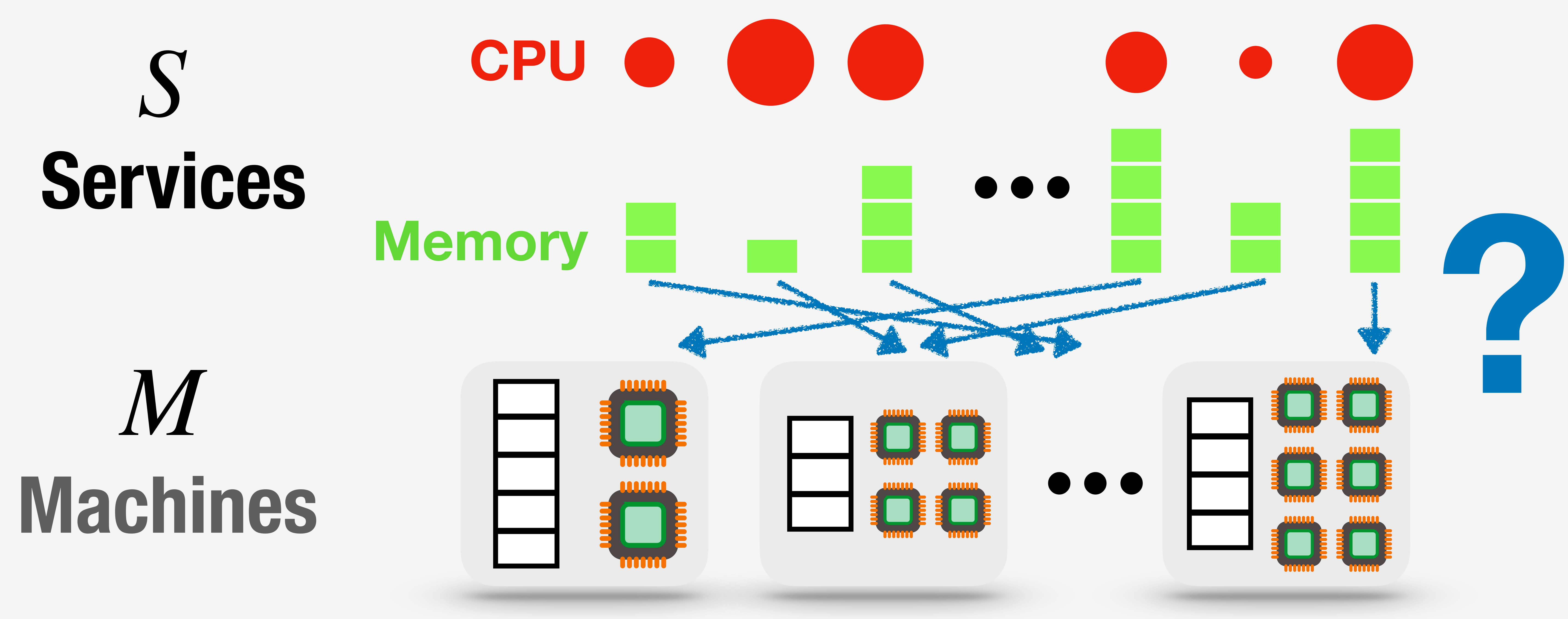
$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}, y \in \{0,1\}^M$

minimize $\sum_{m=1}^{M} y_m$



Constraints:

Each service on one machine only

$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

$$y_m \geq x_{s,m} \quad \forall s, m$$

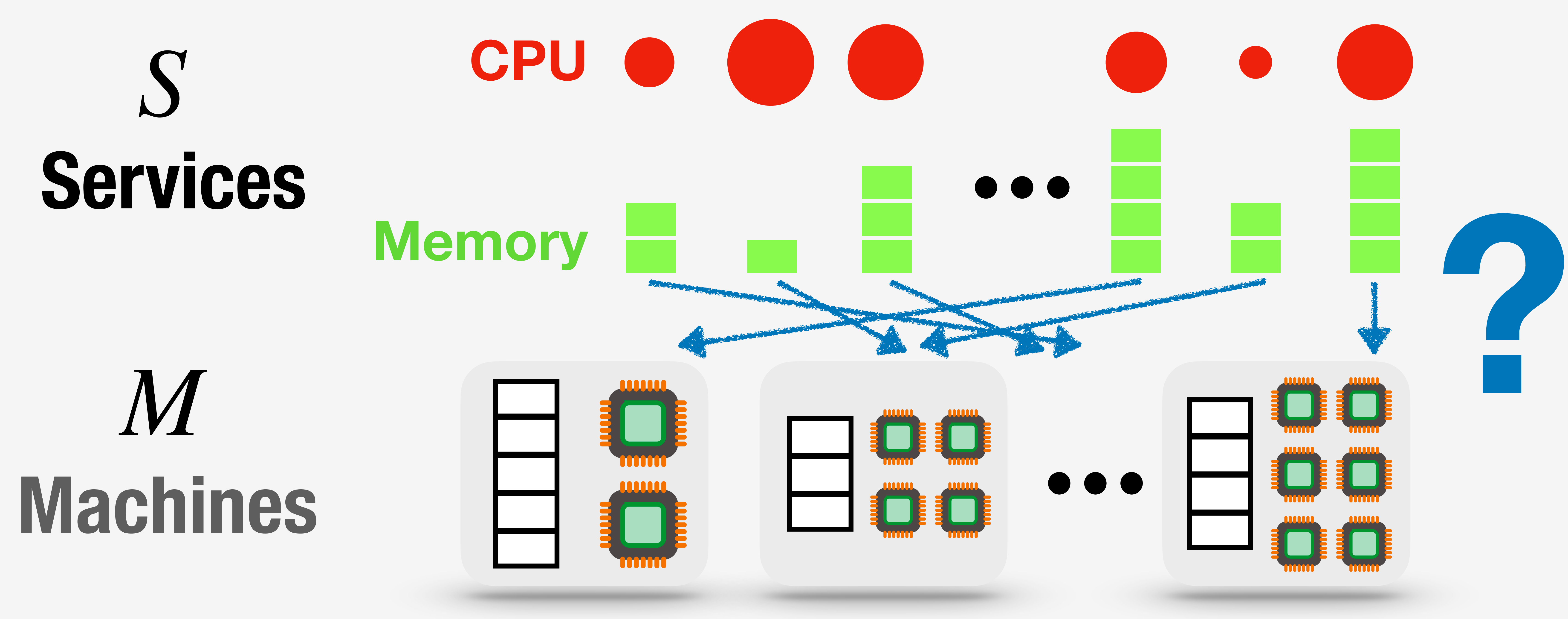
Machine is "ON" if a job is assigned to it

$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M},\ y \in \{0,1\}^M$

minimize $\sum_{m=1}^{M} y_m$



Constraints:

Each service on one machine only

$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

Memory capacity

$$\sum_{s=1}^{S} \textbf{mem}(s) \cdot x_{s,m} \leq \textbf{cap-mem}(m) \quad \forall m$$

$$y_m \geq x_{s,m} \quad \forall s, m$$

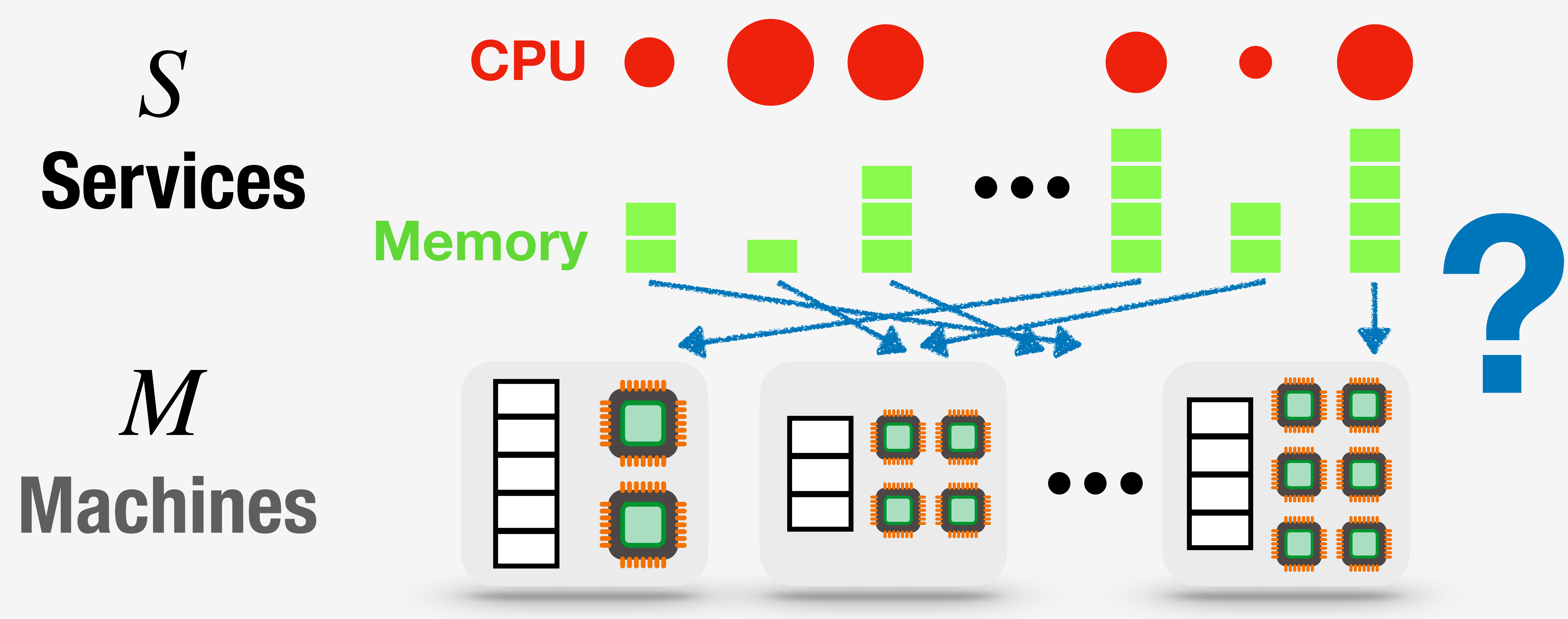
Machine is "ON" if a job is assigned to it

$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$$x \in \{0,1\}^{S \times M},\ y \in \{0,1\}^M$$

$$\textbf{minimize} \sum_{m=1}^{M} y_m$$



Constraints:

Each service on one machine only

$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

Memory capacity

$$\sum_{s=1}^{S} \textbf{mem}(s) \cdot x_{s,m} \leq \textbf{cap-mem}(m) \quad \forall m$$

$$y_m \geq x_{s,m} \quad \forall s, m$$

Machine is "ON" if a job is assigned to it

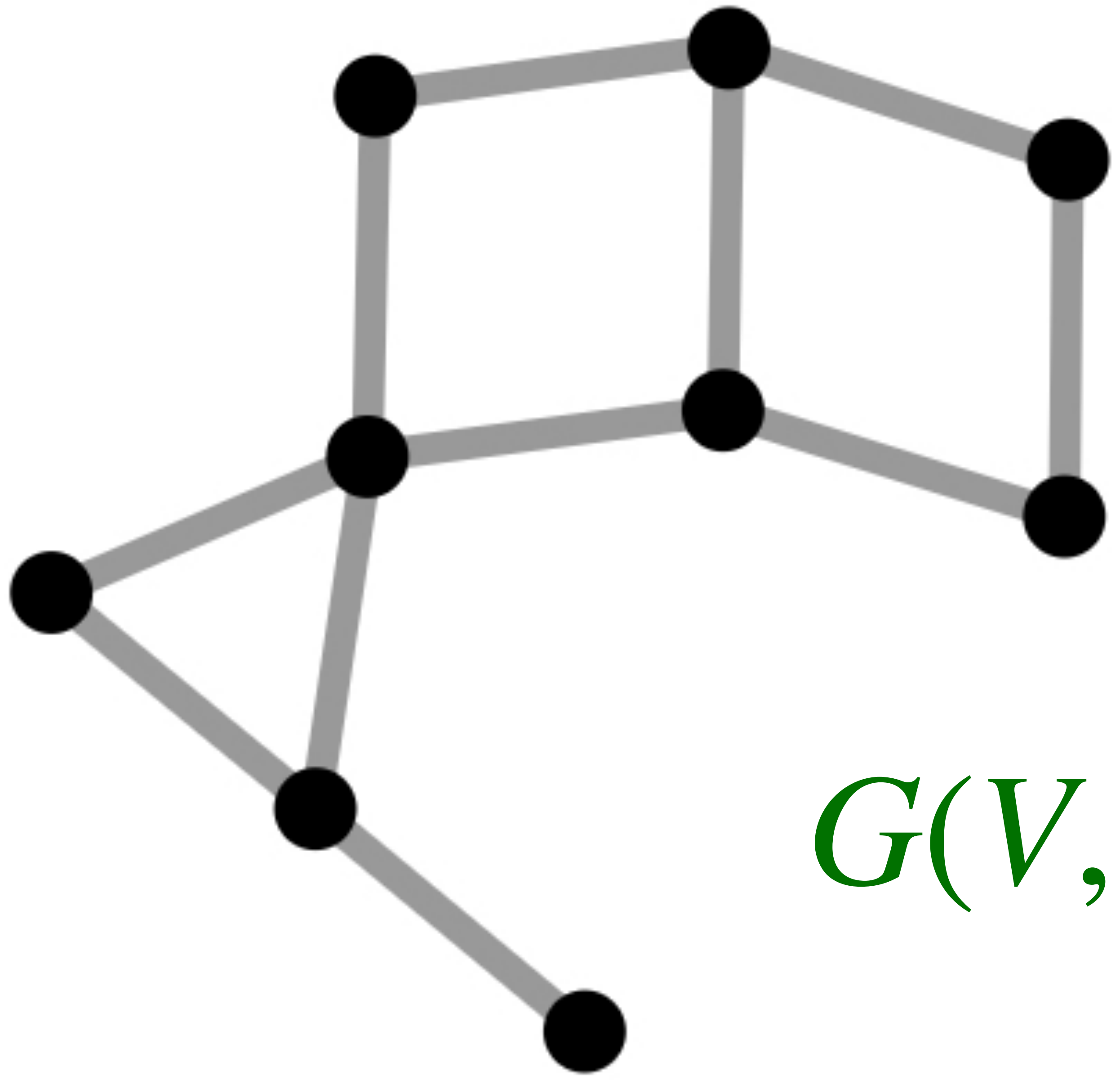
$$\sum_{s=1}^{S} \textbf{cpu}(s) \cdot x_{s,m} \leq \textbf{cap-cpu}(m) \quad \forall m$$

Processor capacity

Forest Harvesting

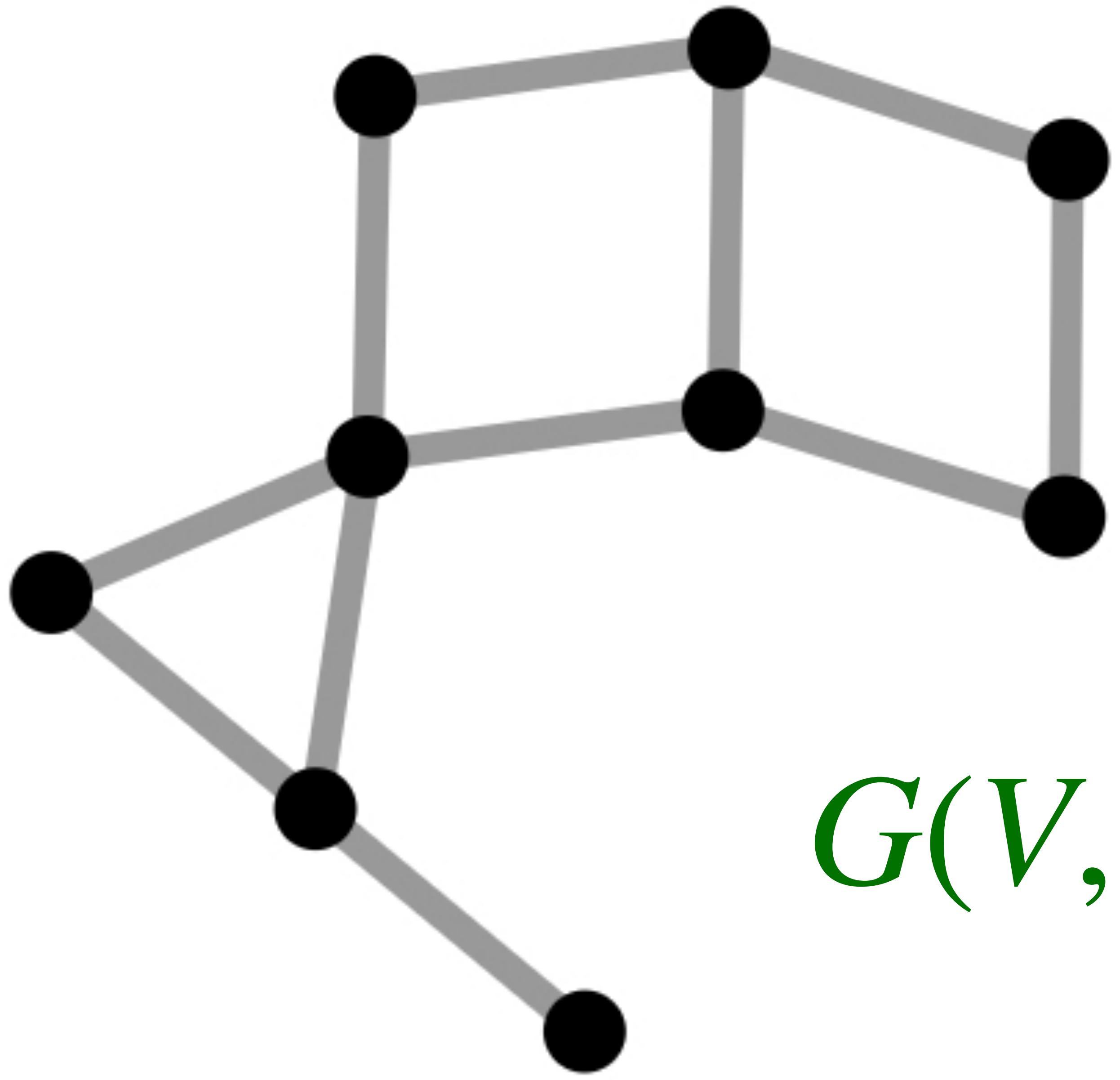
Forest Harvesting

Forest Harvesting



$G(V, E)$

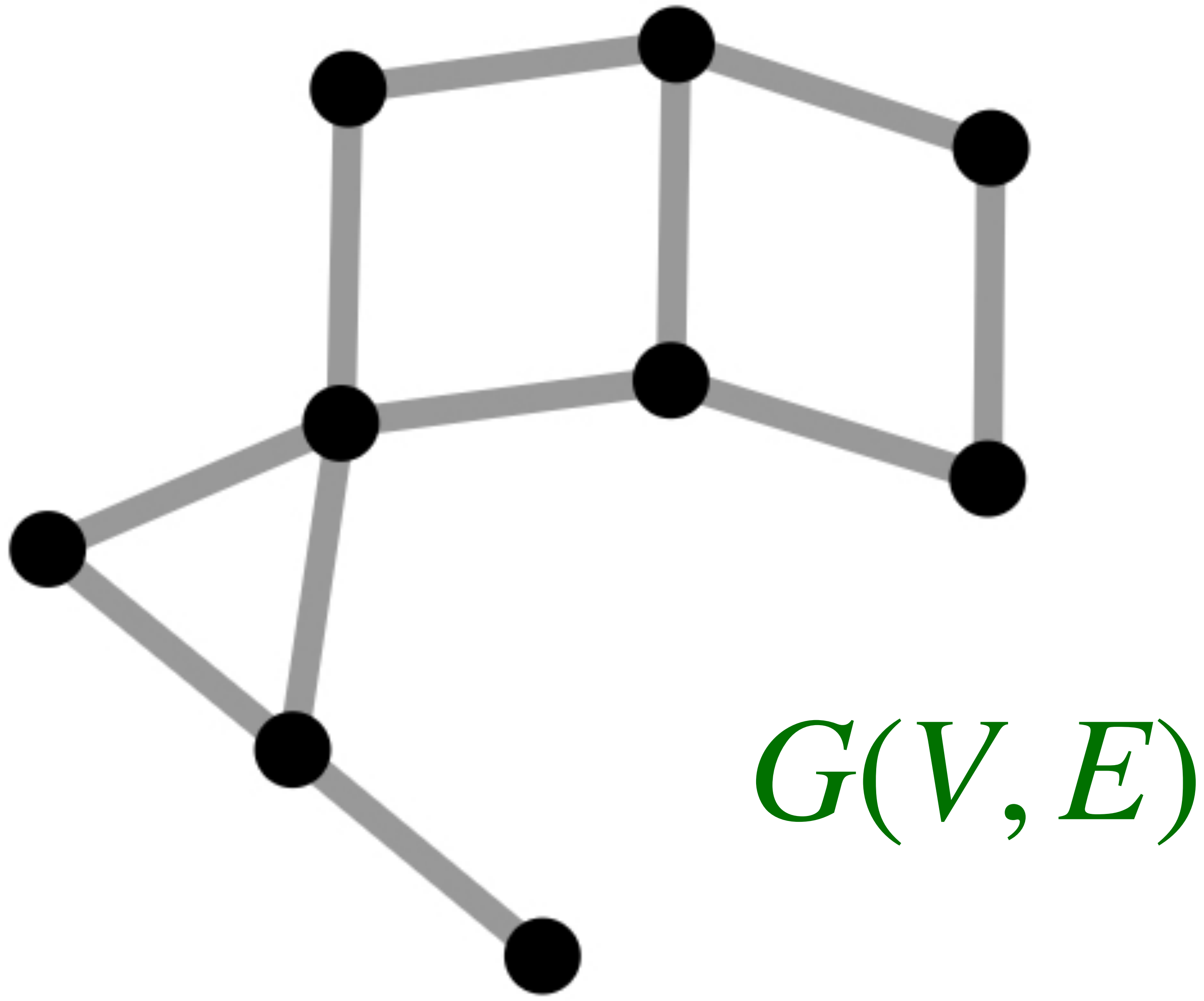
Forest Harvesting



$G(V, E)$

Forest Harvesting

Goal: Harvest subset of parcels to maximize **revenue**; pay **cost** for harvesting adjacent parcels



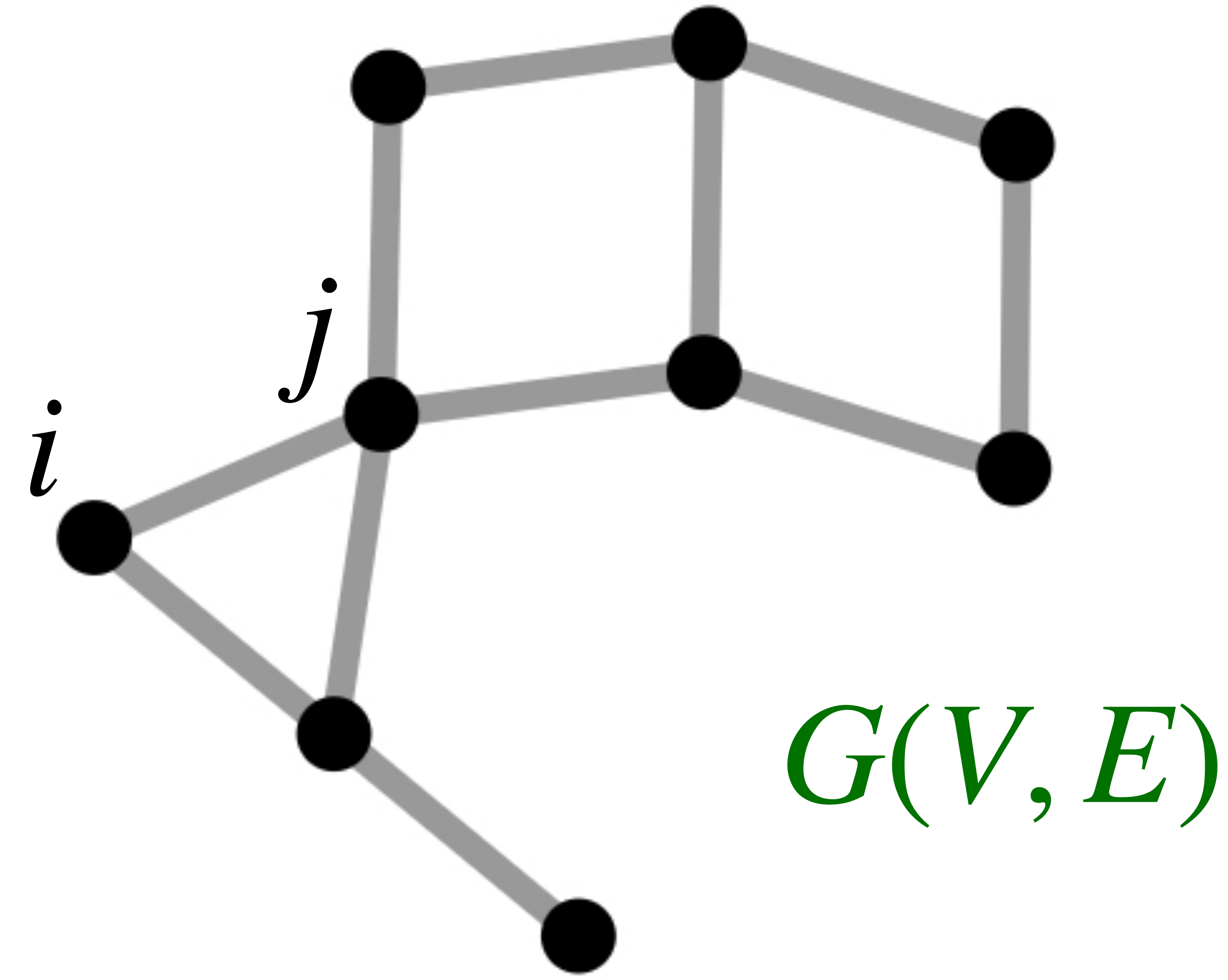
Forest Harvesting

Goal: Harvest subset of parcels to maximize **revenue**; pay **cost** for harvesting adjacent parcels

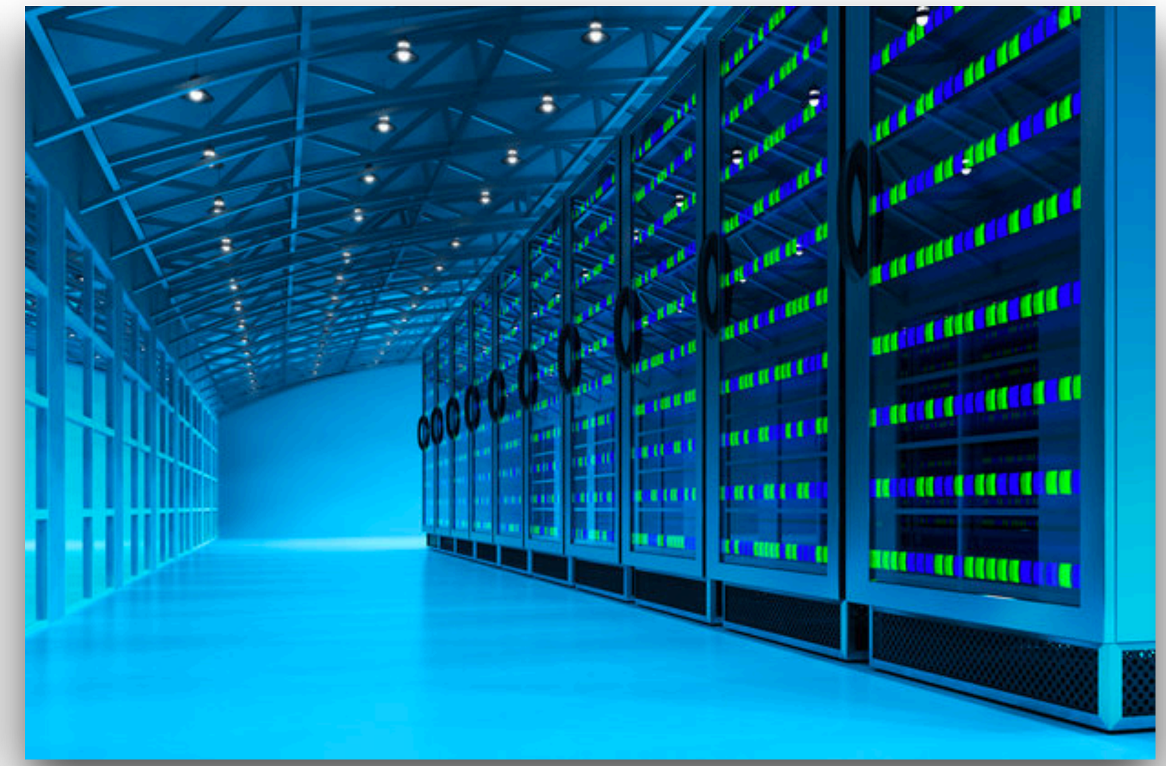
$$\textbf{maximize} \sum_{i \in V} r_i x_i - \sum_{(i,j) \in E} c_{ij} y_{ij}$$

