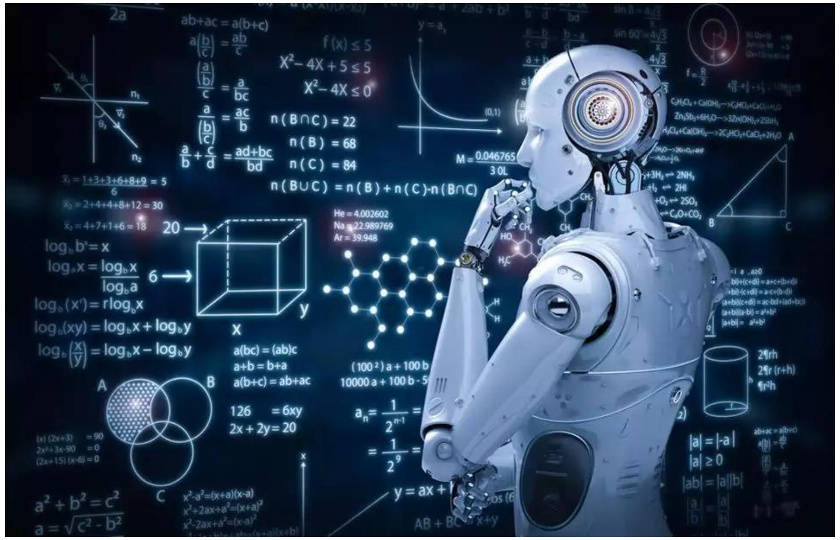
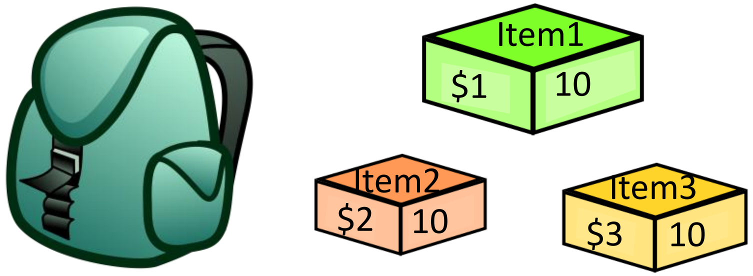
Branch & Learn with Post-hoc Correction for Predict+Optimize with Unknown Parameters in Constraints

Xinyi Hu [1], Jasper C.H. Lee [2], Jimmy H.M. Lee [1]
1. The Chinese University of Hong Kong
2. University of Wisconsin–Madison





Capacity: 2000

Machine learning

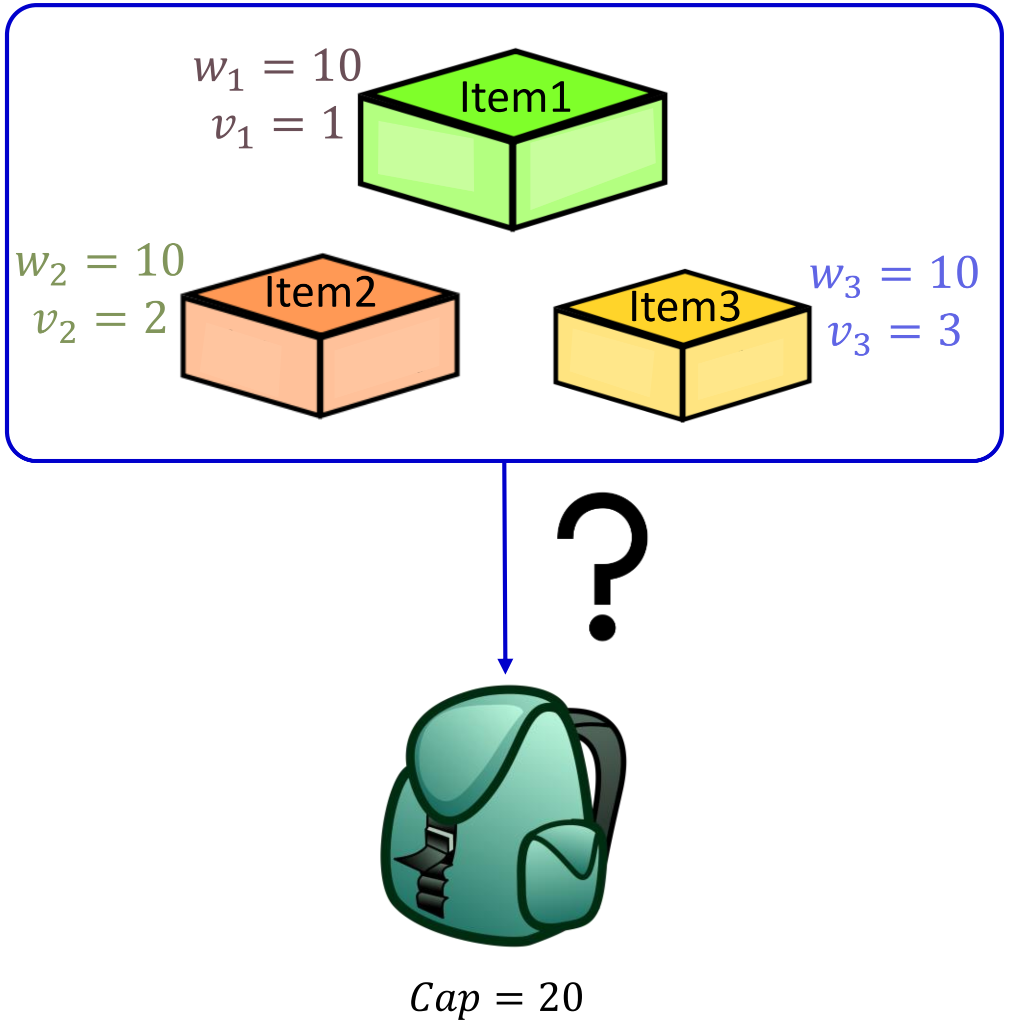
Constraint optimization

Predict+Optimize

[Elmachtoub and Grigas, Management Science 2022]