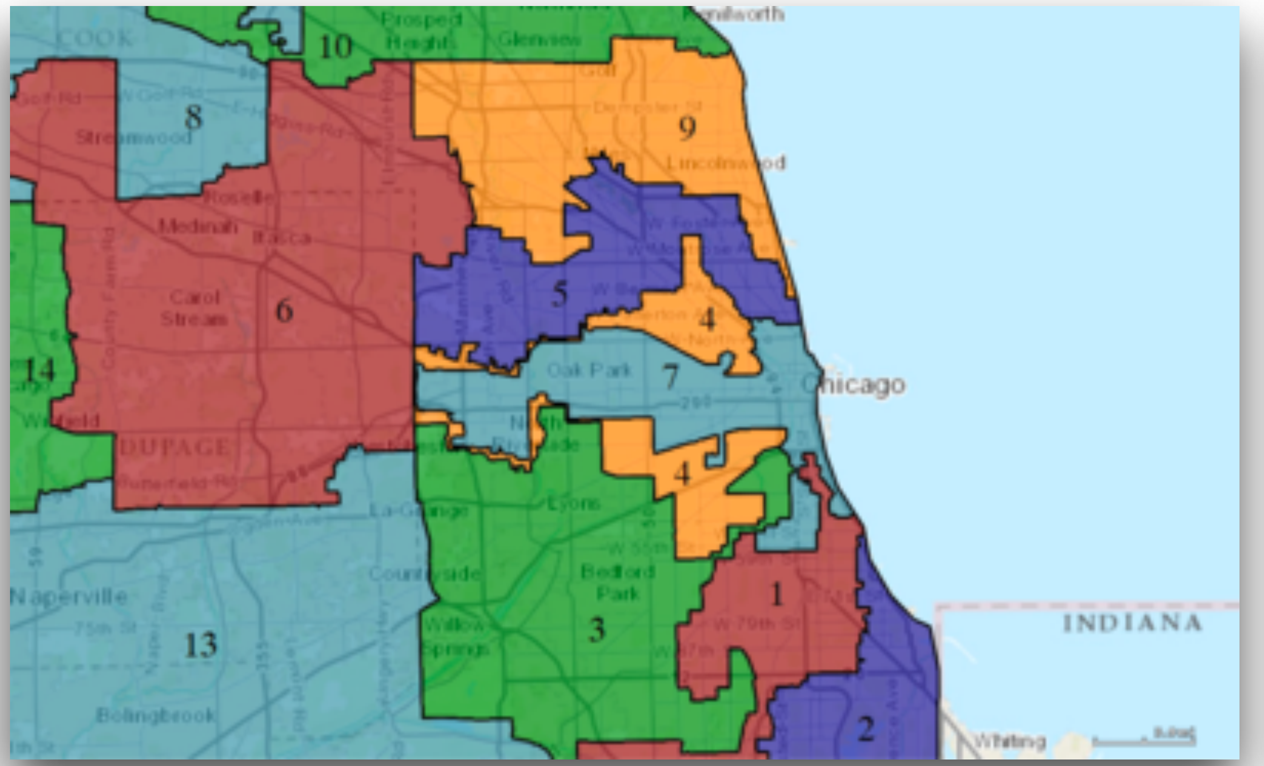
$$\textbf{subject to } x_i + x_j - y_{ij} \leq 1$$

$$x \in \{0,1\}^n, \; y \in \{0,1\}^m$$



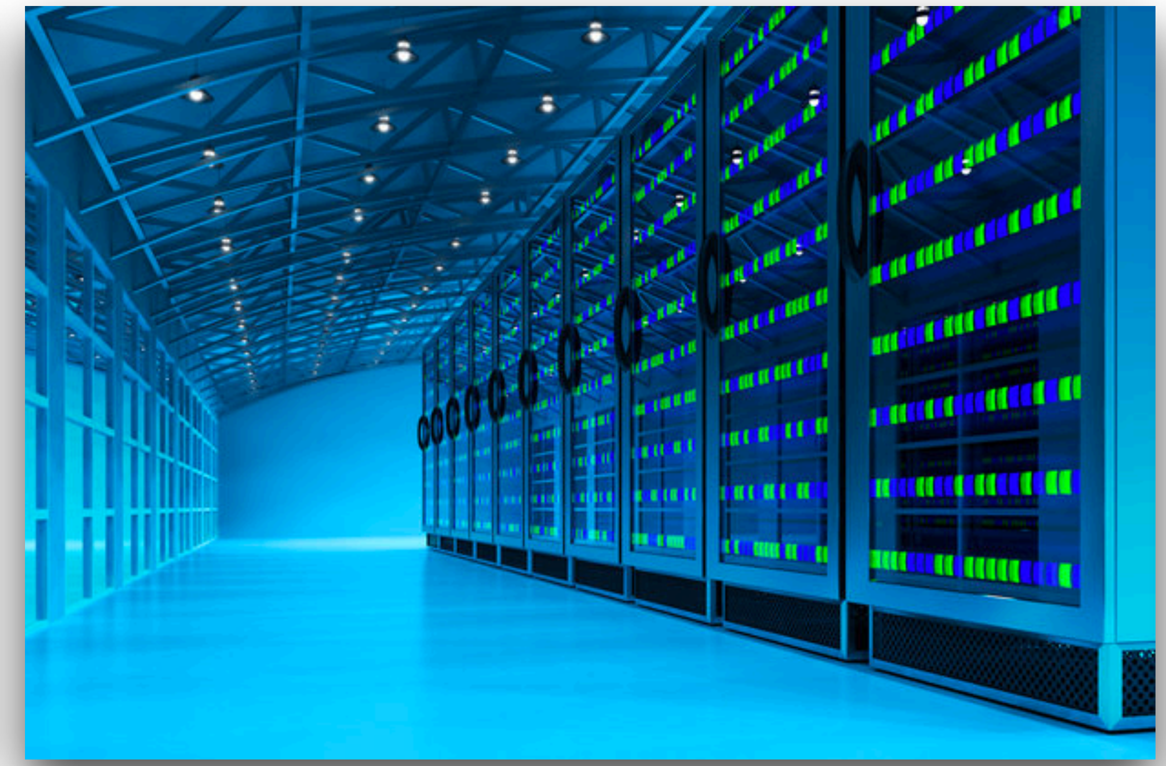
Auction Design **Data Center Management** **Political Districting** **Kidney Exchange**



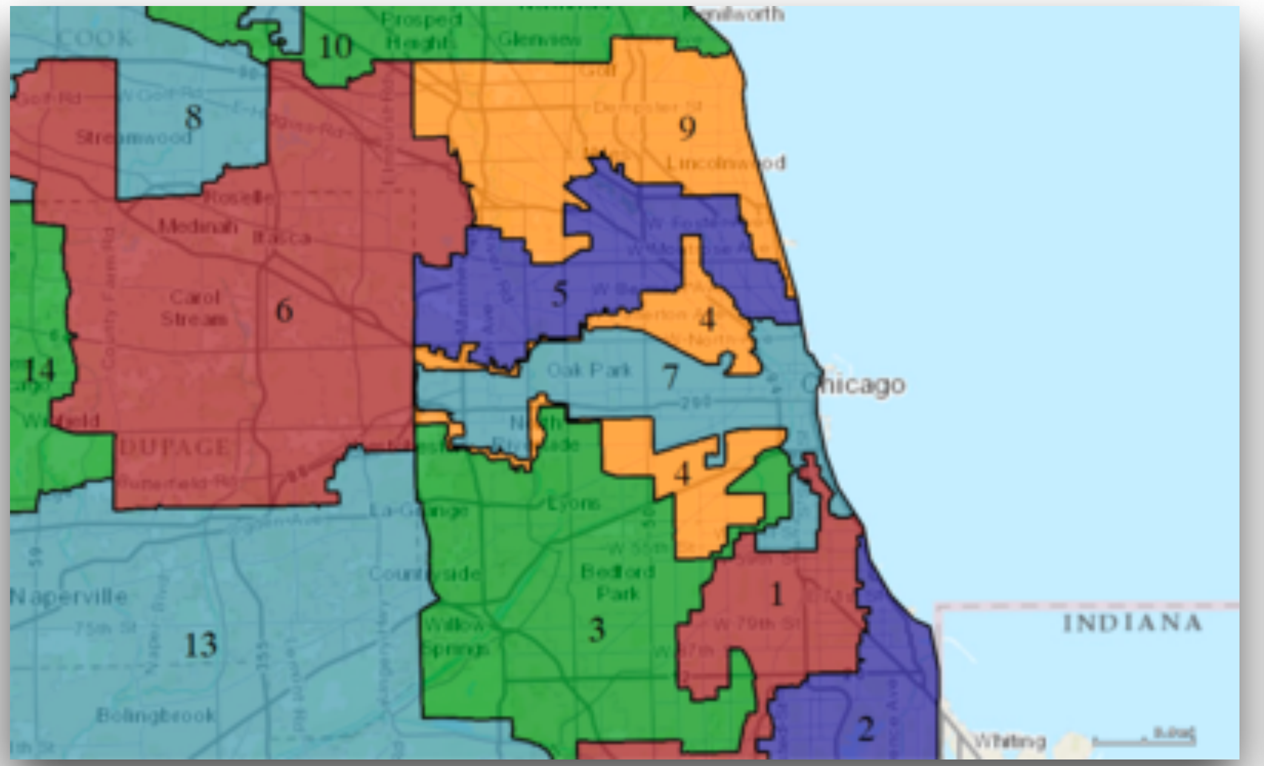


Auction Design

Data Center Management

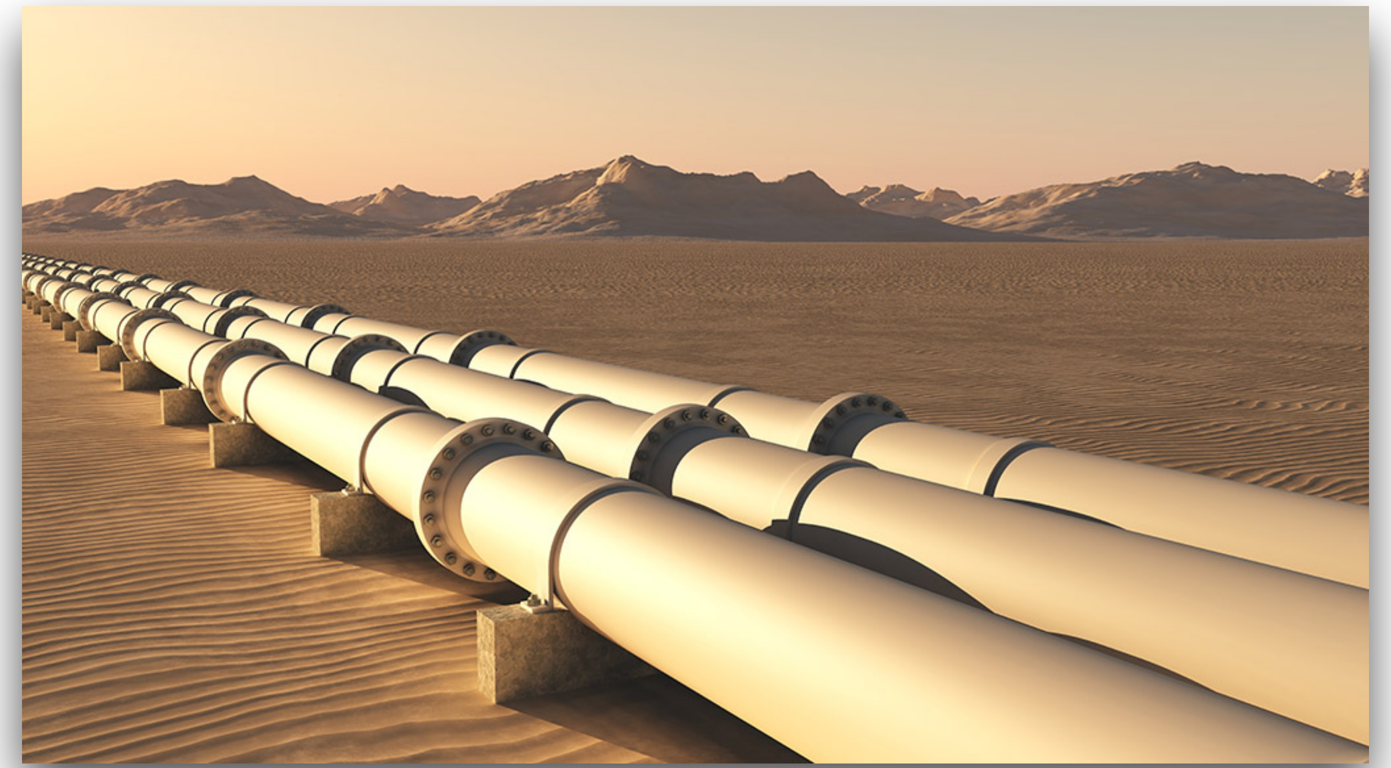


Political Districting

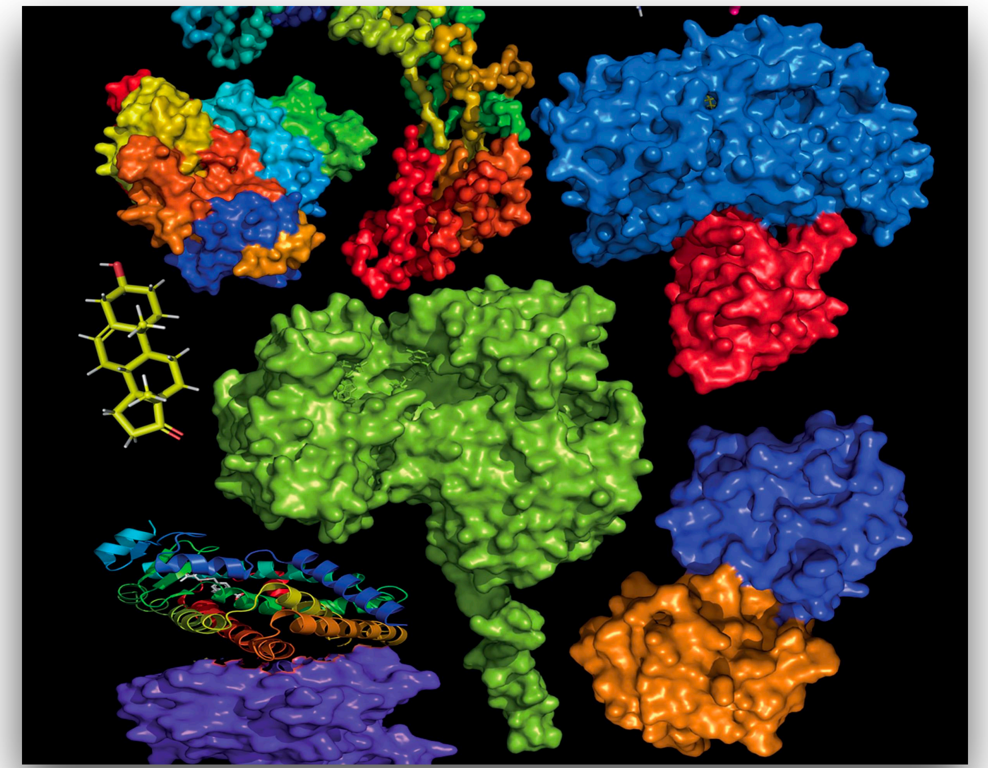


Kidney Exchange

Energy Systems



Scientific Discovery

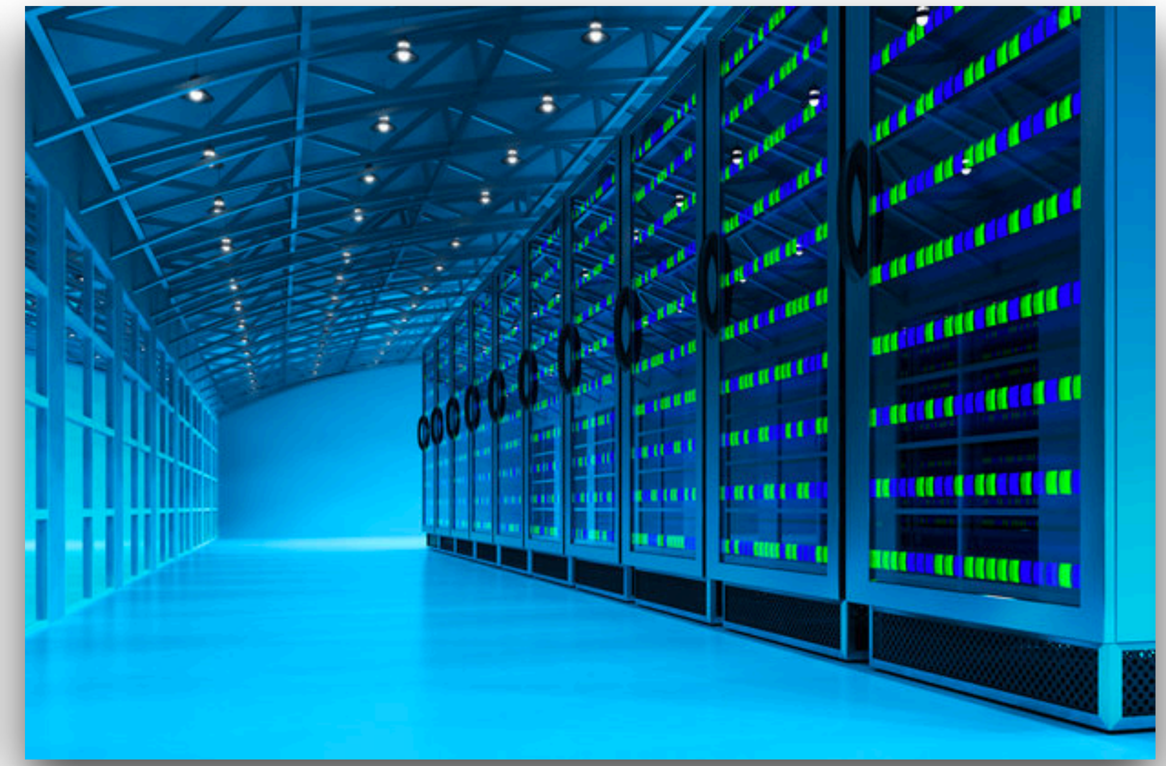


Ridesharing

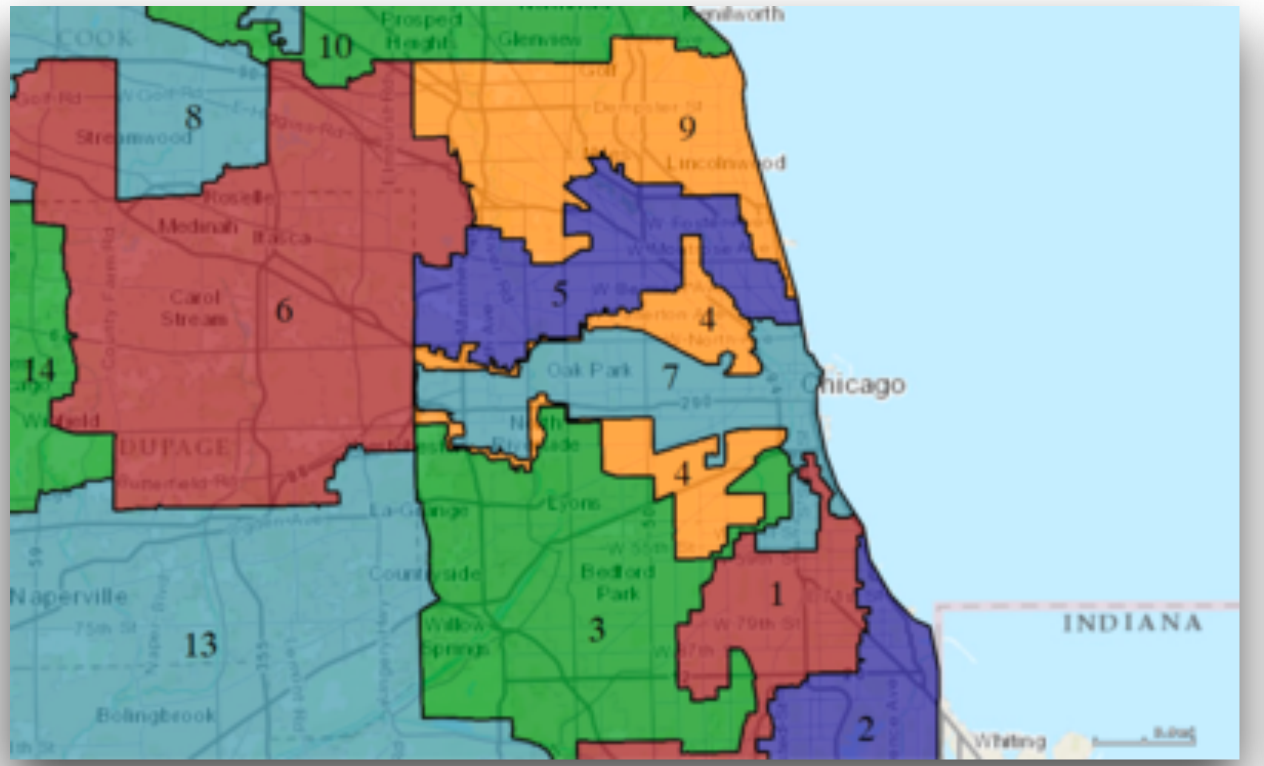
Cancer Therapeutics

Auction Design

Data Center Management

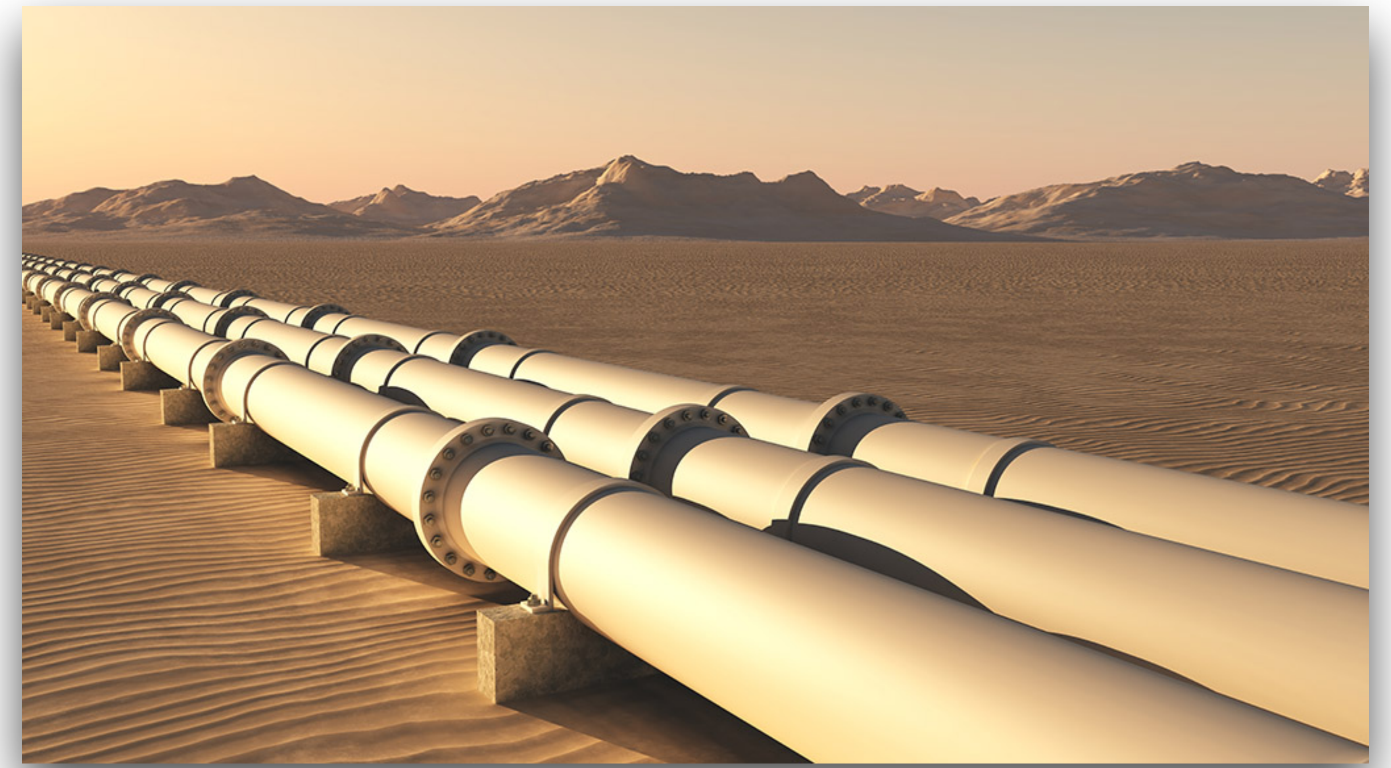


Political Districting

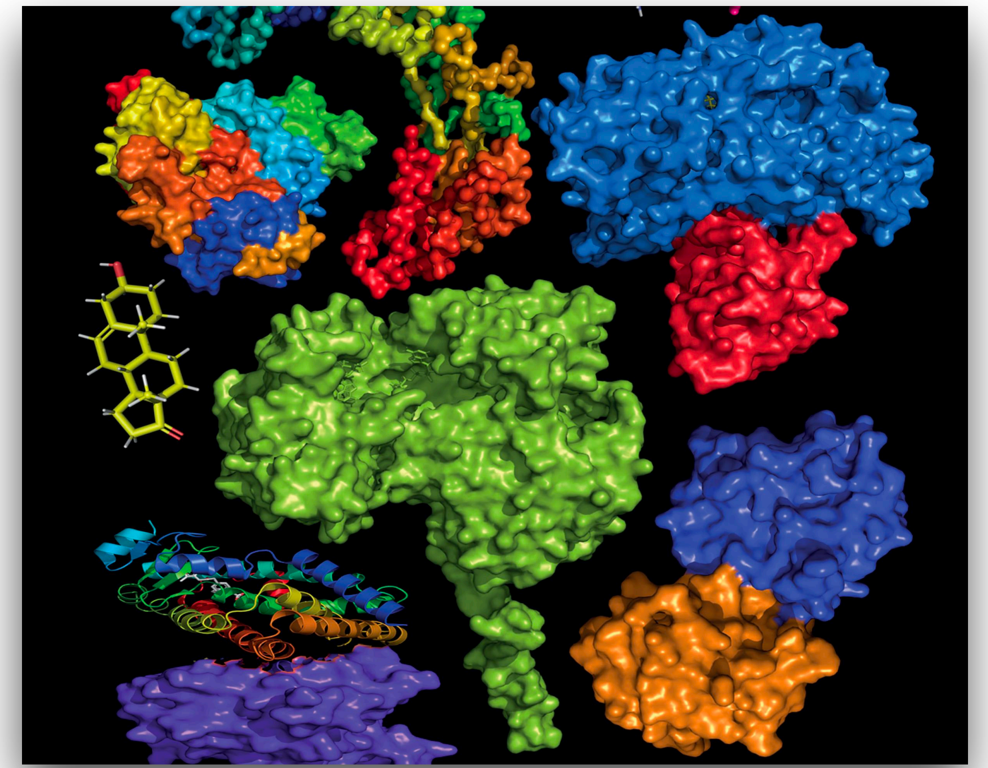


Kidney Exchange

Energy Systems



Scientific Discovery



Ridesharing

Cancer Therapeutics

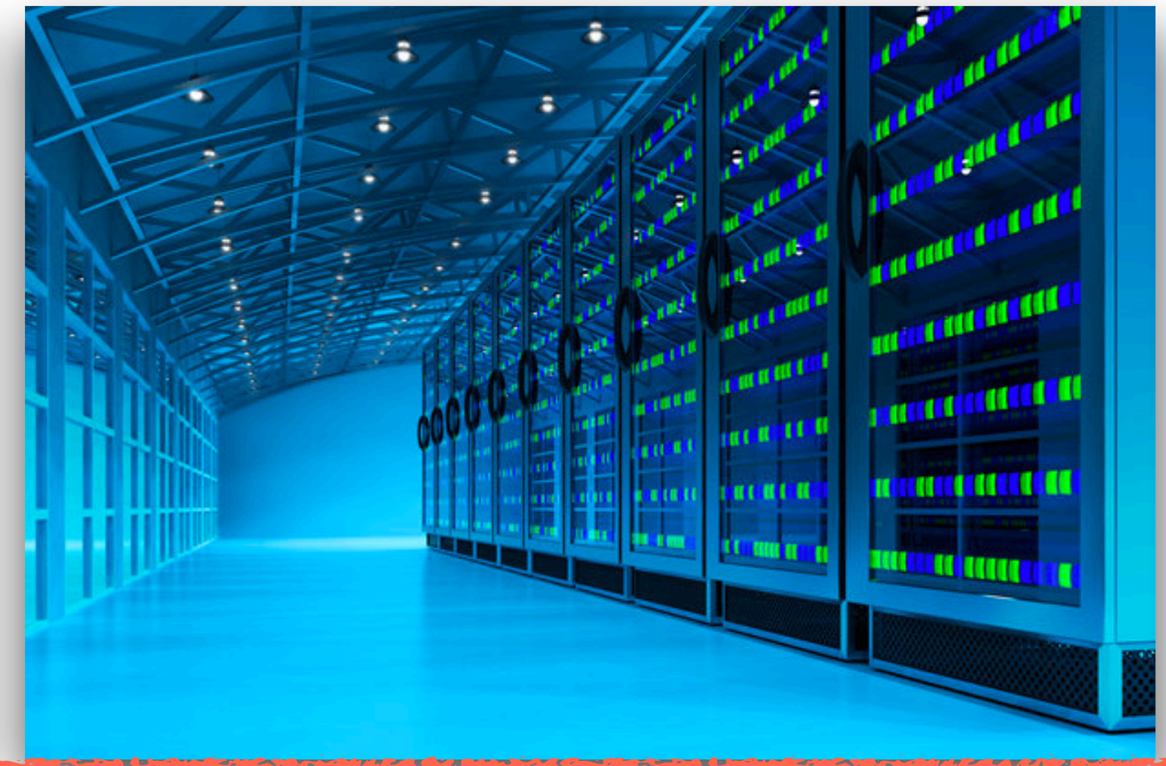
Airline Scheduling

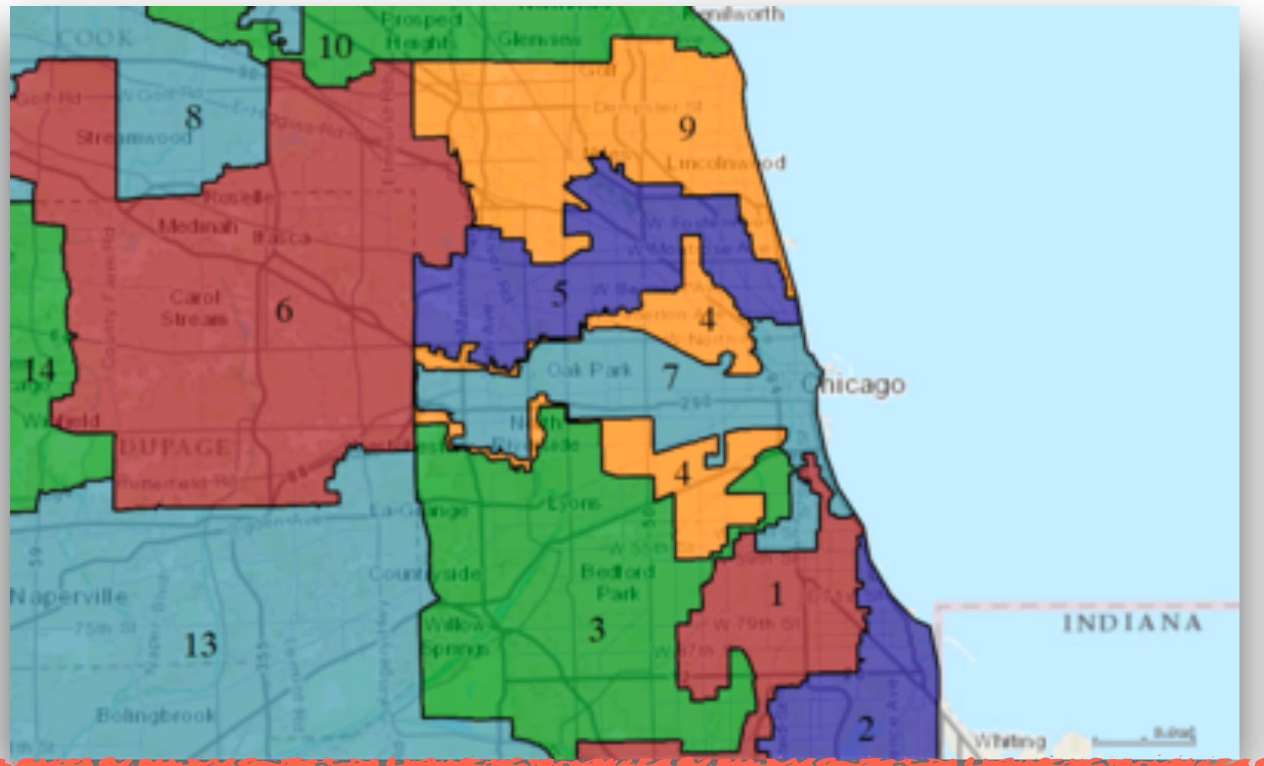
Conservation Planning

Disaster Response

College Admissions

Auction Design | Data Center Management | Political Districting | Kidney Exchange



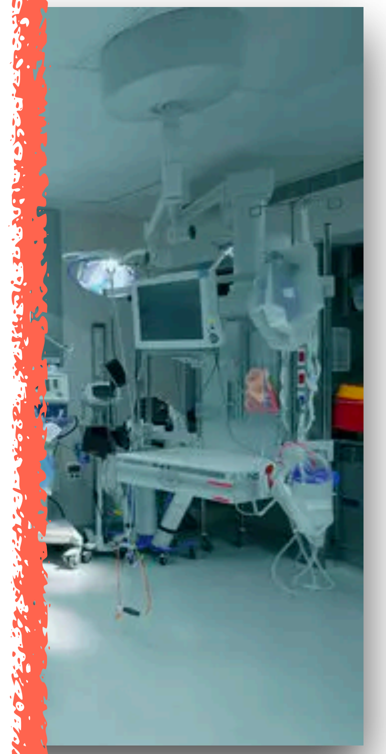


E | utics

> 50% of INFORMS Edelman Award winners use Discrete Optimization

→ **Billions ($) in savings/profit**

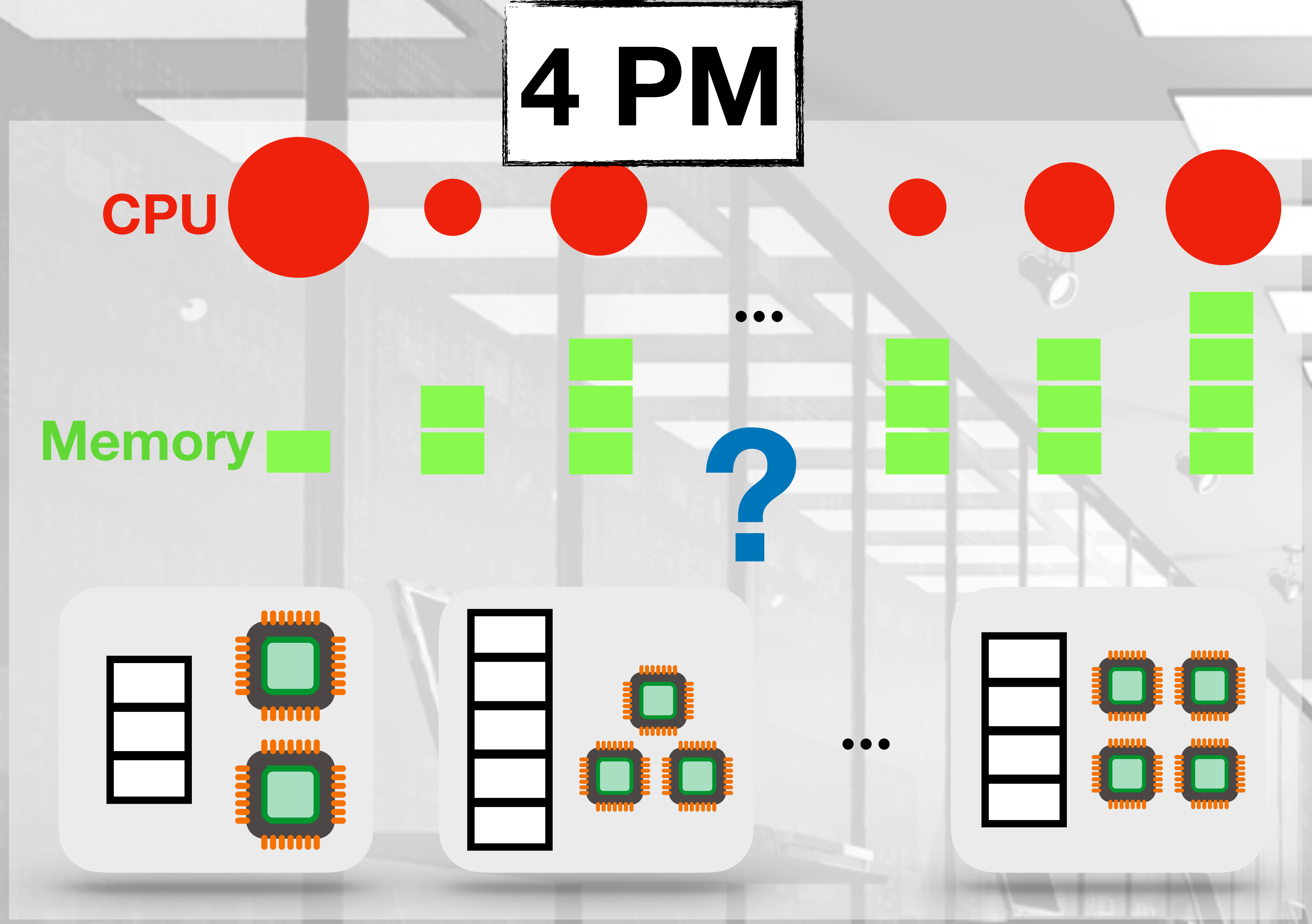
George Nemhauser, Plenary at EURO INFORMS, 2013



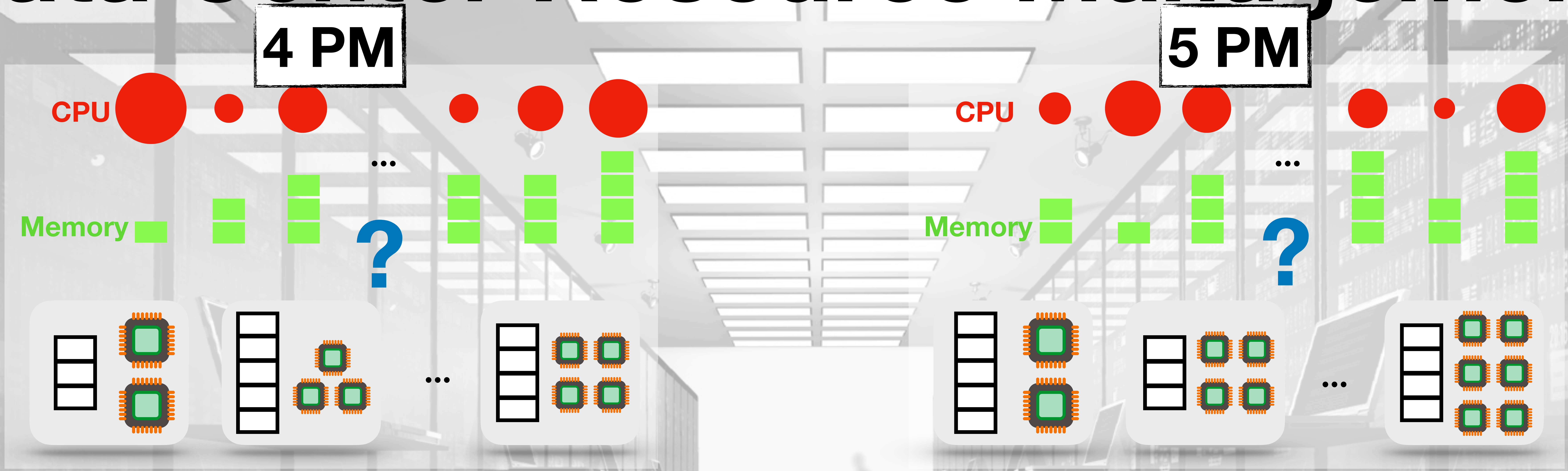
Airlin | ions

Data Center Resource Management

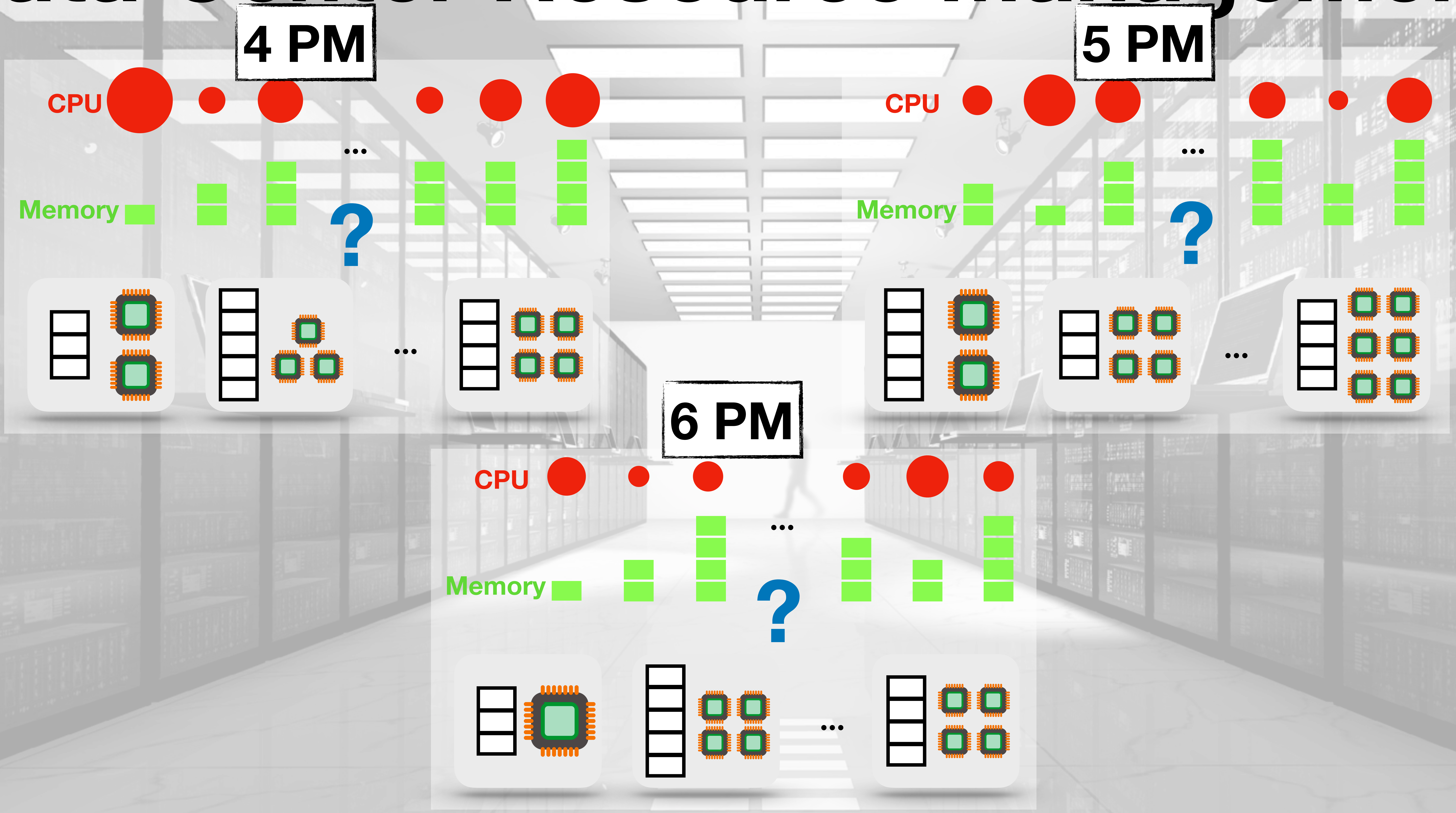
Data Center Resource Management



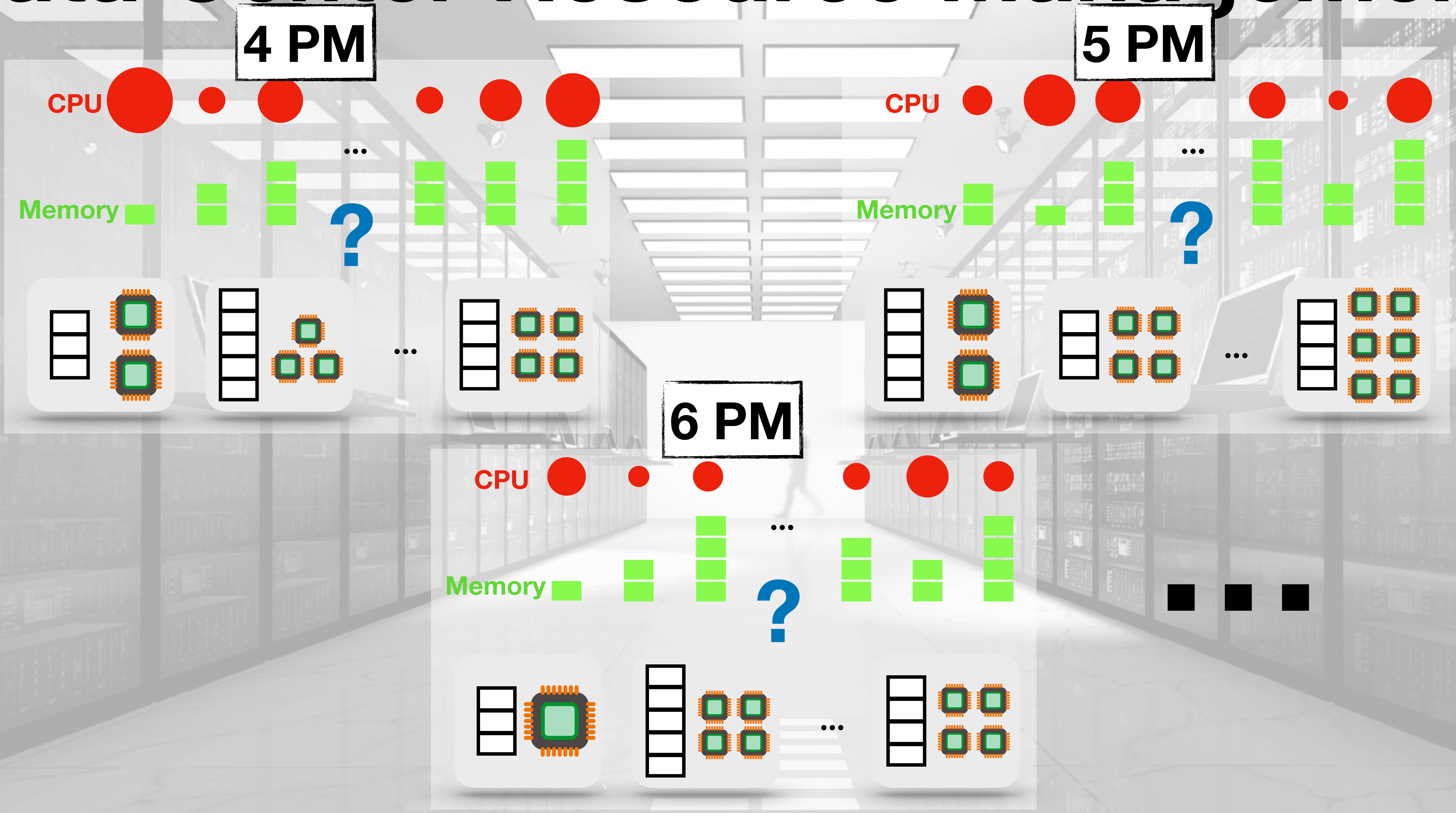
Data Center Resource Management



Data Center Resource Management



Data Center Resource Management



Tackling NP-Hard Problems

Paradigm	Design Rationale

Tackling NP-Hard Problems

Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers

Tackling NP-Hard Problems

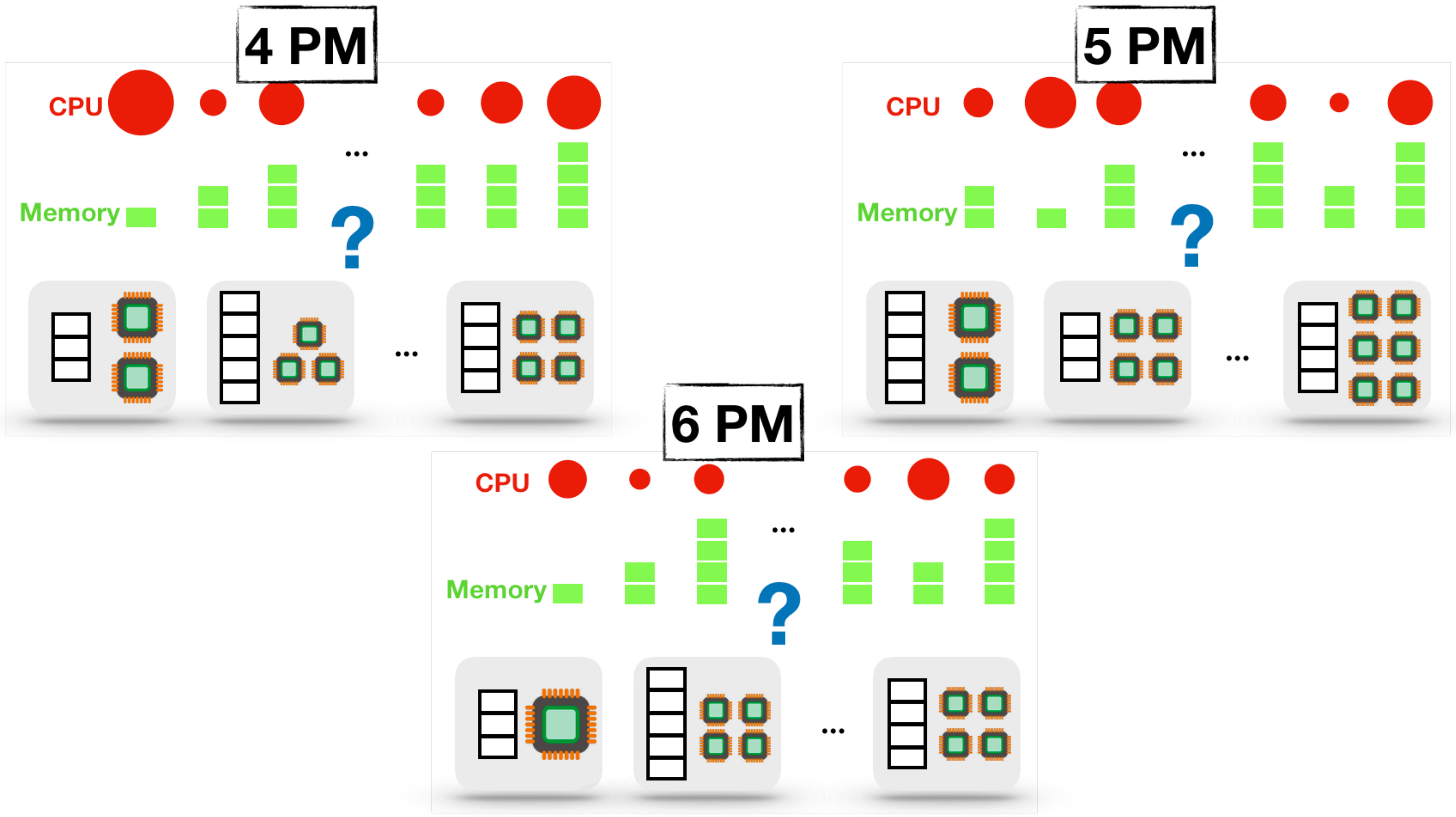
Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees

Tackling NP-Hard Problems

Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

Tackling NP-Hard Problems

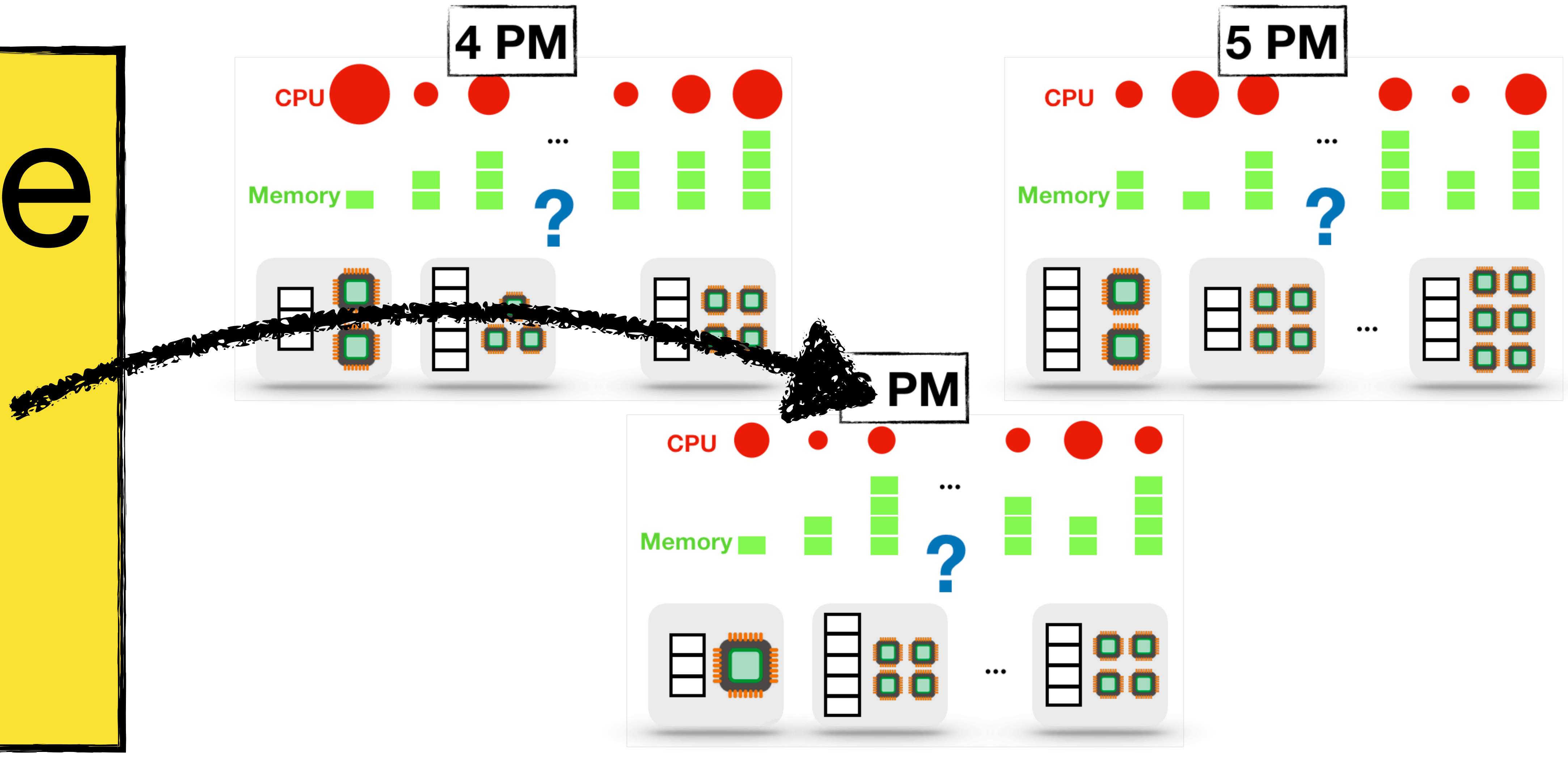
Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**



Tackling NP-Hard Problems

Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

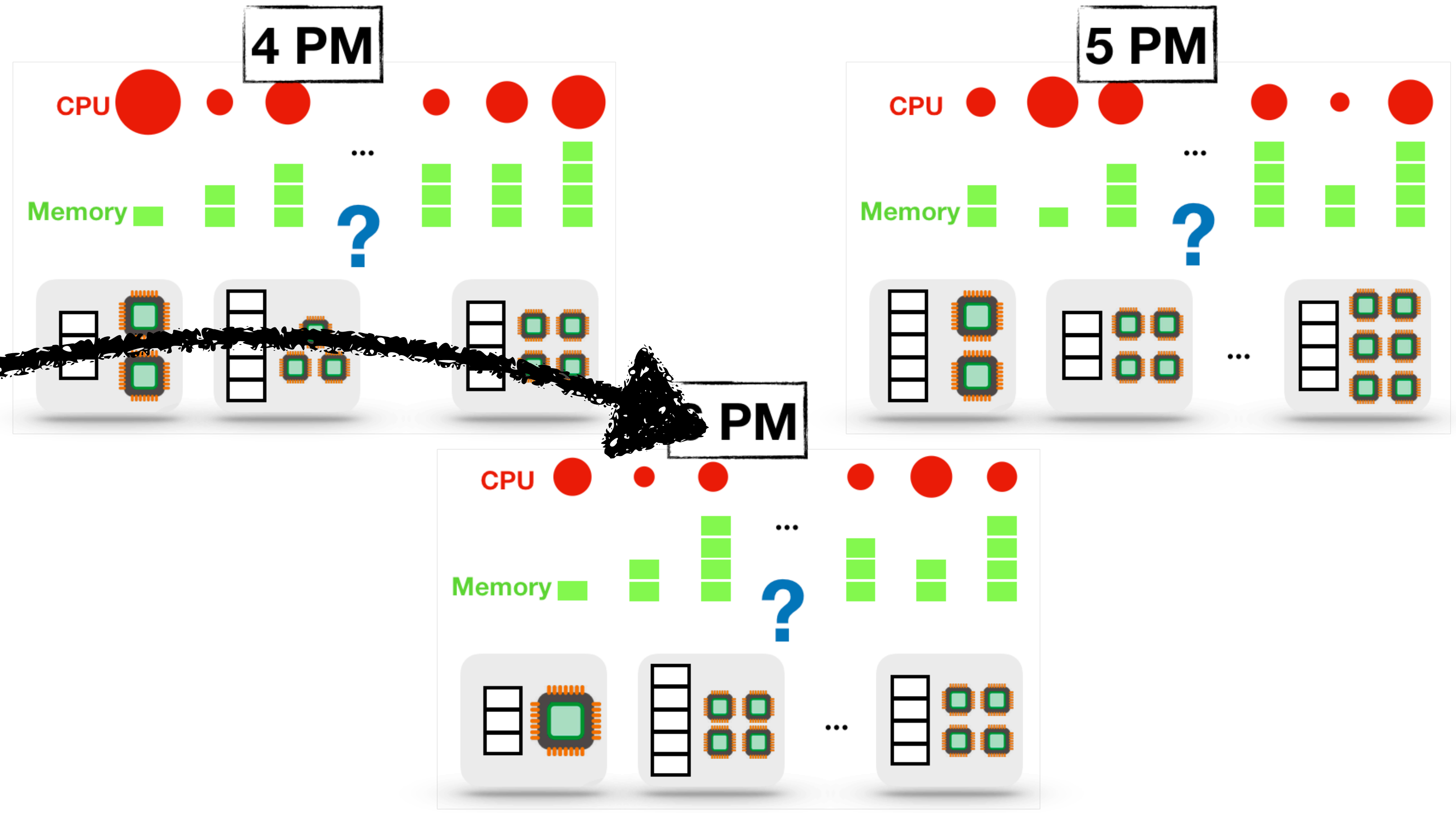
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via...
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

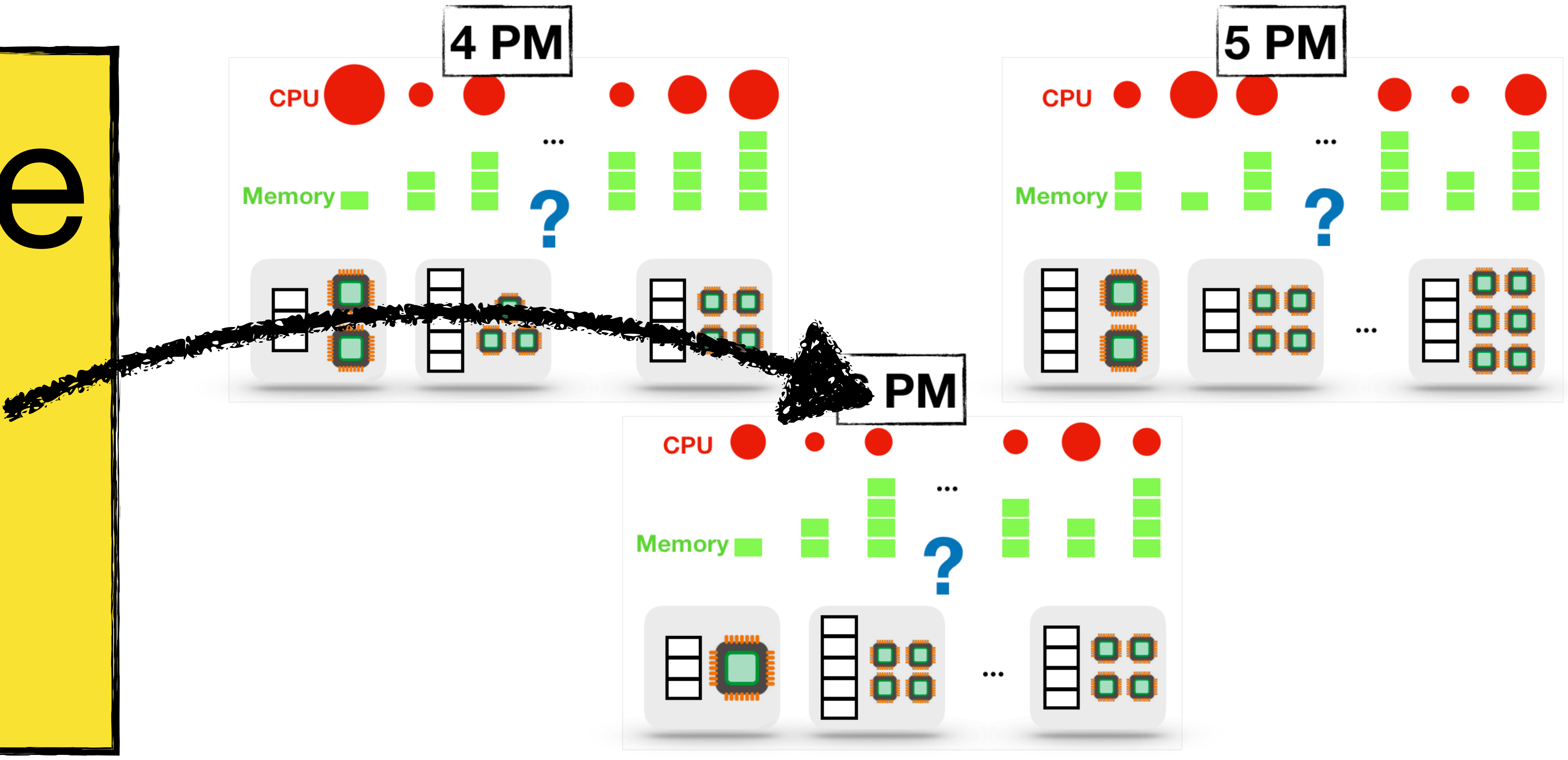
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via...
Exhaustive Search	**Problem-Specific Bounding** functions or **search rules**
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

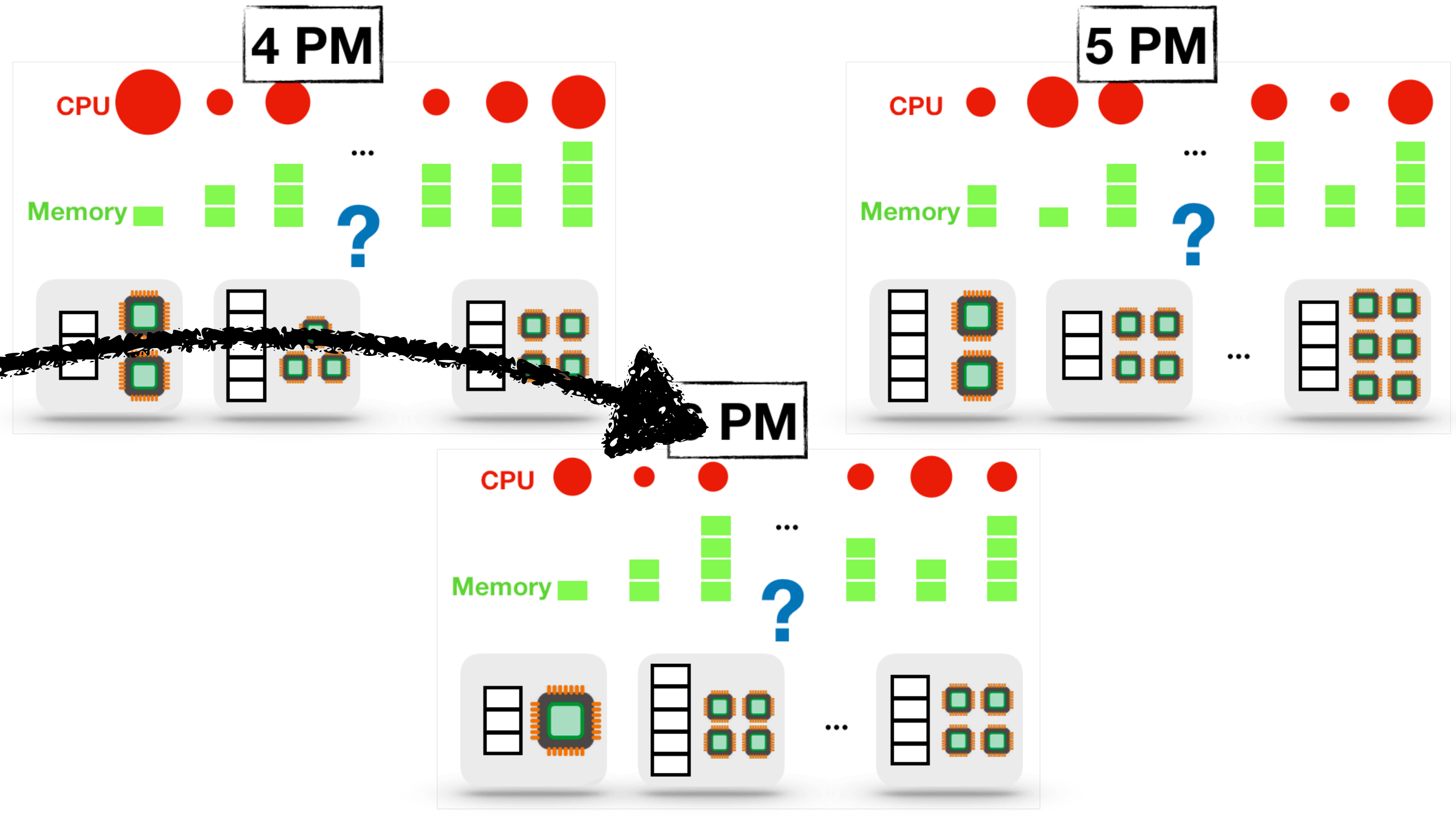
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via…
Exhaustive Search	**Problem-Specific Bounding** functions or **search rules**
Approximation Algorithms	Make explicit **assumptions** on input distribution and **redesign algo.**
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

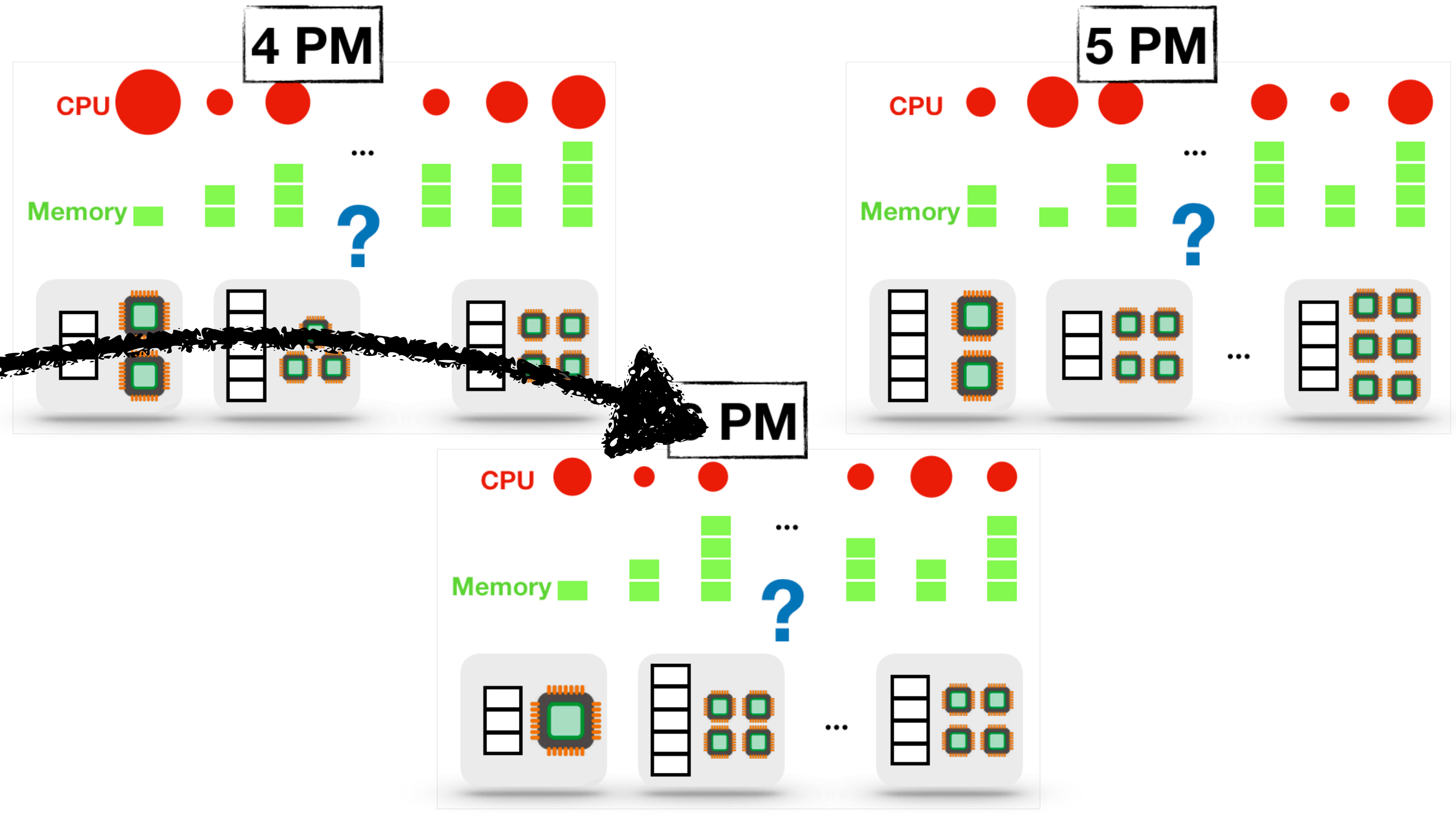
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via...
Exhaustive Search	**Problem-Specific Bounding** functions or **search rules**
Approximation Algorithms	Make explicit **assumptions** on input distribution and **redesign algo.**
Heuristics	**Analyze** algorithm **behavior** on your inputs; look for **patterns** to exploit

How do you **tailor** the algorithm to **YOUR** instances?

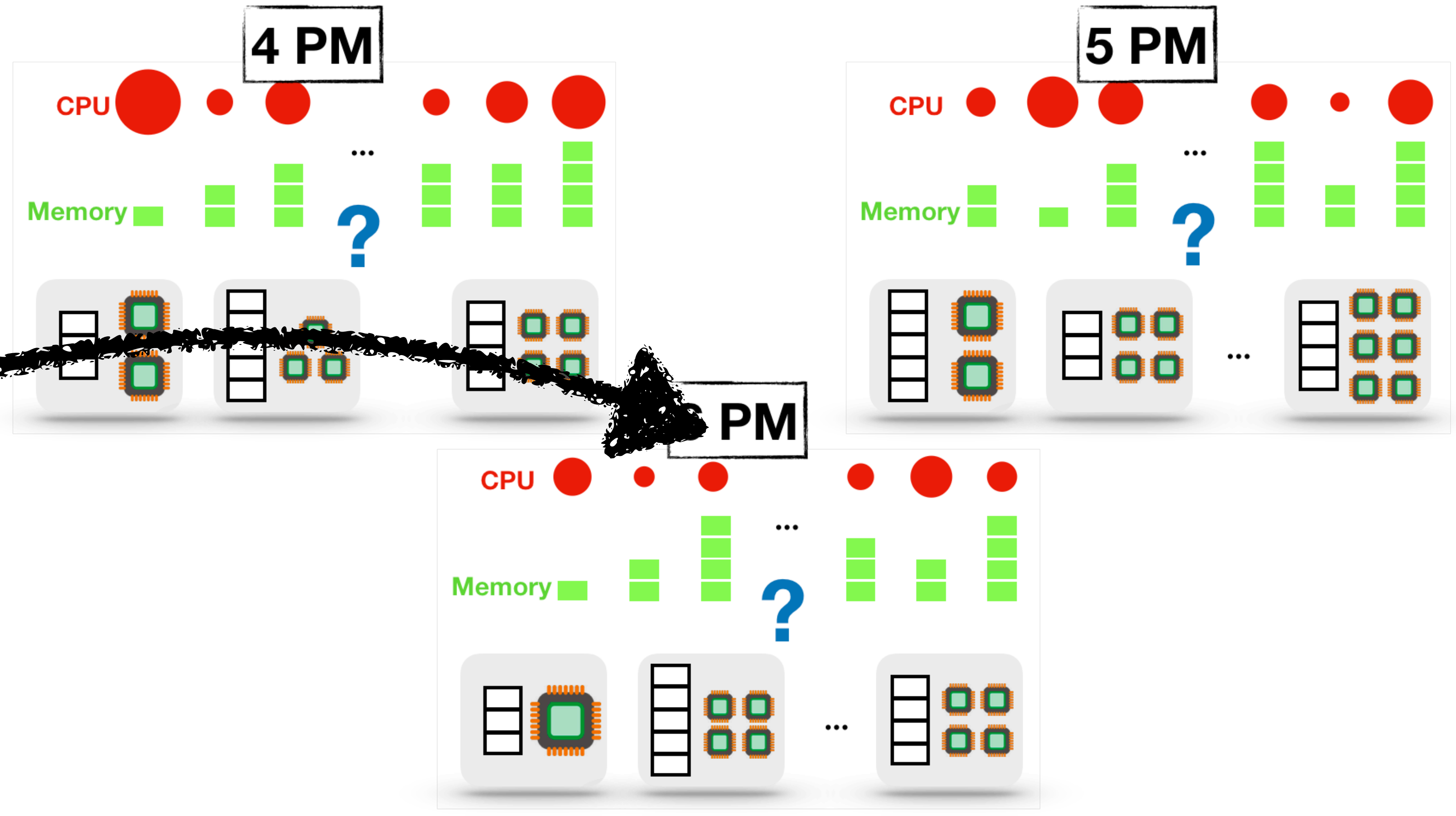


Tackling NP-Hard Problems

Paradigm	Customization via...
	Problem-Specific Bounding functions or **search rules**
	Make explicit **assumptions** on input distribution and **redesign algo.**
	Analyze algorithm **behavior** on your inputs; look for **patterns** to exploit

ANSWER:
Manual intellectual/ experimental **effort** required

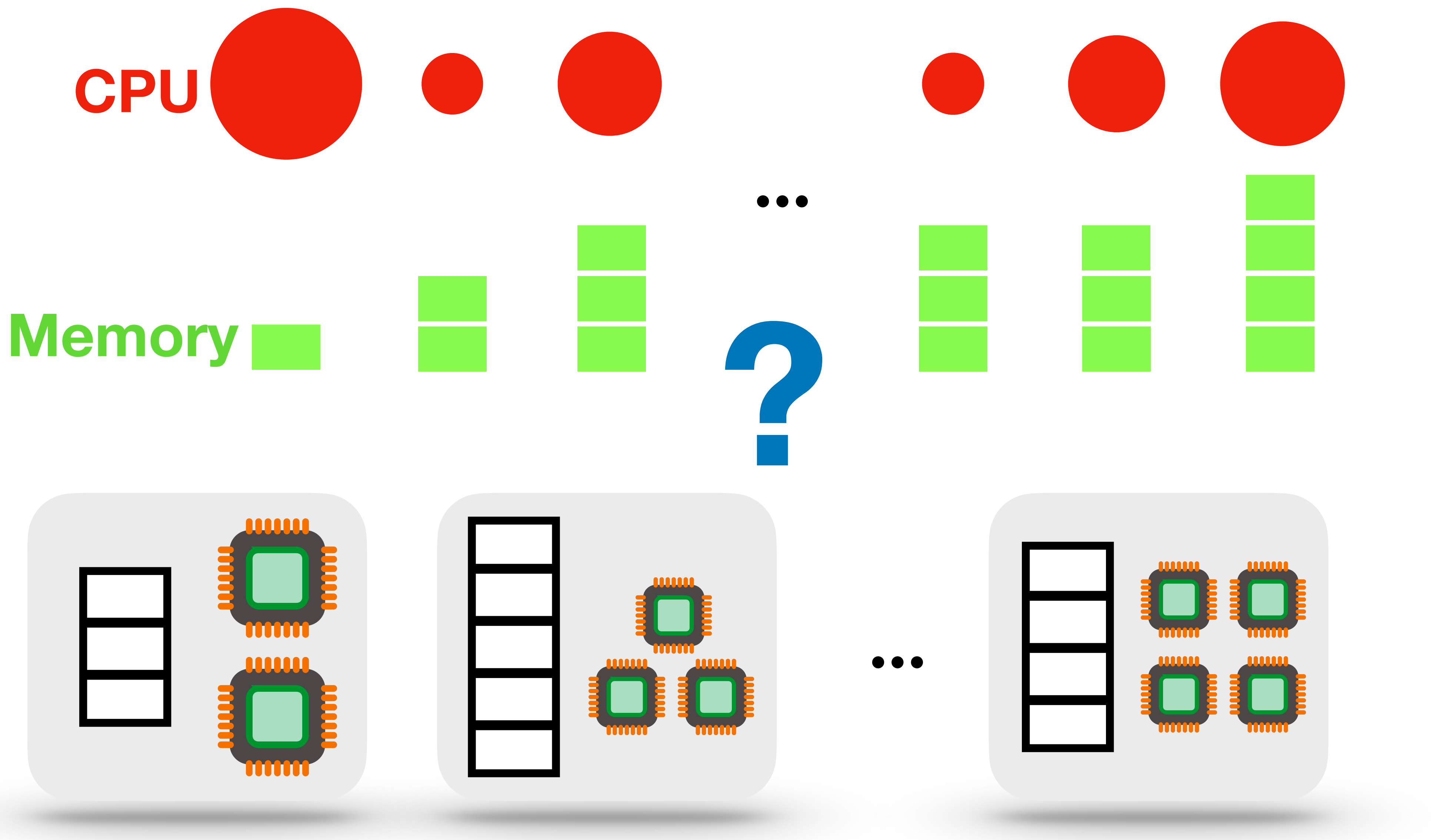
How do you **tailor** the algorithm to **YOUR** instances?



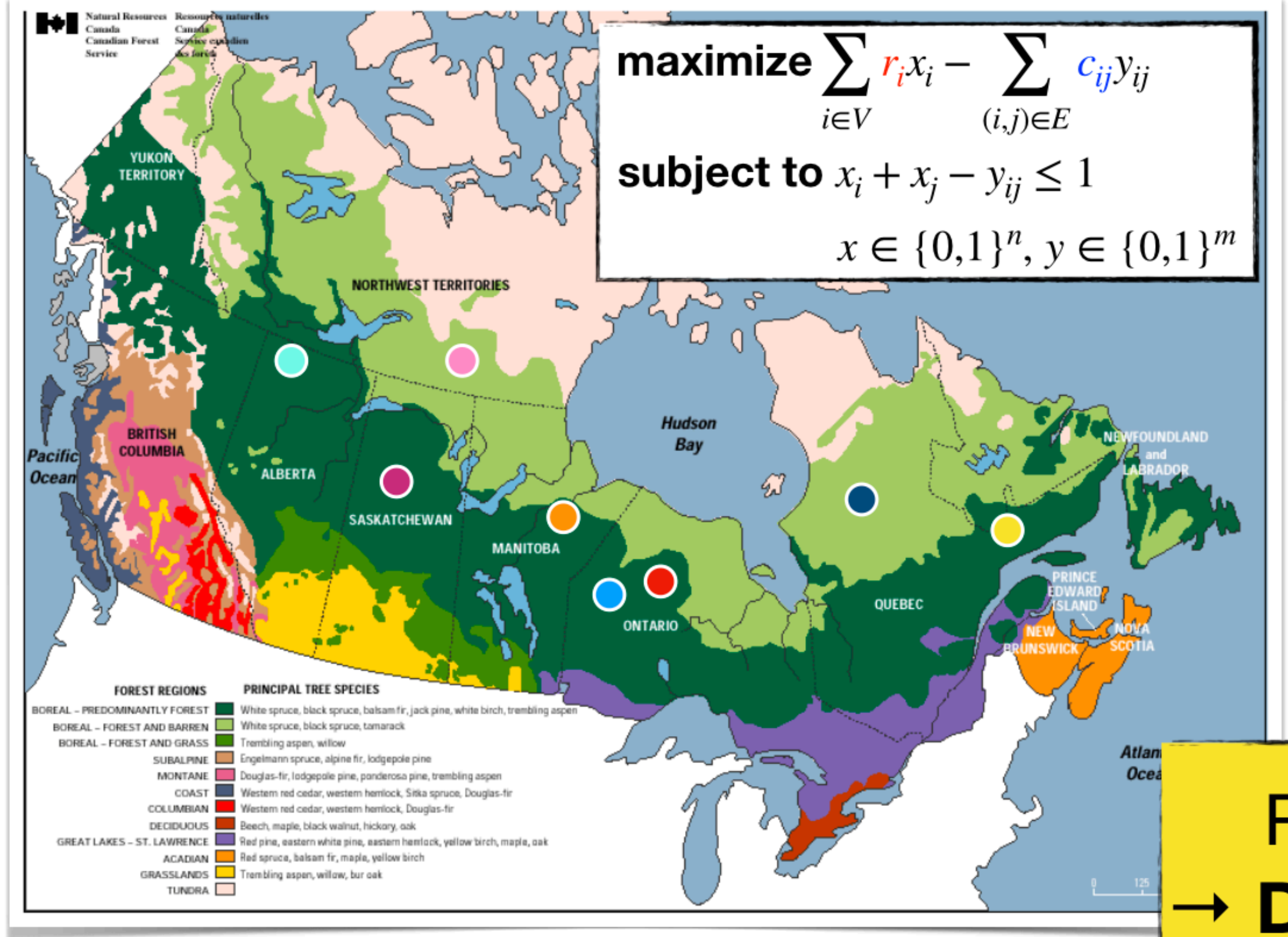
Opportunity

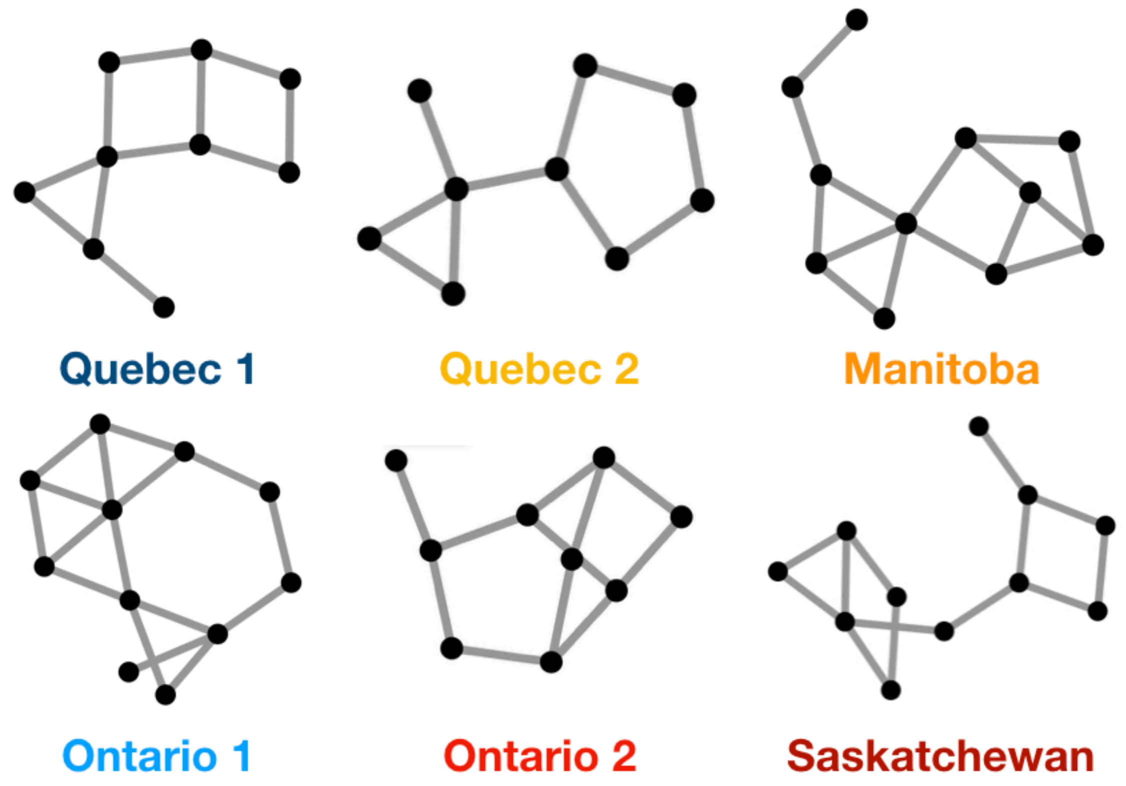
Automatically tailor algorithms to a **family** of **instances**

Data Center Resource Management



Forest Harvesting….

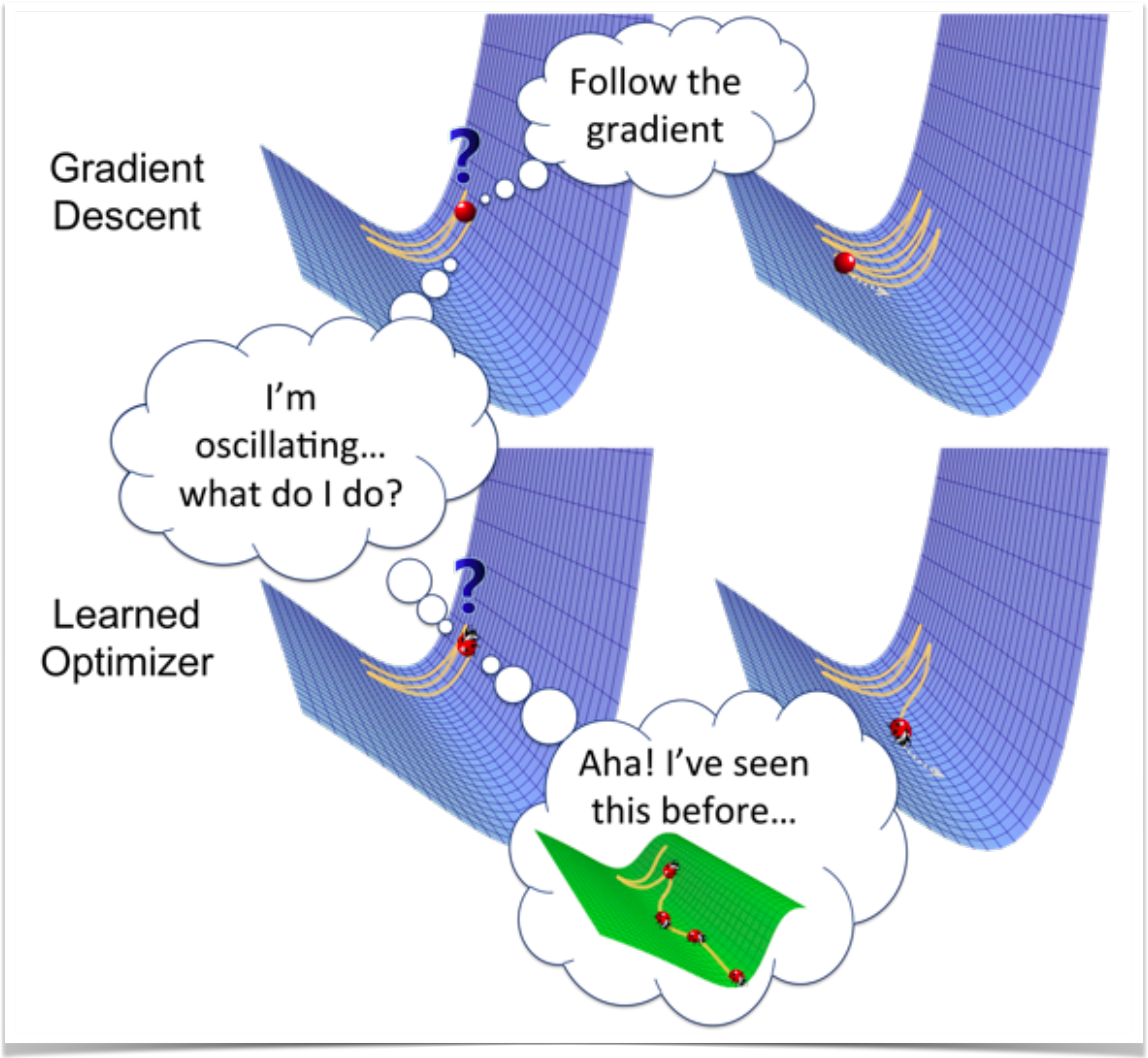




Revenue varies **over time**
→ **Different** instances, **same** problem

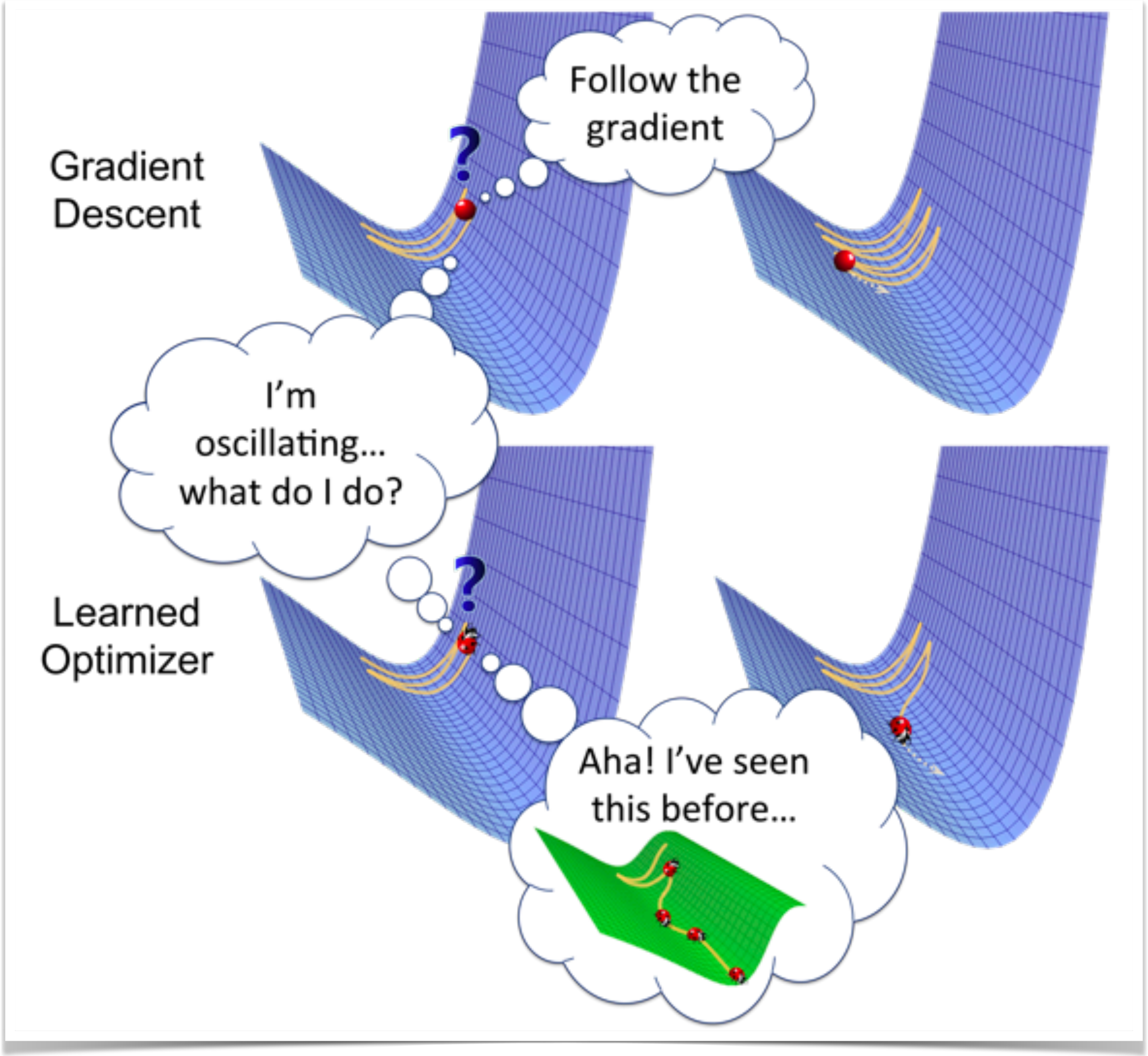
Warm-up: Learning in Gradient Descent

Source: Ke Li, http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl
Li, Ke, and Jitendra Malik. "Learning to optimize." arXiv:1606.01885, 2016 and ICLR, 2017



Warm-up: Learning in Gradient Descent

Source: Ke Li, http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl
Li, Ke, and Jitendra Malik. "Learning to optimize." arXiv:1606.01885, 2016 and ICLR, 2017



Algorithm 1 General structure of optimization algorithms

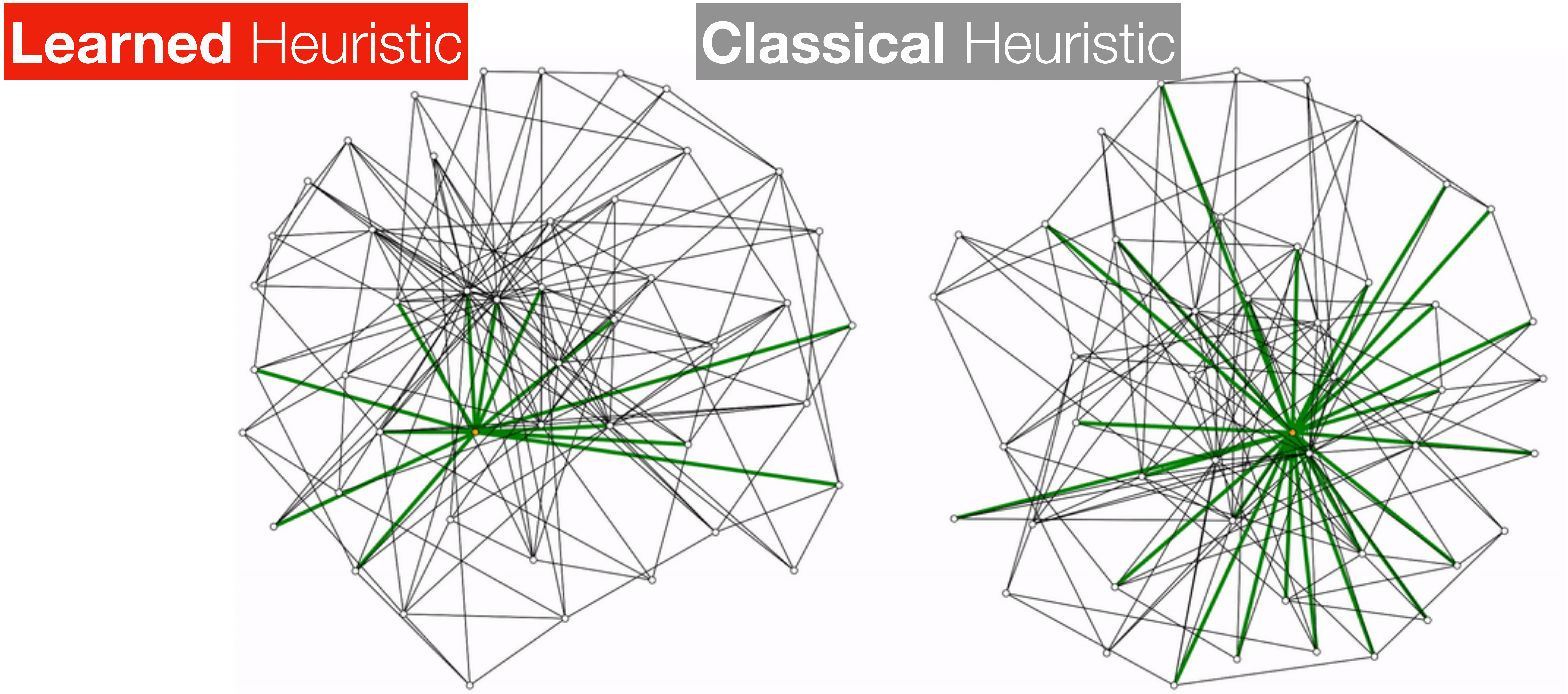
Require: Objective function f
$x^{(0)} \leftarrow$ random point in the domain of f
for $i = 1, 2, \ldots$ **do**
  $\Delta x \leftarrow \boxed{\phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})}$
  if stopping condition is met **then**
    return $x^{(i-1)}$
  end if
  $x^{(i)} \leftarrow x^{(i-1)} + \Delta x$
end for

Gradient Descent	$\phi(\cdot) = -\gamma \nabla f(x^{(i-1)})$
Momentum	$\phi(\cdot) = -\gamma \left(\sum_{j=0}^{i-1} \alpha^{i-1-j} \nabla f(x^{(j)}) \right)$
Learned Algorithm	$\phi(\cdot) =$ Neural Net

Data-Driven Algorithm Design

automatically **discovers**

novel search **strategies**



Minimum Vertex Cover

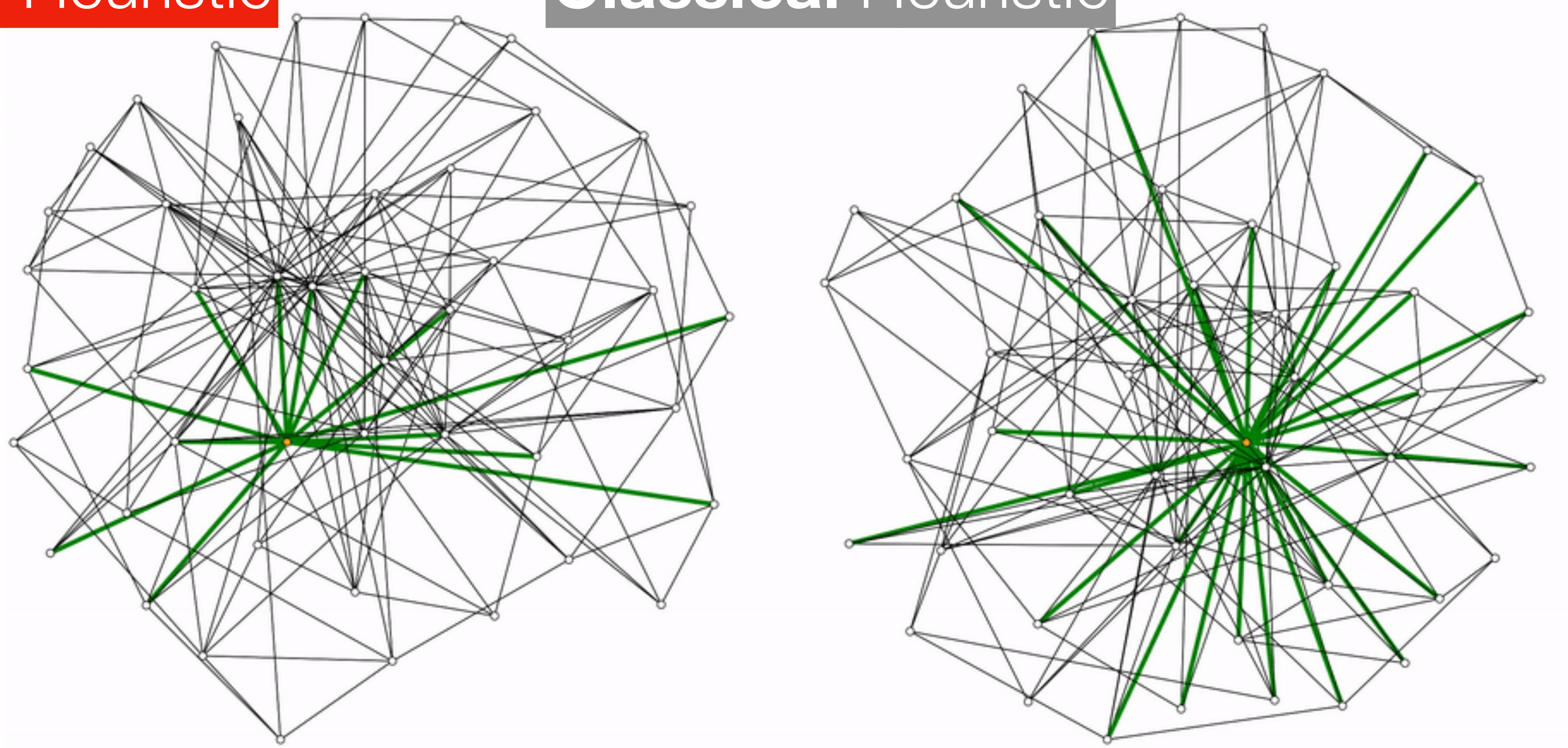
Find **smallest vertex subset** such that each **edge is covered**

Data-Driven Algorithm Design

automatically **discovers** **novel** search **strategies**

Learned Heuristic

Classical Heuristic



Minimum Vertex Cover

Find **smallest vertex subset** such that each **edge is covered**

$$
\begin{aligned}
\max \quad & 5.5x_1 + 2.1x_2 \\
& -x_1 + x_2 \le 2 \\
& 8x_1 + 2x_2 \le 17 \\
& x_1,\ x_2 \ge 0 \\
& x_1,\ x_2 \text{ integer}
\end{aligned}
$$

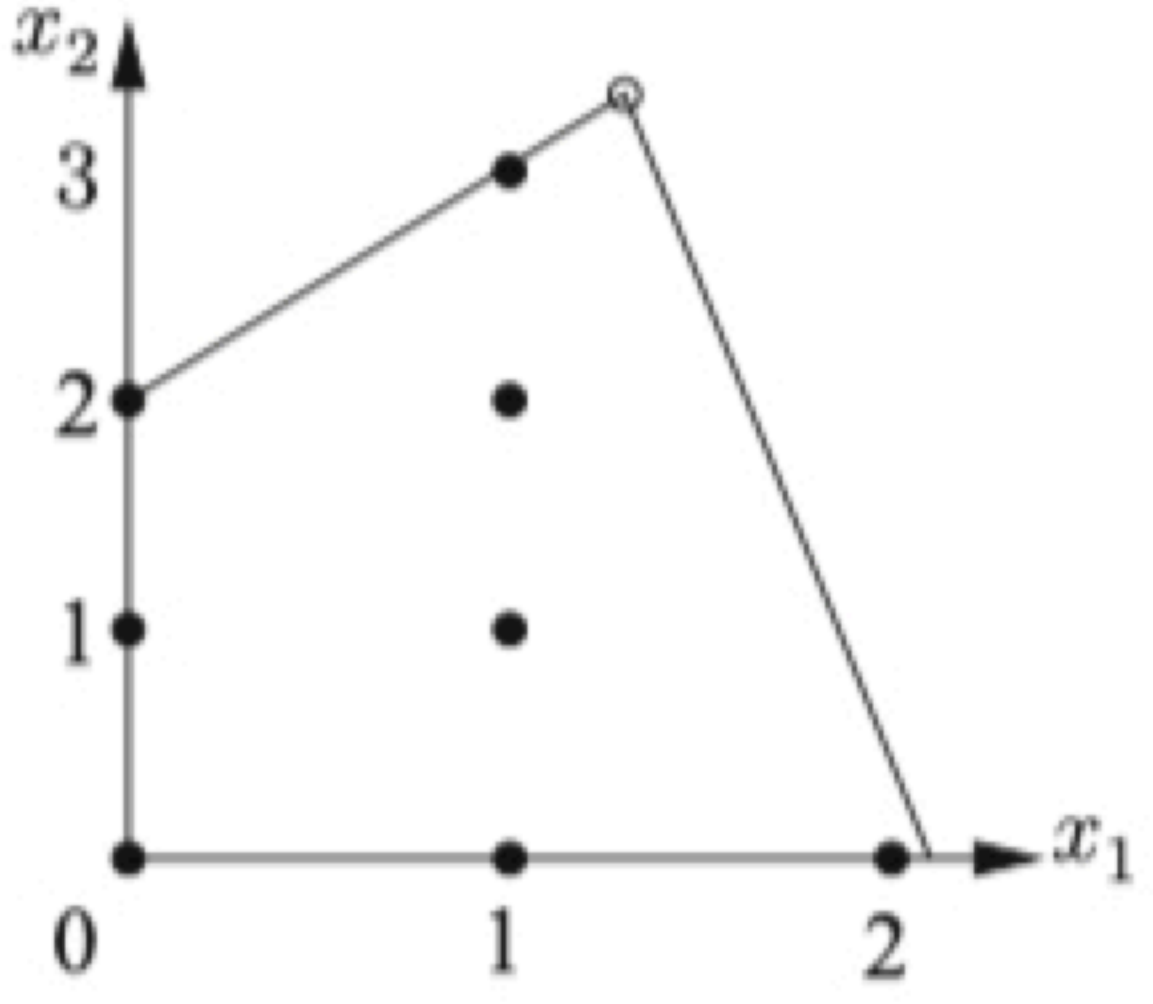


Figure 1.2: A 2-variable integer program

From Integer Programming, 2014.

1

Branch & Bound for Integer Optimization

LP-based $\min_x c^T x$ **s.t.** $Ax \leq b, x \in \{0,1\}^n$

Land & Doig, 1960

Repeat:

1 **Select Node**

2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**

1

Branch & Bound for Integer Optimization

LP-based

$$\min_x c^T x \textbf{ s.t. } Ax \leq b, x \in \{0,1\}^n$$

Land & Doig, 1960

Repeat:

1 Select Node

2 Solve LP Relaxation

3 Prune?

4 Add Cuts

5 Run Heuristics

6 Branch

N_0

1

Branch & Bound for Integer Optimization

LP-based

Land & Doig, 1960

$$\min_{x} c^T x \textbf{ s.t. } Ax \le b, x \in \cancel{\{0,1\}}^n$$

$$[0,1]^n$$

Repeat:

1 **Select Node**

2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**

N_0

Solve LP Relaxation

→ Lower Bound on OPT

1

Branch & Bound for Integer Optimization

LP-based

$$\min_x c^T x \textbf{ s.t. } Ax \leq b, x \in \cancel{\{0,1\}}^n$$

$[0,1]^n$

Land & Doig, 1960

Repeat:

1 **Select Node**

2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**

N_0

Solve LP Relaxation

→ Lower Bound on OPT

worse than best solution?

Prune!

1

Branch & Bound for Integer Optimization

LP-based $\min_x c^T x$ **s.t.** $Ax \leq b, x \in \{0,1\}^n$

Land & Doig, 1960

Repeat:

1 **Select Node**

2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**

N_0

Add Cuts:
Tightening Constraints

1

Branch & Bound for Integer Optimization

LP-based

$$\min_x c^T x \text{ s.t. } Ax \leq b, x \in \{0,1\}^n$$

Land & Doig, 1960

Repeat:

1 Select Node

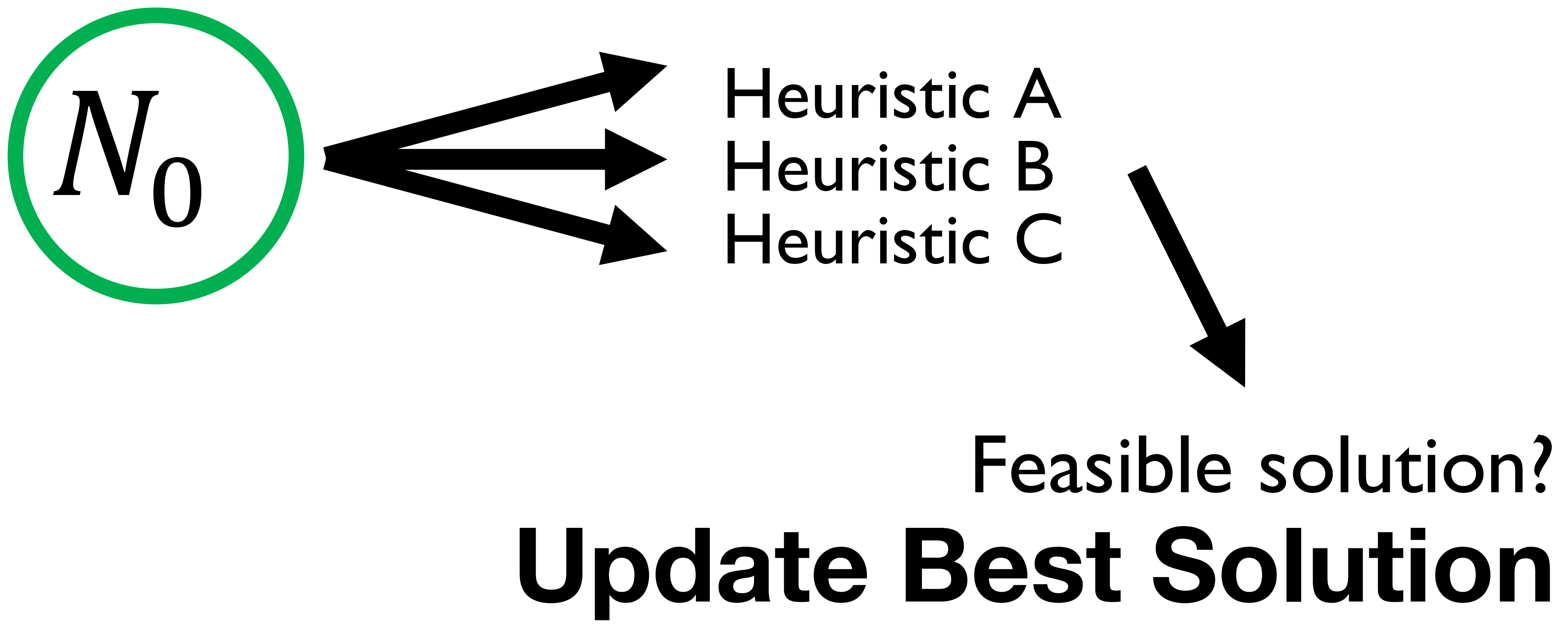
2 Solve LP Relaxation

3 Prune?

4 Add Cuts

5 Run Heuristics

6 Branch



1

Branch & Bound for Integer Optimization

LP-based $\min_x c^T x$ **s.t.** $Ax \le b, x \in \{0,1\}^n$

Land & Doig, 1960

Repeat:

1 **Select Node**

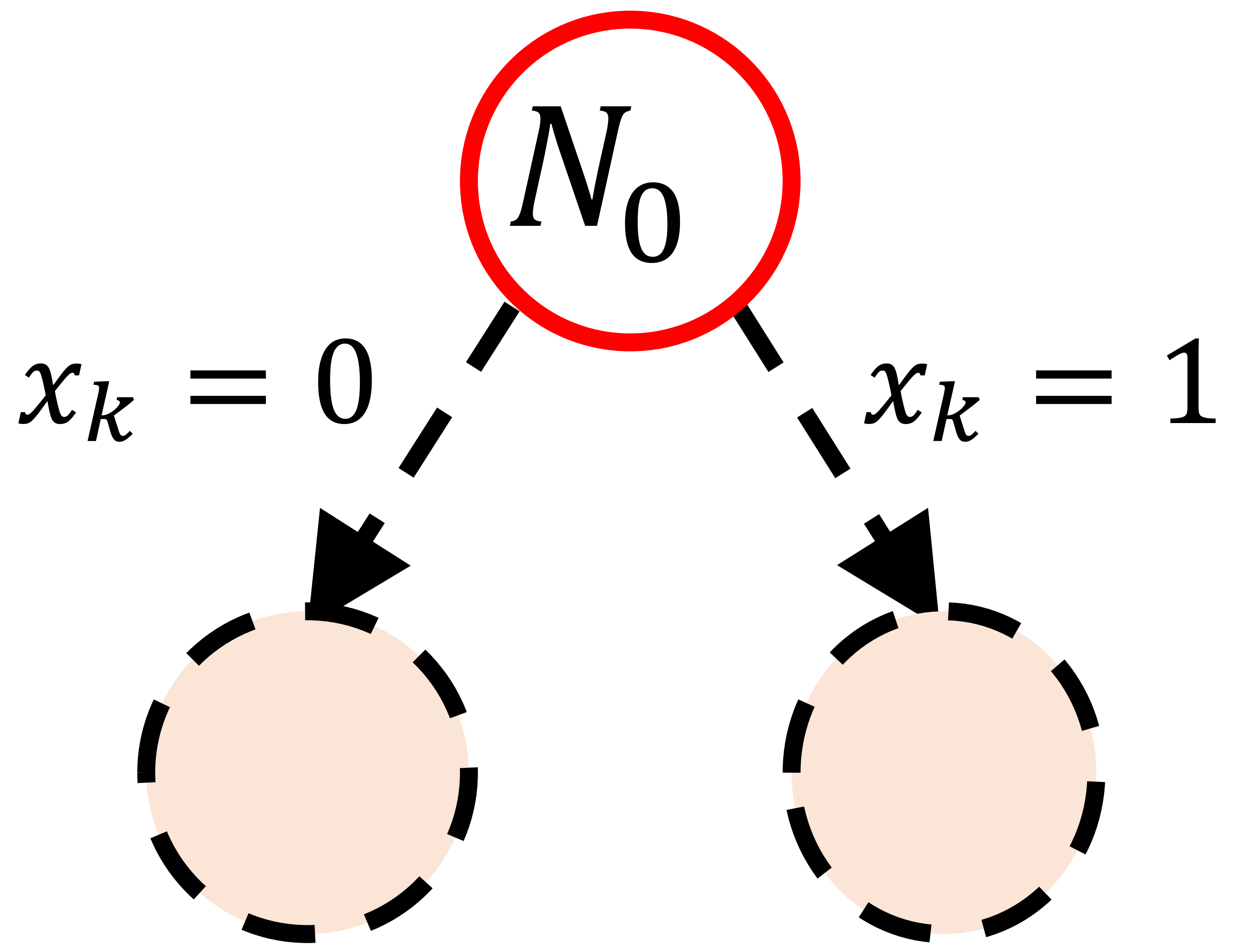
2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**



x_1?
x_2?
...
x_n?

1

Branch & Bound for Integer Optimization

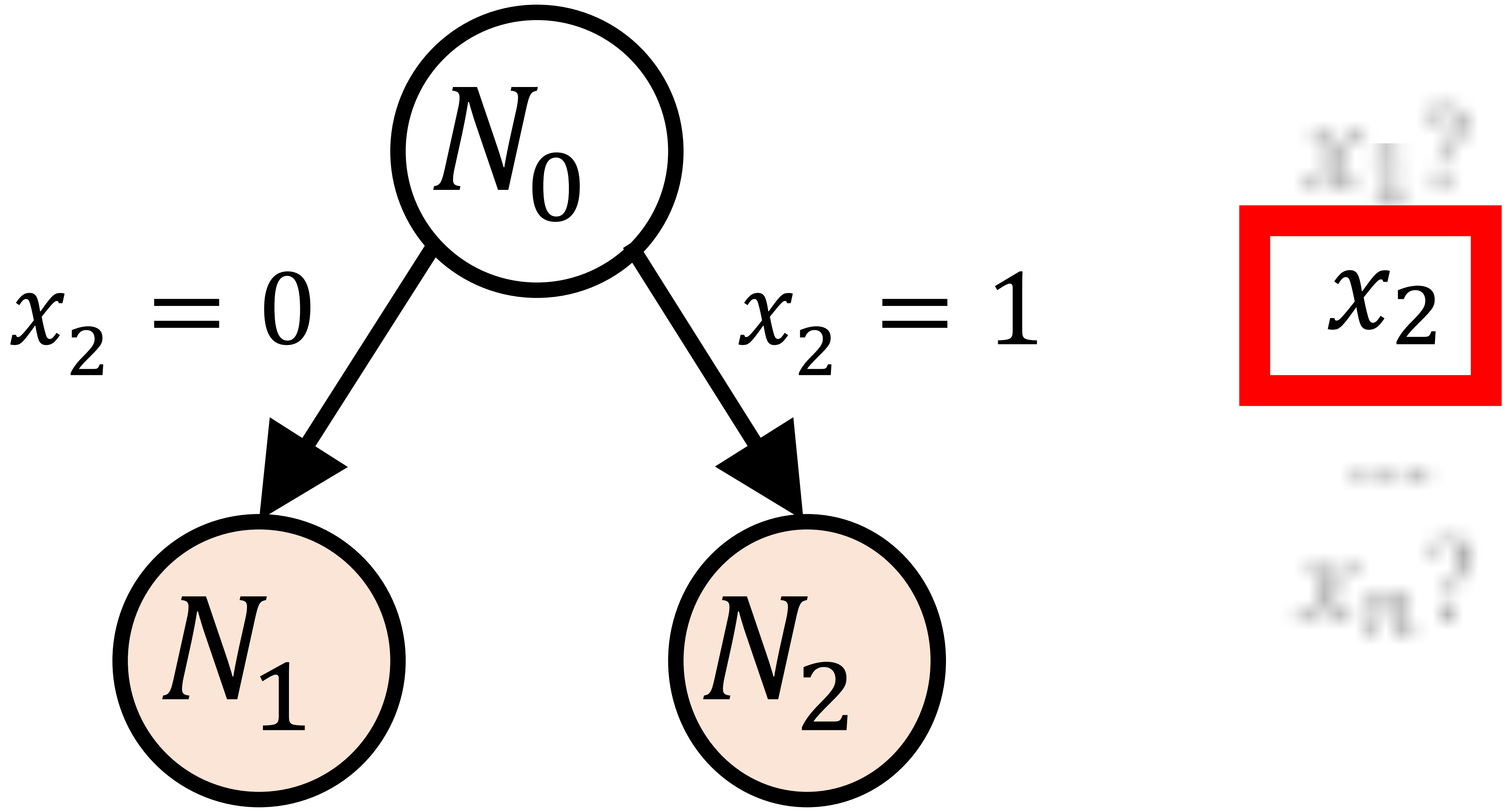
LP-based

$$\min_x c^T x \textbf{ s.t. } Ax \le b, x \in \{0,1\}^n$$

Land & Doig, 1960

Repeat:

1 Select Node
2 Solve LP Relaxation
3 Prune?
4 Add Cuts
5 Run Heuristics
6 Branch



1

Branch & Bound for Integer Optimization

LP-based $\min_x c^T x$ **s.t.** $Ax \leq b, x \in \{0,1\}^n$

Land & Doig, 1960

Repeat:

1 **Select Node**

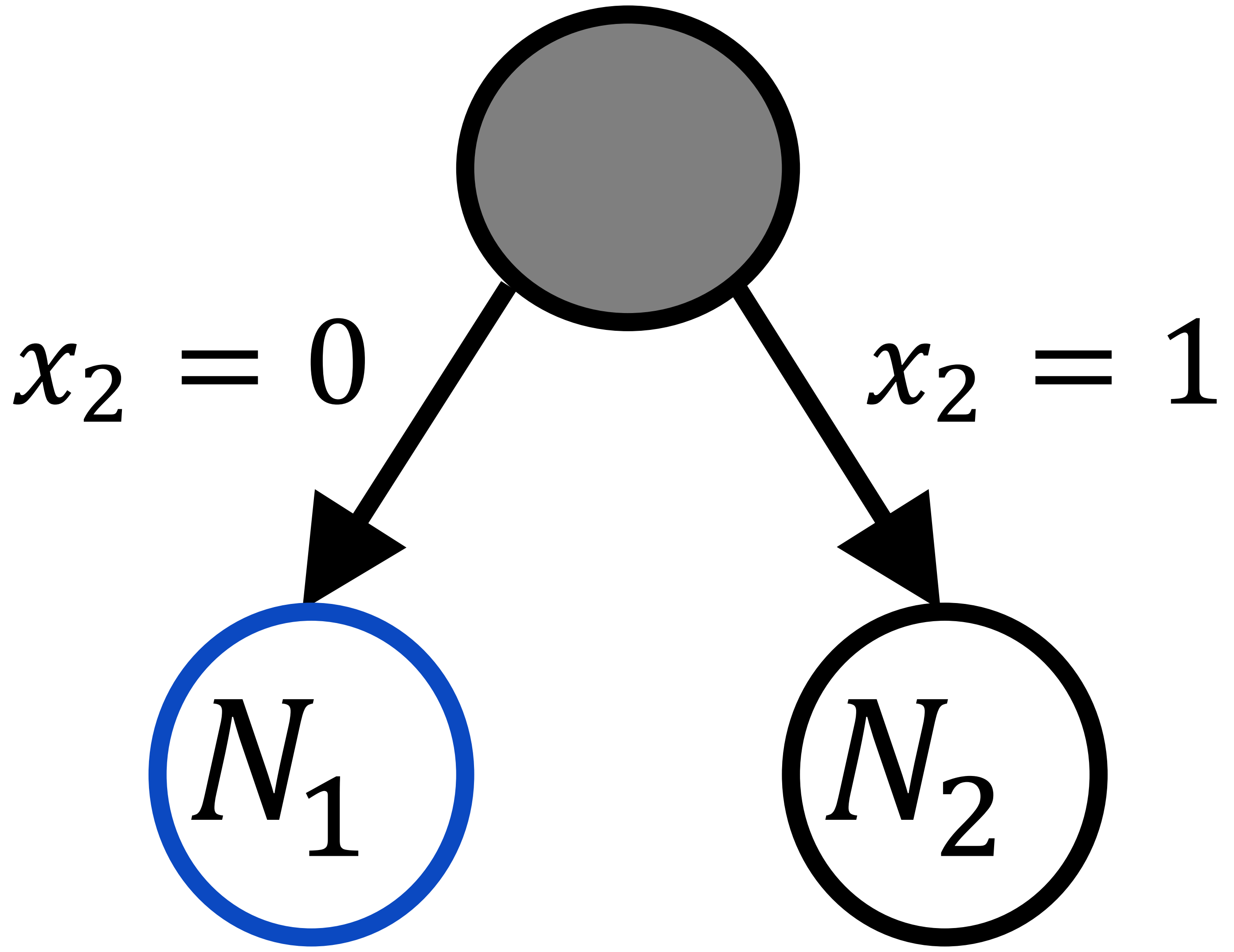
2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**



1

Branch & Bound for Integer Optimization

Land & Doig, 1960

LP-based $\min_x c^T x$ **s.t.** $Ax \leq b, x \in \{0,1\}^n$

Repeat:

1 **Select Node**

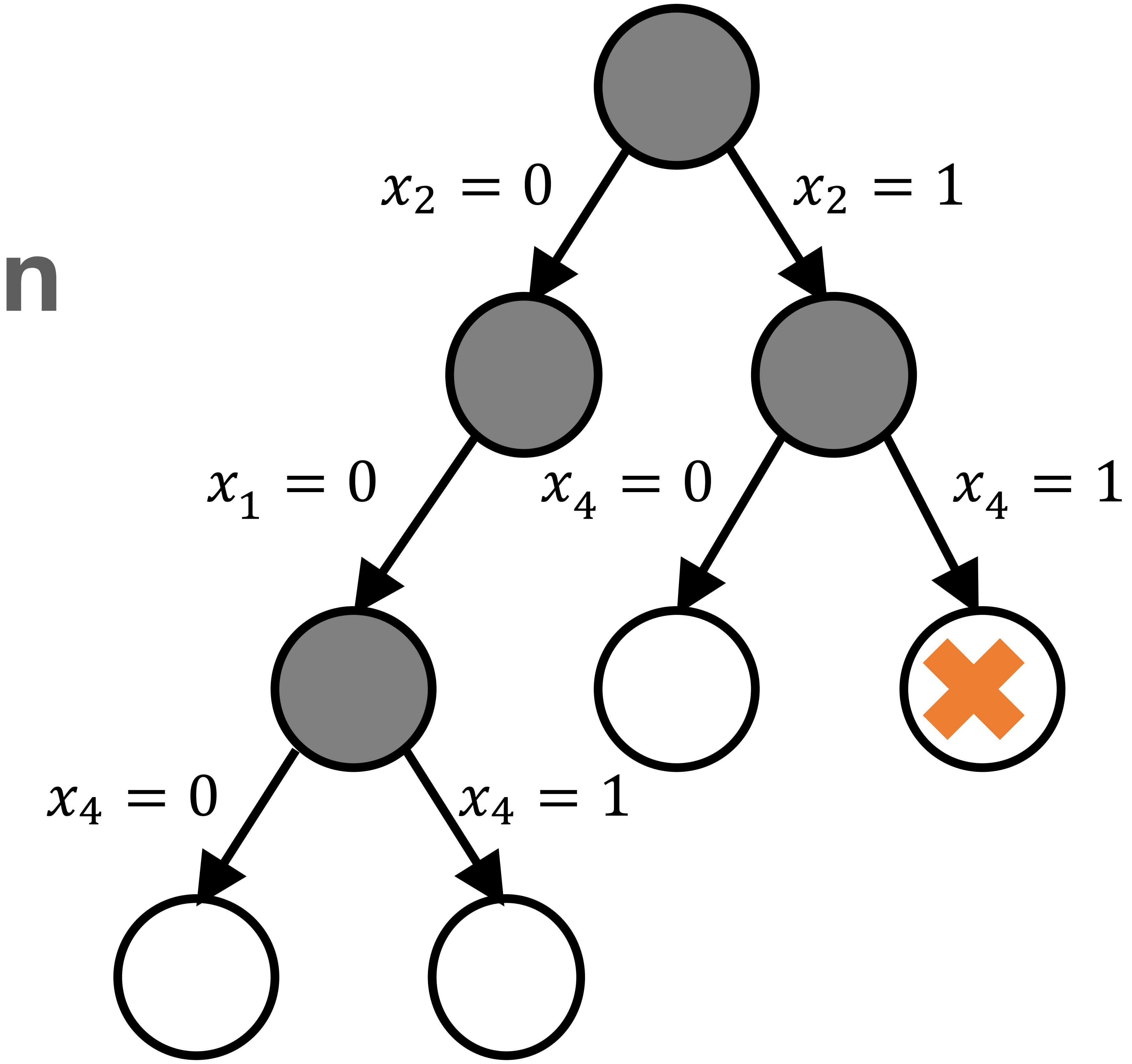
2 **Solve LP Relaxation**

3 **Prune?**

4 **Add Cuts**

5 **Run Heuristics**

6 **Branch**



1

Empirical algorithmics

How do we evaluate and compare algorithms?

162 Chapter 4 *Empirical Analysis of SLS Algorithms*

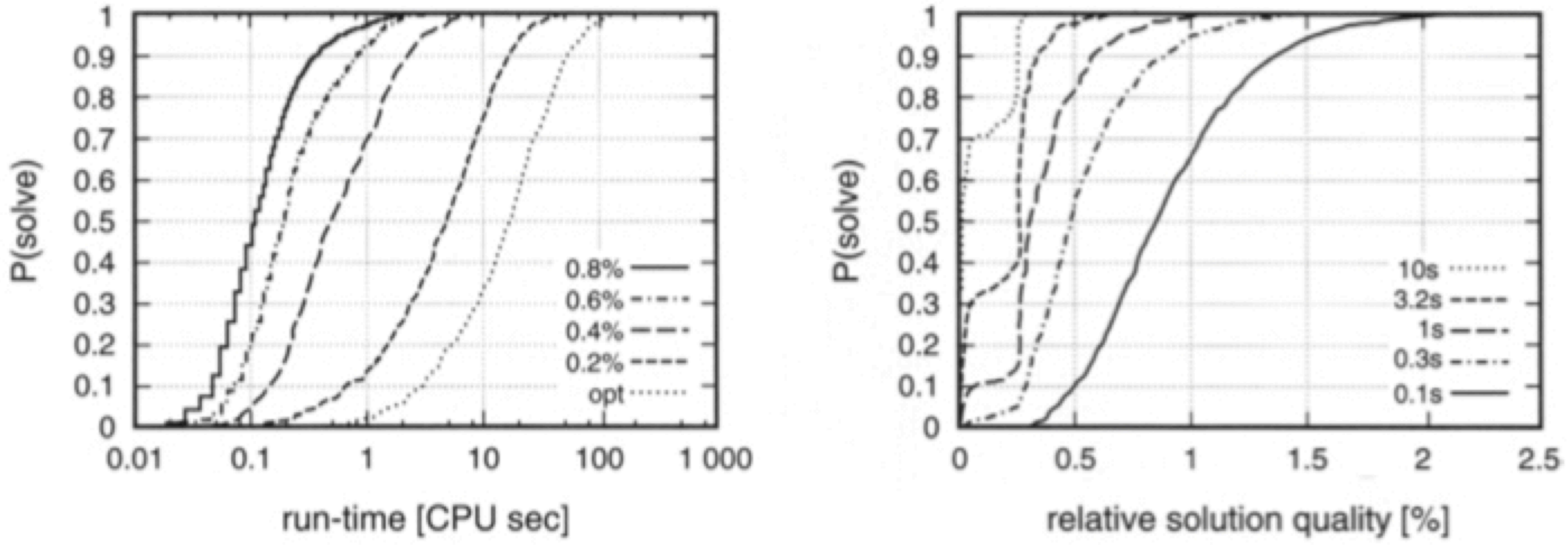


Figure 4.2 *Left:* Qualified RTDs for the bivariate RTD from Figure 4.1. *Right:* SQDs for the same RTD .

2

Towards Tailored Algorithms

IBM Knowledge Center

- Managing sets of parameters
- Parameter names
- Correspondence of parameters between APIs
- Saving parameter settings to a file in the C API
- − **Topical list of parameters**
  - Barrier
  - Benders algorithm
  - Distributed MIP
  - − **MIP**
    - MIP general
    - **MIP strategies**
    - MIP cuts
    - MIP tolerances
    - MIP limits

Here are links to parameters controlling MIP strategies.

algorithm for initial MIP relaxation

Benders strategy

MIP subproblem algorithm

MIP variable selection strategy

MIP strategy best bound interval

MIP branching direction

backtracking tolerance

MIP dive strategy

MIP heuristic effort

CPLEX Documentat

Towards Tailored Algorithms

MIP variable selection strategy

Value	Symbol	Meaning
-1	CPX_VARSEL_MININFEAS	Branch on variable with minimum infeasibility
0	CPX_VARSEL_DEFAULT	Automatic: let CPLEX choose variable to branch on; **default**
1	CPX_VARSEL_MAXINFEAS	Branch on variable with maximum infeasibility
2	CPX_VARSEL_PSEUDO	Branch based on pseudo costs
3	CPX_VARSEL_STRONG	Strong branching
4	CPX_VARSEL_PSEUDOREDUCED	Branch based on pseudo reduced costs

MIP heuristic frequency

Value	Meaning
-1	None
0	Automatic: let CPLEX choose; **default**
Any positive integer	Apply the periodic heuristic at this frequency

CPLEX Documentat 3

Towards Tailored Algorithms

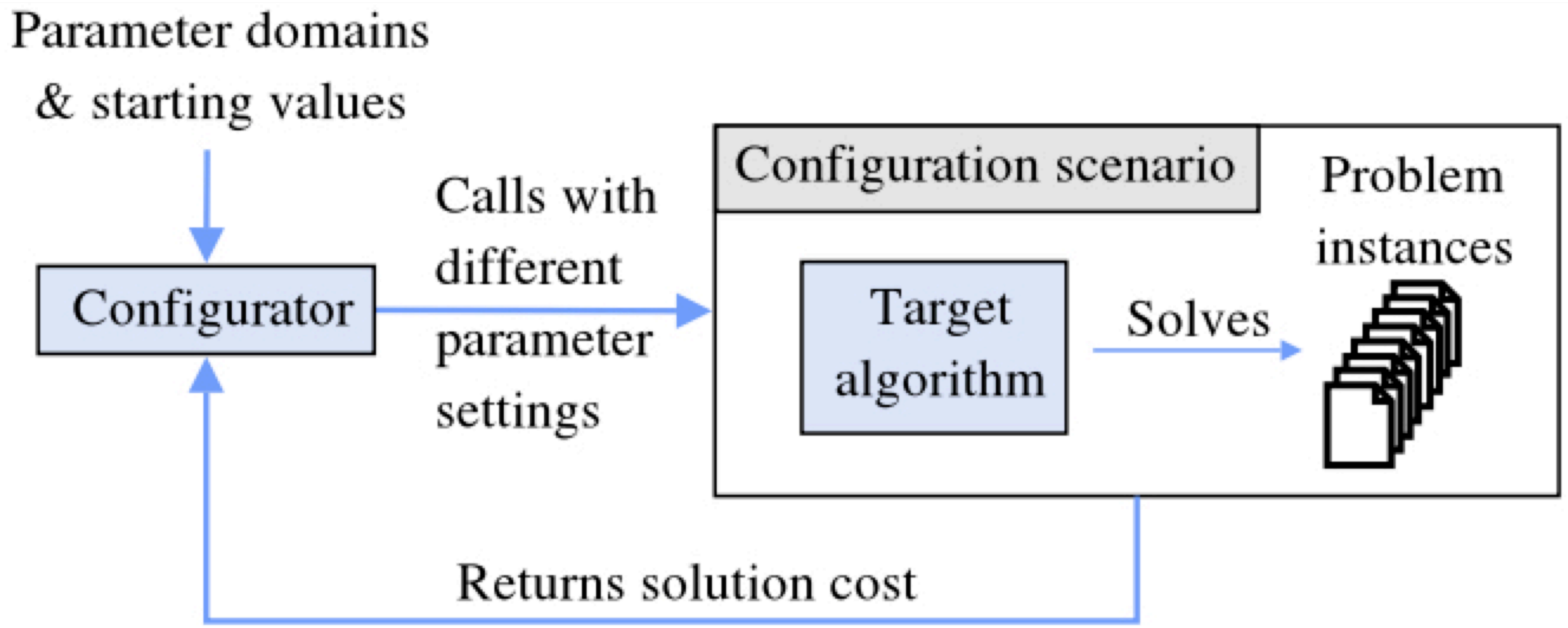


Fig. 1. A configuration procedure (short: configurator) executes the target algorithm with specified parameter settings on one or more problem instances, observes algorithm performance, and uses this information to decide which subsequent target algorithm runs to perform. A configuration scenario includes the target algorithm to be configured and a collection of instances.

Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown. "Automated configuration of mixed integer programming solvers." CPAIOR, 2010.

3

Towards Tailored Algorithms

See IJCAI-20 Tutorial: https://www.automl.org/tutorial_ac_ijcai20/

Automated Algorithm Configuration

- ParamILS [Hutter et al., JAIR 2009], SMAC [Hutter et al., LION 2011]
- Key Idea: search over parameter configurations
  - Stochastic Local Search or Bayesian Optimization

- Great for algorithms with many parameters
  - 2-52x speedups for CPLEX on some problem distributions [Hutter et al., CPAIOR 2010]

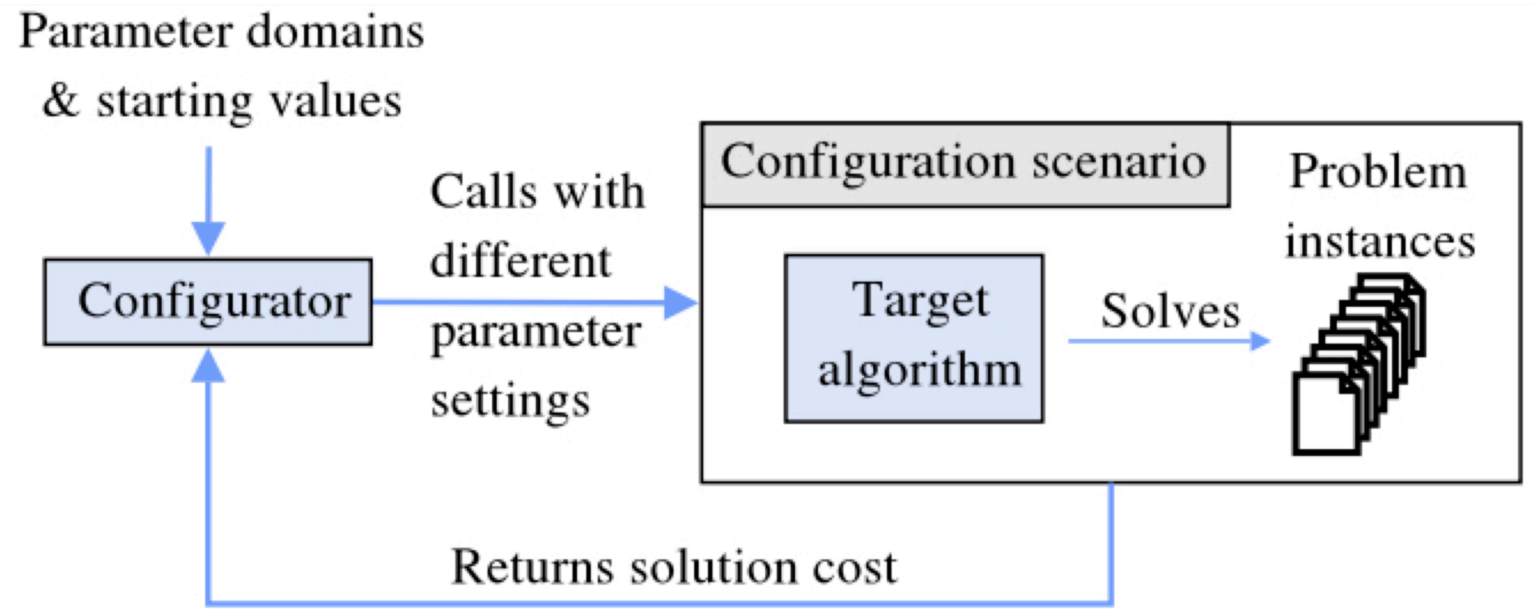


Fig. 1. A configuration procedure (short: configurator) executes the target algorithm with specified parameter settings on one or more problem instances, observes algorithm performance, and uses this information to decide which subsequent target algorithm runs to perform. A configuration scenario includes the target algorithm to be configured and a collection of instances.

Limitations

- Operates at the instance-level, not the **algorithm iteration-level**
- Assumes human-designed parameter space is rich enough

3

Learning in Exact Solvers

Algorithm: LP-based Branch-and-Bound

Input: a MIP $\min\{c^T x \mid Ax \leq b, x \in \mathbb{R}^n, x_j \in \mathbb{Z}\ \forall j \in I\}$

Output: an optimal solution x^*, $z^* := c^T x^*$

1 Initialize: Queue of sub-problems (nodes) $\mathcal{L} := \{N_0\}$, Best value $z^* := \infty$, Best solution $x^* := \emptyset$

2 Terminate? If $\mathcal{L} = \emptyset$, **return** x^*

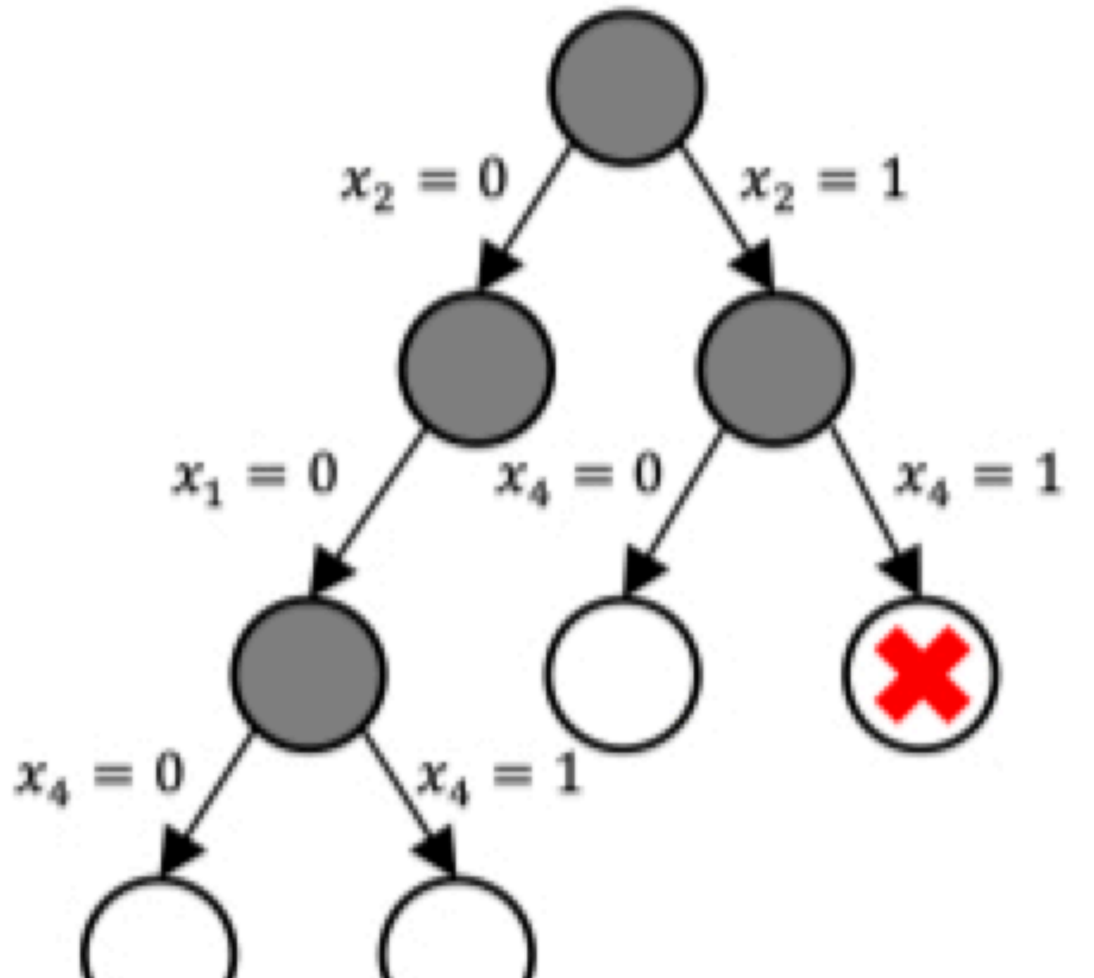
3 Select Node **[what selection rule?]**: Choose a node N_i to process from $\mathcal{L}$

4 Evaluate & Prune: Solve the LP relaxation of N_i and prune node if applicable.

5 Add Cuts **[which cuts to add?]**: new constraints that tighten the formulation.

6 Run Heuristics **[which heuristics to run?]**: try to find a better solution.

7 Select Branching Variable **[what selection rule?]**: Choose a variable that has fractional value in the LP solution of N_i. Create two new subproblems N_{i1} and N_{i2}. Go to line 2.



Learning in Exact Solvers

Algorithm: LP-based Branch-and-Bound

Input: a MIP $\min\{c^T x \mid Ax \leq b, x \in \mathbb{R}^n, x_j \in \mathbb{Z}\ \forall j \in I\}$

Output: an optimal solution $x^*, z^* := c^T x^*$

1 Initialize: Queue of sub-problems (nodes) $\mathcal{L} := \{N_0\}$, Best value $z^* := \infty$, Best solution $x^* := \emptyset$

2 Terminate? If $\mathcal{L} = \emptyset$, **return** x^*

3 Select Node **[what selection rule?]**: Choose a node N_i to process from $\mathcal{L}$

4 Evaluate & Prune: Solve the LP relaxation of N_i and prune node if applicable.

5 Add Cuts **[which cuts to add?]**: new constraints that tighten the formulation.

6 Run Heuristics **[which heuristics to run?]**: try to find a better solution.

7 Select Branching Variable **[what selection rule?]**: Choose a variable that has fractional value in the LP solution of N_i. Create two new subproblems N_{i1} and N_{i2}. Go to line 2.

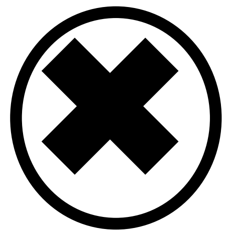
Task	Issue	Current Approach
Select Branching Variable	what selection rule?	single **hand-designed** ranking metric
Add Cuts	which cuts to add?	**hand-designed** ranking formula
Run Heuristics	which heuristics to run?	run each every k nodes (**fixed parameter** k)
Select Node	what selection rule?	best-first

4

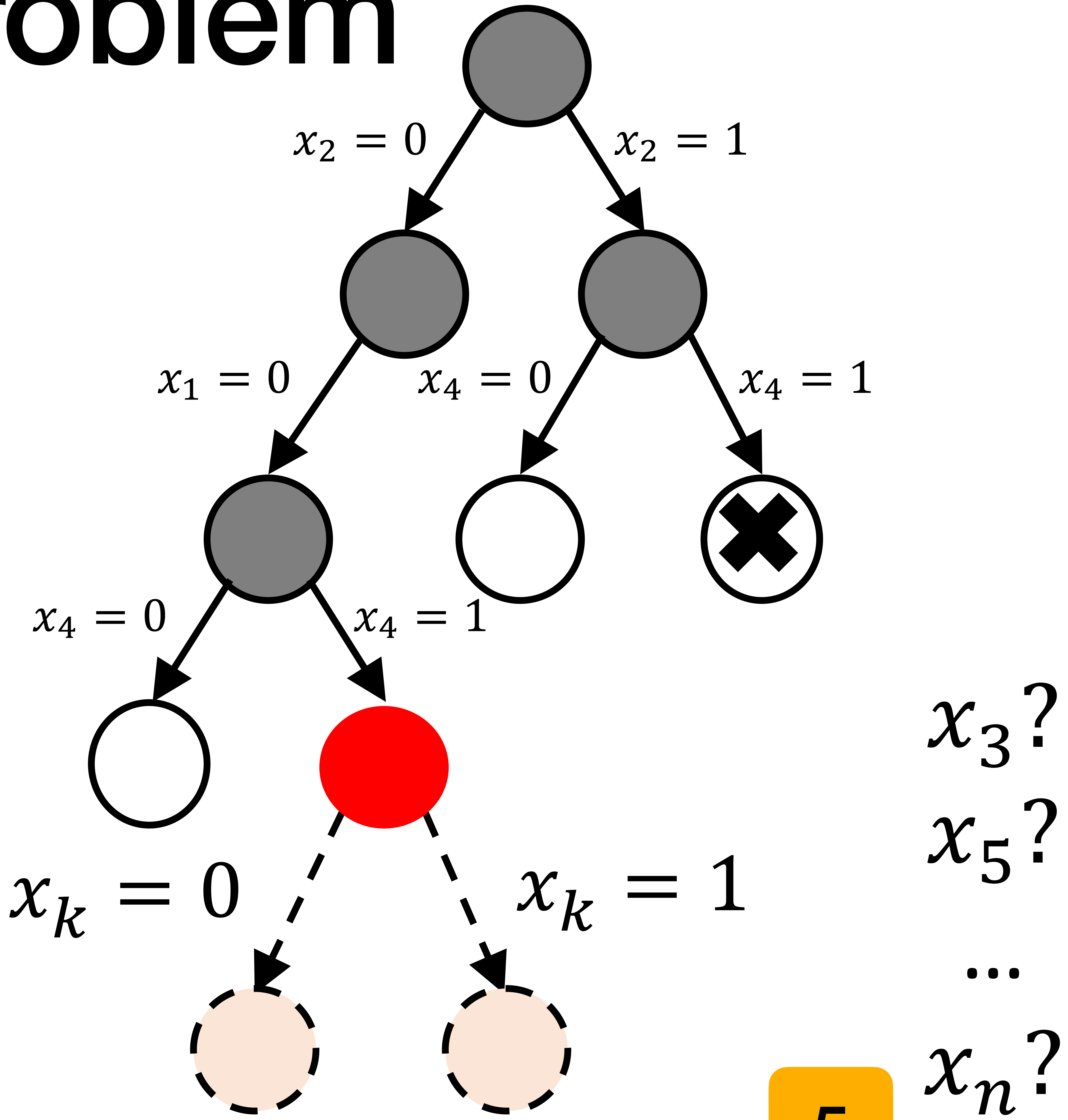
The Branching Problem

A **key step** of Branch-and-Bound

Ideally, select variables that lead to small sub-tree ↔ many infeasible nodes



Strong Branching (SB) achieves that, but is extremely costly



5

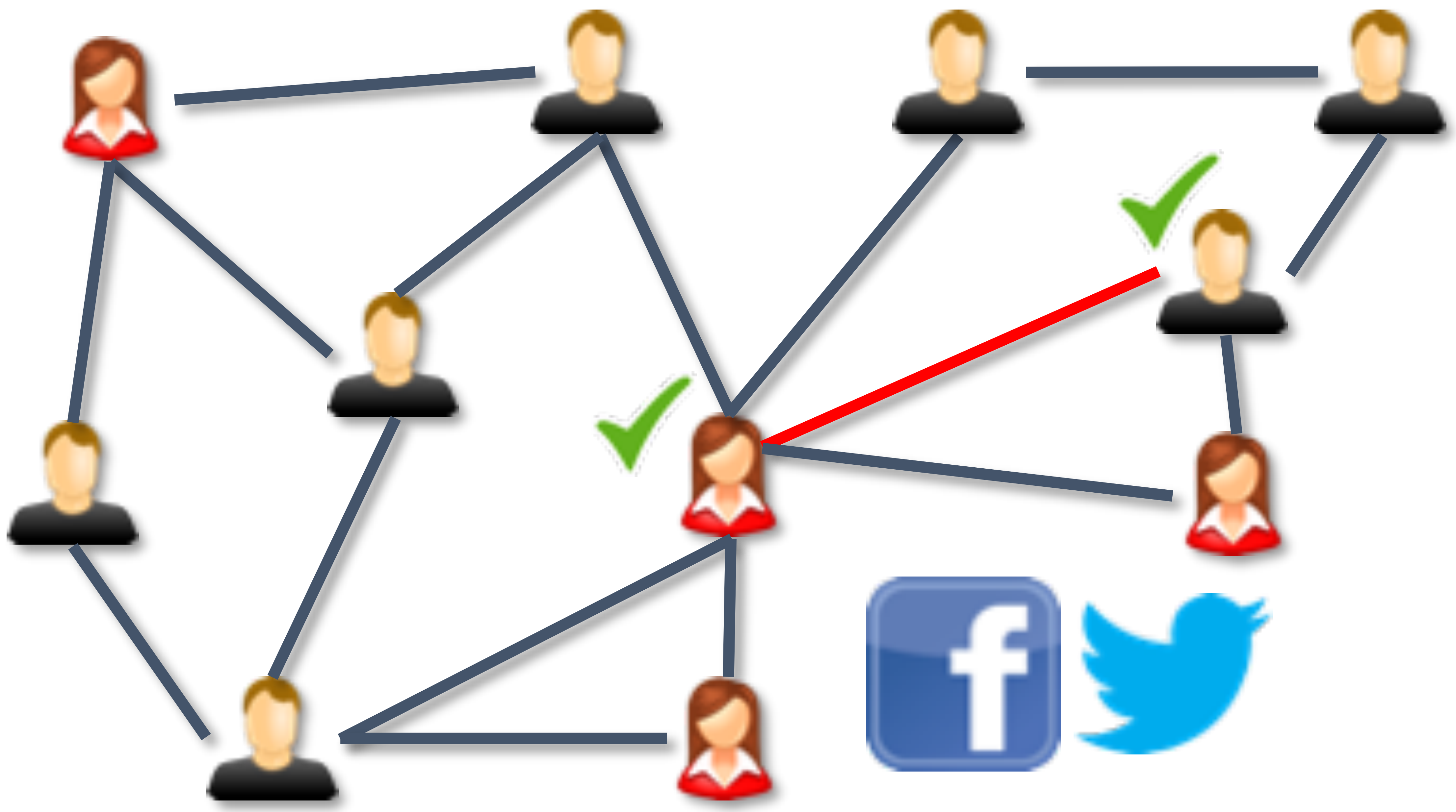
Greedy Graph Optimization

Minimum Vertex Cover

Find smallest vertex subset such that each edge is covered

2-Approximation:

Greedily add vertices of edge with **max degree sum**



6

Greedy Graph Optimization

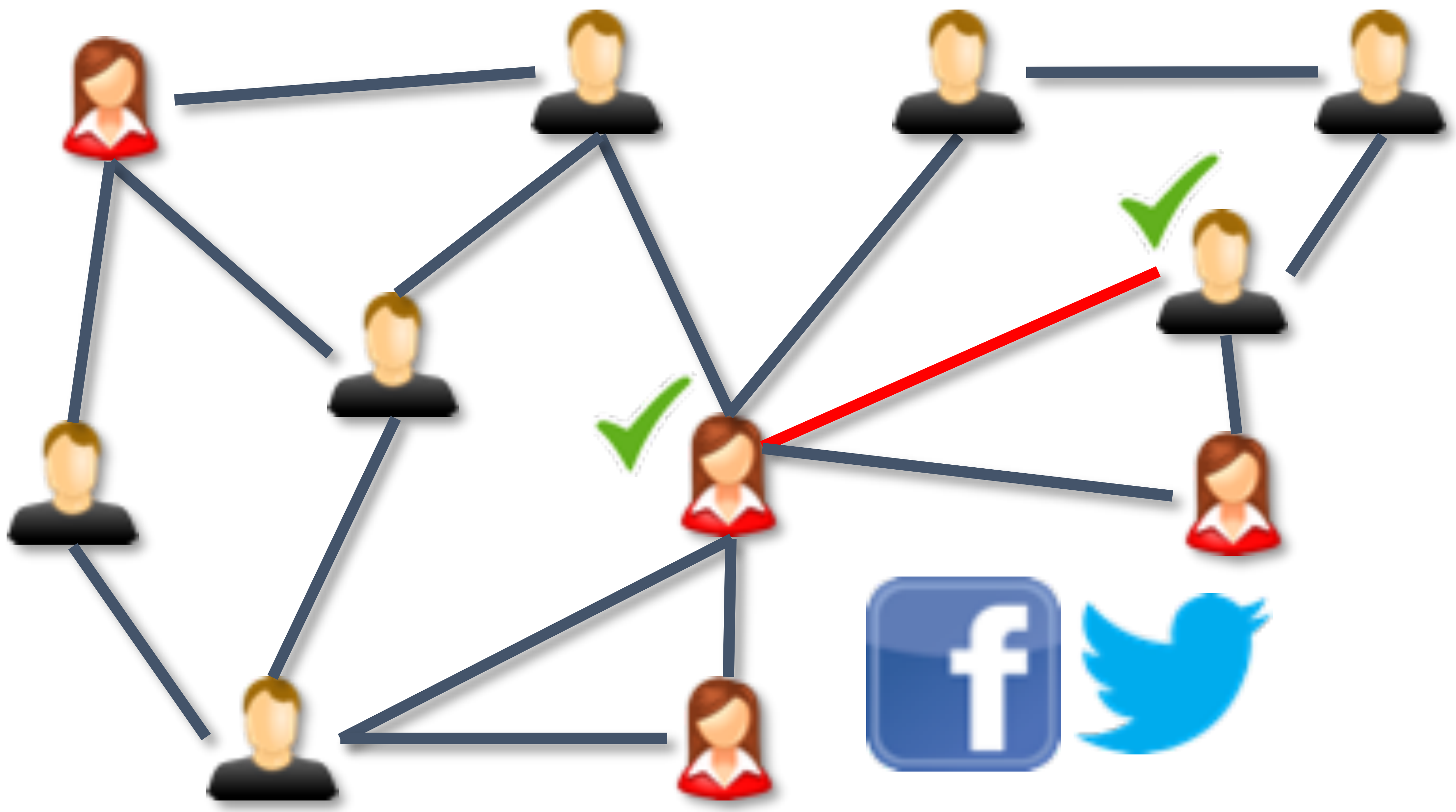
Minimum Vertex Cover

Find smallest vertex subset such that each edge is covered

Learning Greedy Graph Heuristics
[Dai*, Khalil*, Zhang, Dilkina, Song, 2017]

Given: graph problem, family of graphs

Learn: a **scoring function** to **guide** a **greedy** algorithm



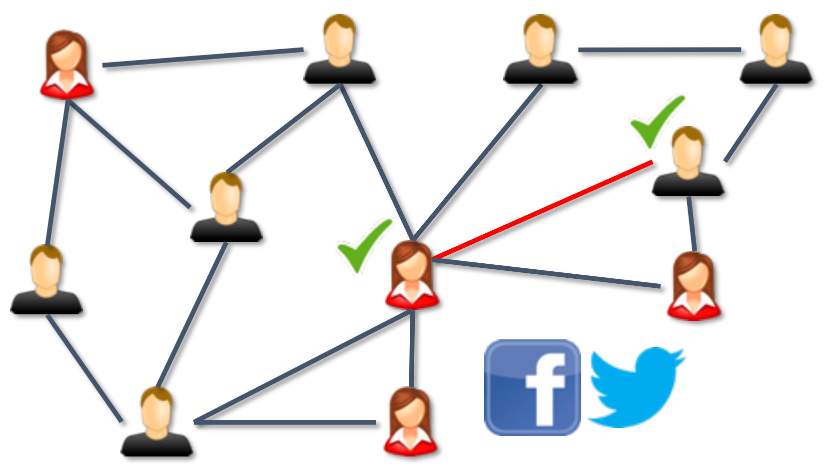
6

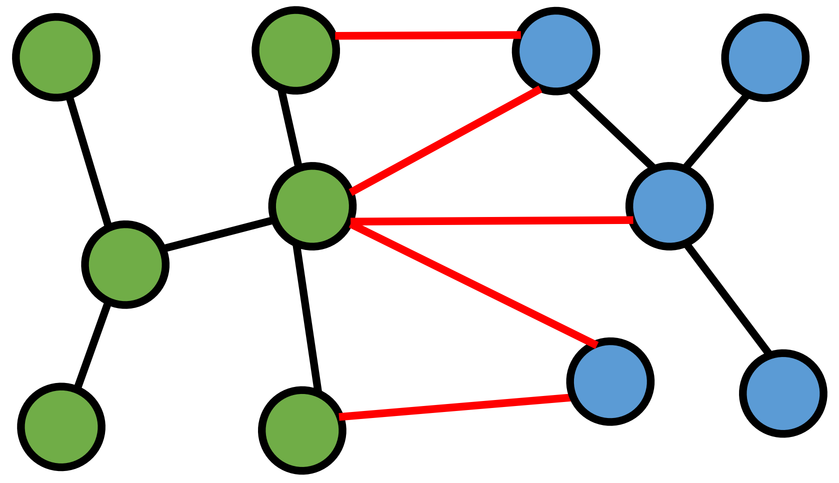
Learning Greedy Heuristics

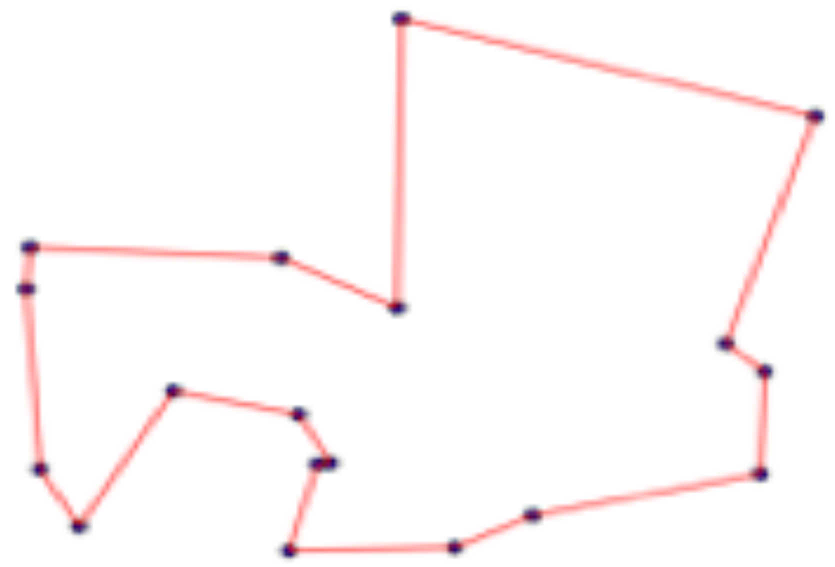
Given: graph problem, family of graphs

Learn: a **scoring function** to **guide** a **greedy** algorithm

Problem	**Minimum Vertex Cover**	**Maximum Cut**	**Traveling Salesman Problem**
Domain	Social network snapshots	Spin glass models	Package delivery
Greedy operation	Insert nodes into cover	Insert nodes into subset	Insert nodes into sub-tour





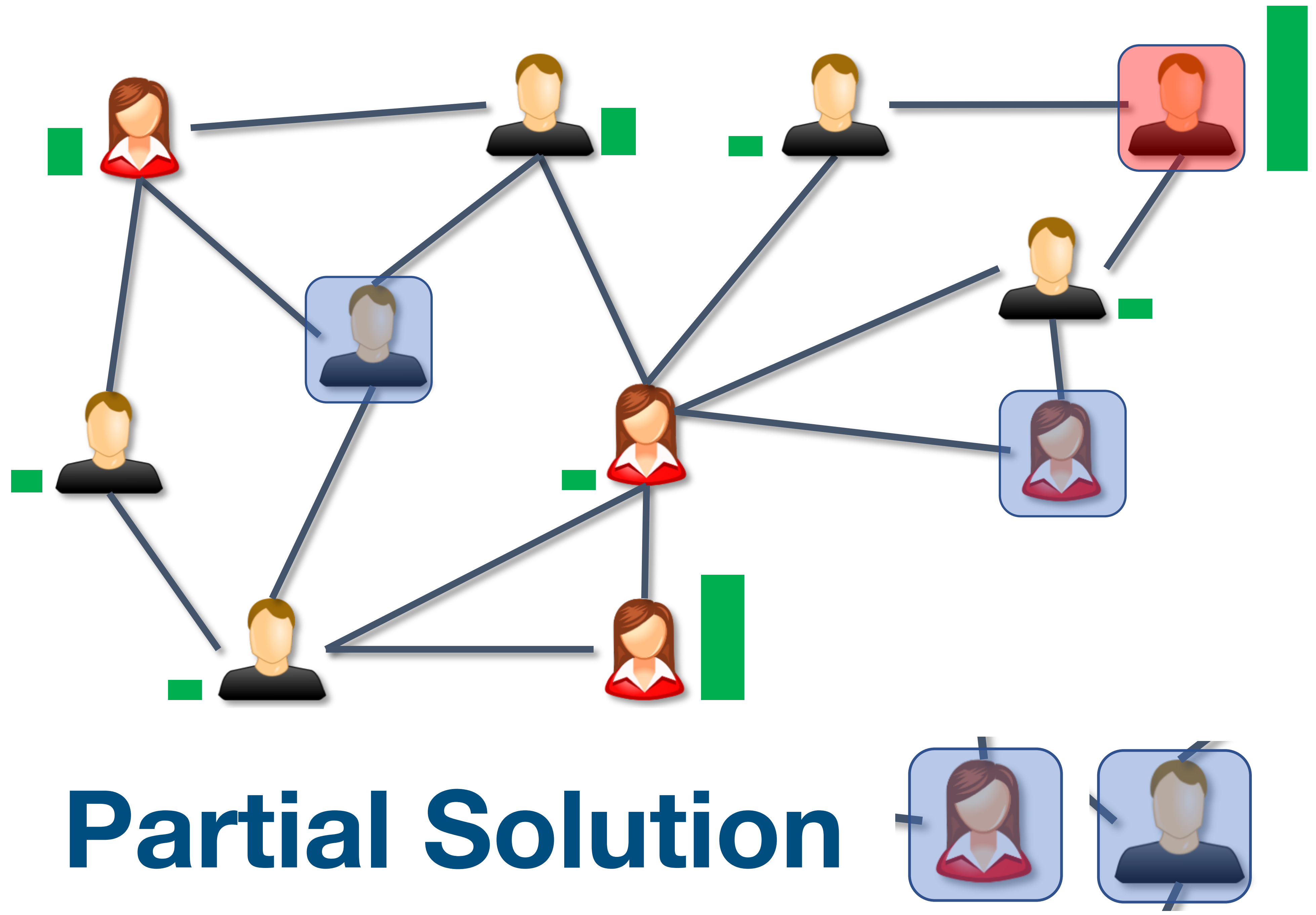


Reinforcement Learning

Greedy Algorithm		Reinforcement Learning
Partial solution	≡	State
Scoring function	≡	Q-function
Select best node	≡	Greedy Policy

Repeat until all edges are covered:

1. Compute node scores
2. Select best node w.r.t. score
3. Add best node to partial sol.



6

Combinatorial Optimization & Reasoning with Graph Neural Networks

IJCAI 2021 MONTREAL

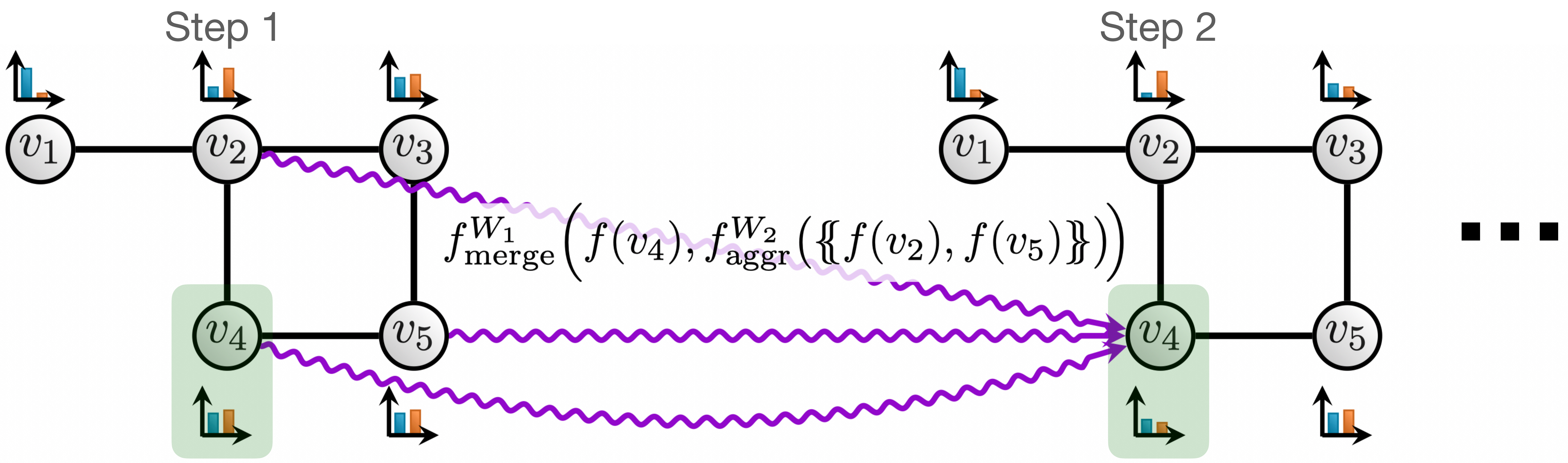


Figure 3: Illustration of the neighborhood aggregation step of a GNN around node v

Figure from our full-length survey at: https://arxiv.org/abs/2102.0

Combinatorial Optimization & Reasoning Graph Neural Networks

GNN for Comb. Opt.

- **Invariant** to node permutations
- Model parameters (W_1, W_2) are shared —> **applies to graphs of arbitrary size**
- **Expressive** local/global features are learned through non-linear layers

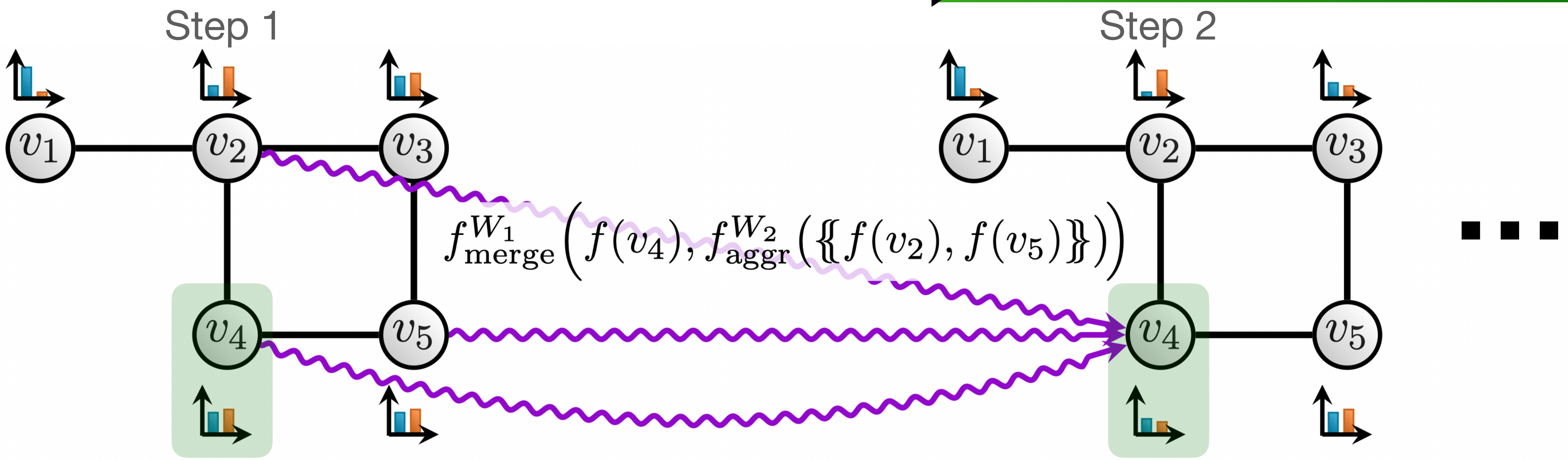


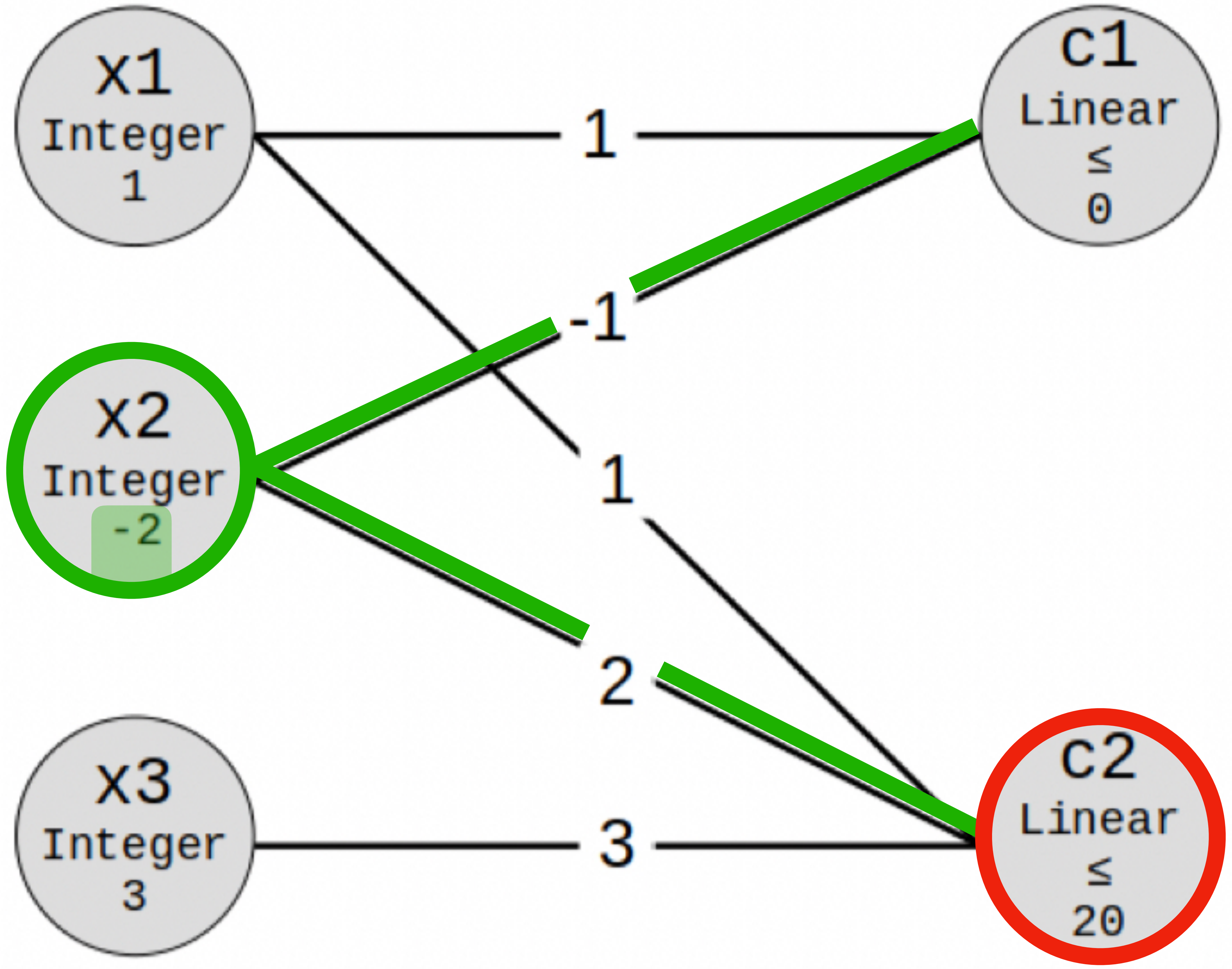
Figure 3: Illustration of the neighborhood aggregation step of a GNN around node v_4

Figure from our full-length survey at: https://arxiv.org/abs/2102.0

Graphs from Integer Programs

… or Constraint Programs, or SAT formulas, etc.

$$\begin{aligned} \text{minimize} \quad & x_1 - 2x_2 + 3x_3 \\ \text{subject to} \quad & x_1 - x_2 \leq 0 \\ & \boxed{x_1 + 2x_2 + 3x_3 \leq 20} \\ & x_i \in \mathbb{Z} \end{aligned}$$



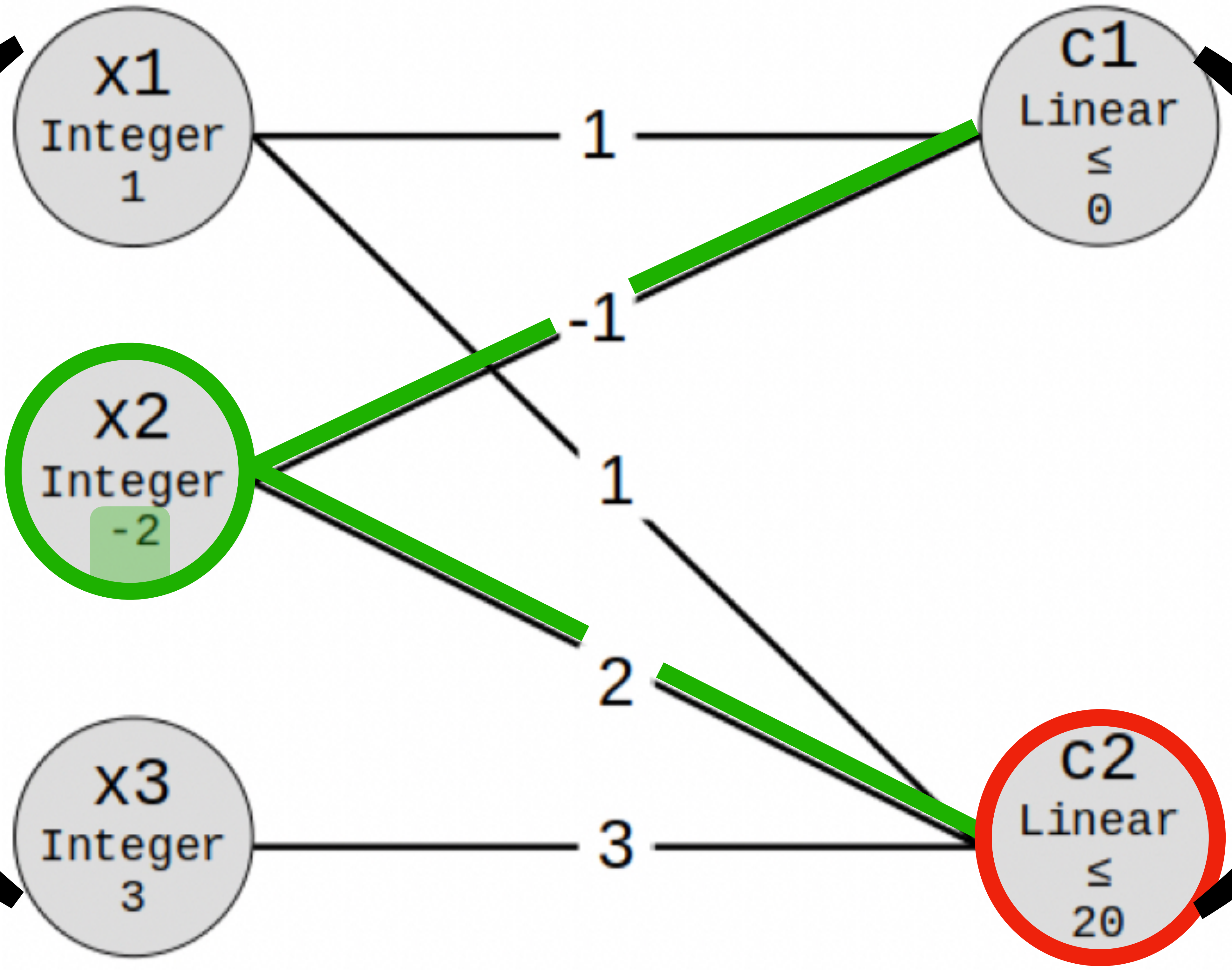
7

Graphs from Integer Programs

… or Constraint Programs, or SAT formulas, etc.

$$
\begin{aligned}
\text{minimize} \quad & x_1 - 2x_2 + 3x_3 \\
\text{subject to} \quad & x_1 - x_2 \leq 0 \\
& x_1 + 2x_2 + 3x_3 \leq 20 \\
& x_i \in \mathbb{Z}
\end{aligned}
$$

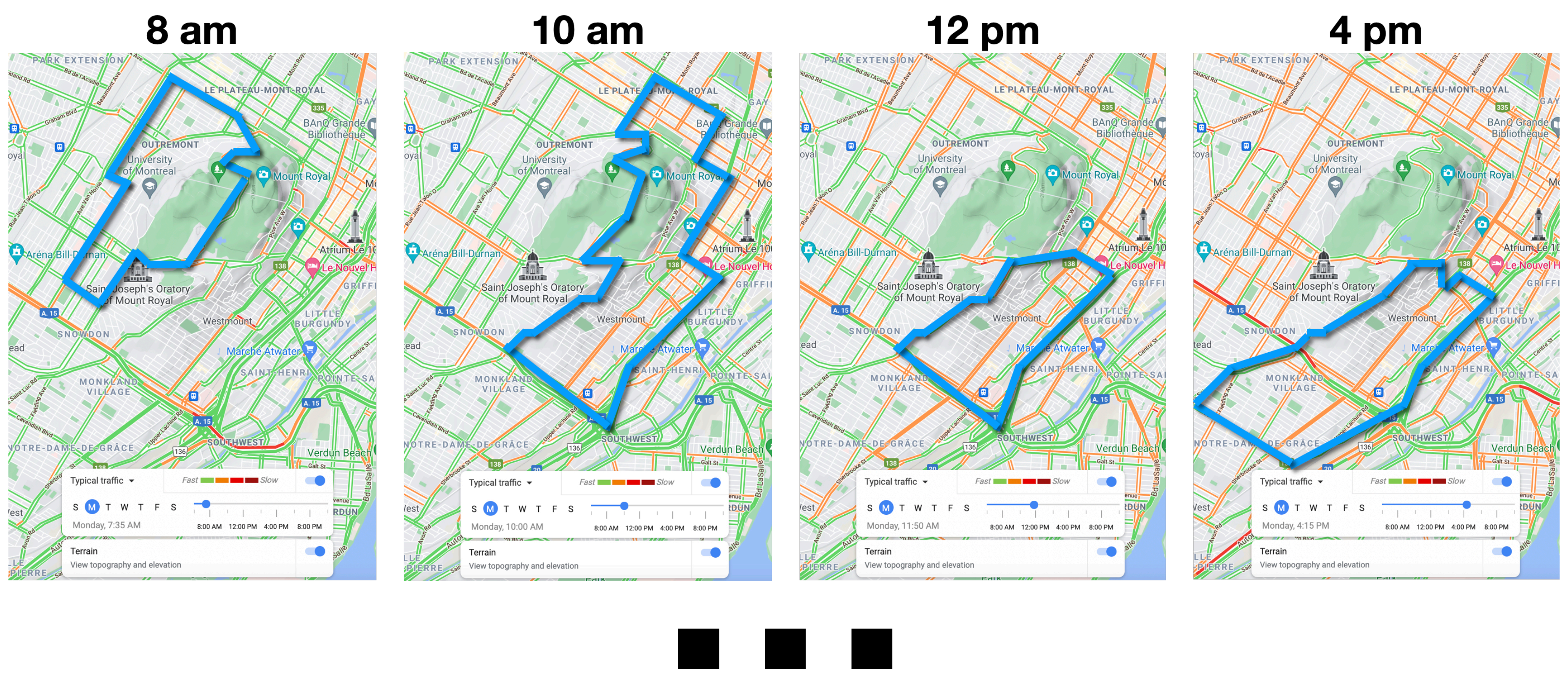
$\mathbf{GNN}_{W_1,W_2}\Bigg($



$\Bigg)$

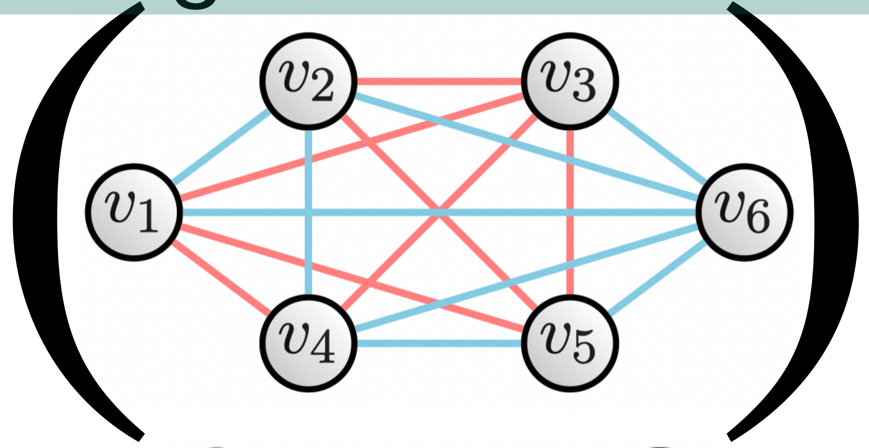
7

Training dataset of TSP instances



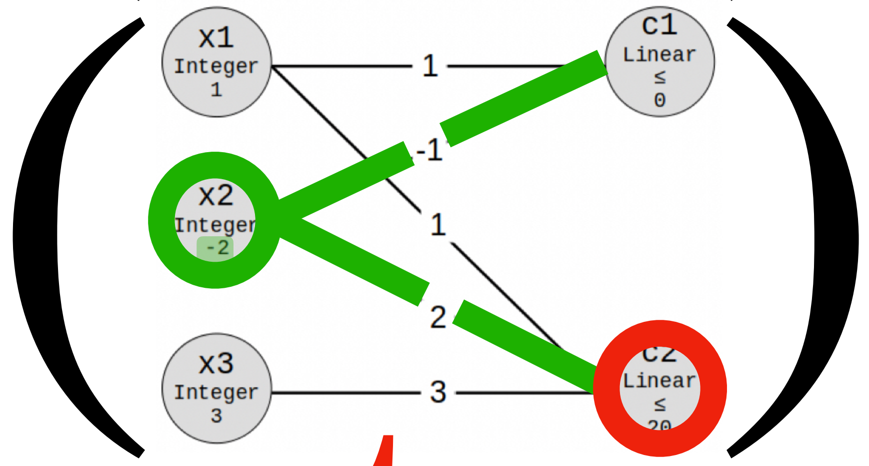
GNN "fit" to training instances

(original graph) $\mathbf{GNN}_{W_1, W_2}$



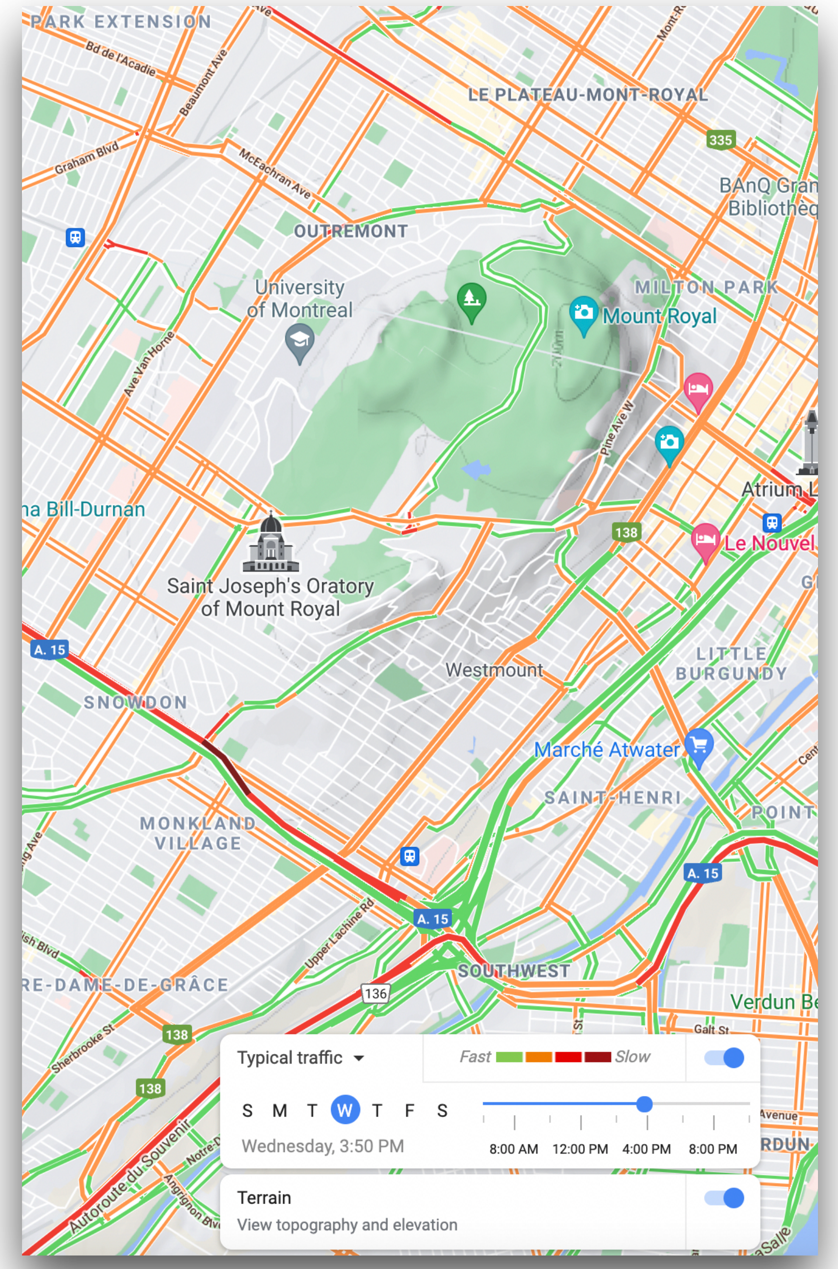
OR

(constraint graph) $\mathbf{GNN}_{W_1, W_2}$



Problem Instance

(Unseen test instance)



Optimization Formulation

Mixed Integer Programming (MIP)

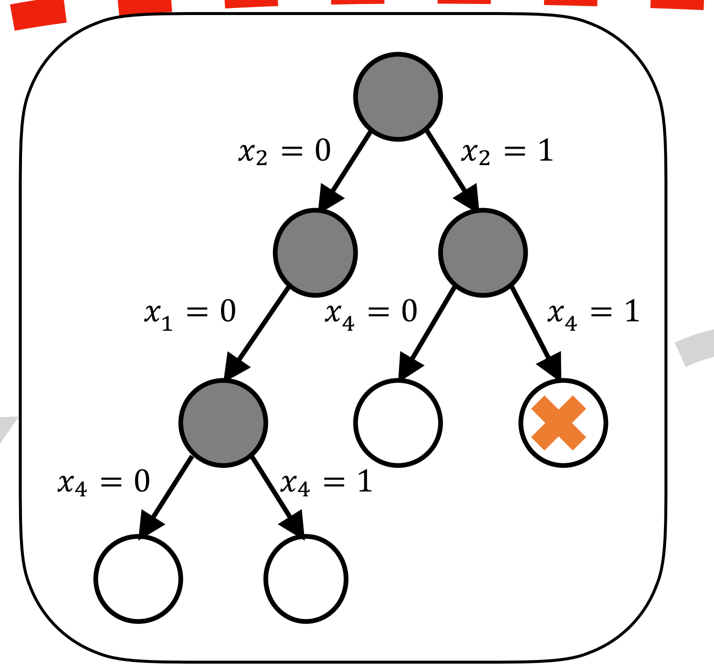
$$\min \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} w_{ij} x_{ij}$$

$$\text{subject to} \sum_{i=1, i \neq j}^{n} x_{ij} = 1 \qquad j \in [n],$$

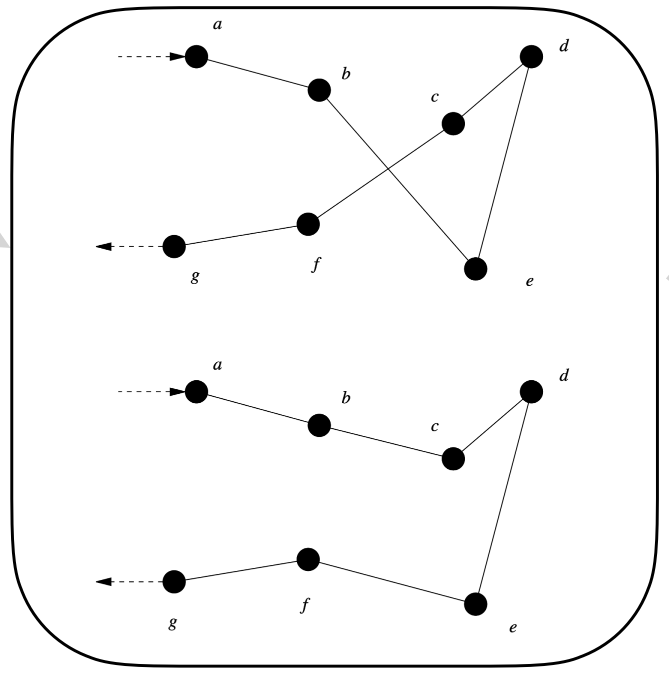
$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \qquad i \in [n],$$

$$\sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} \geq 1 \qquad \forall Q \subsetneq [n], |Q| \geq 2.$$

Algorithm

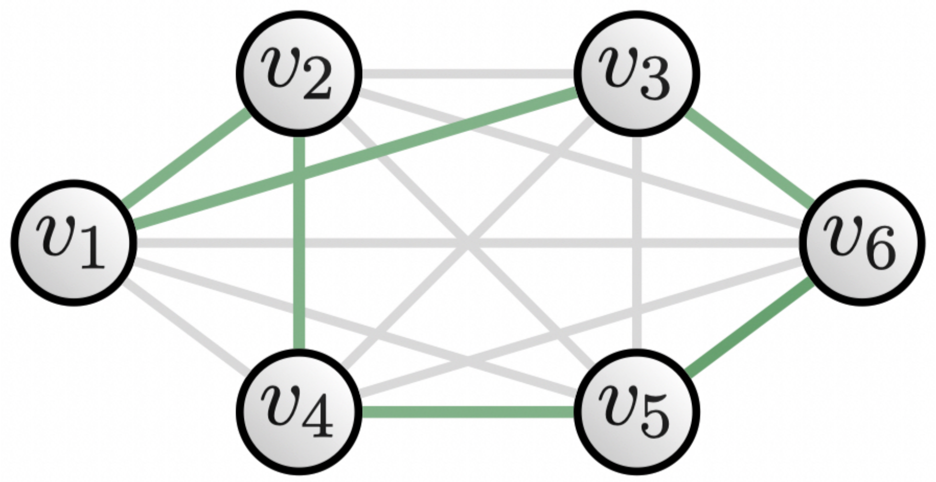


Tree search



Local or Large Neighborhood search

Feasible Solution



7

Pierre Selim, 2-opt procedure, distributed under a CC BY-SA 3.0 license.