Optimization problems
with unknown parameters

Knapsack Problem



- 3 items, each with a weight w_i and a value v_i, the capacity Cap is 20.

Problem parameters

- Select items so that
  - the total weight is no more than the capacity and
  - the total value is maximized

- **Optimization Problem (OP):**

Decision variable

$$x_i = \begin{cases} 0, & \text{the } i\text{th item is not selected} \\ 1, & \text{the } i\text{th item is selected} \end{cases}$$

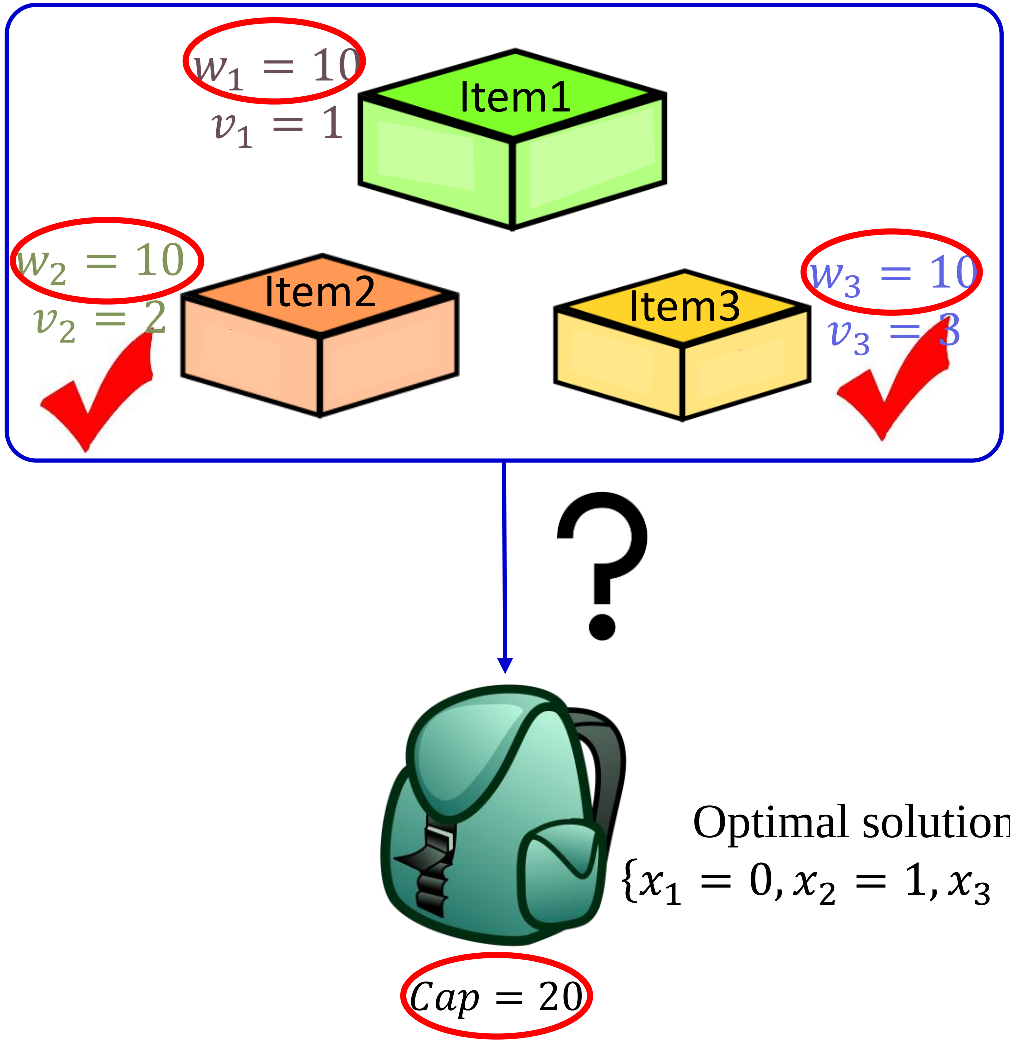
Objective function

$$\max_x x_1 + 2x_2 + 3x_3$$

Constraints

$$s.t.\ 10x_1 + 10x_2 + 10x_3 \leq 20$$
$$x_i \in \{0,1\}\ \forall i \in \{1,2,3\}$$

Knapsack Problem



Optimal solution:
$\{x_1 = 0, x_2 = 1, x_3 = 1\}$

- 3 items, each with a weight w_i and a value v_i, the capacity Cap is 20.

Problem parameters

- Select items so that
  - the total weight is no more than the capacity and
  - the total value is maximized

- **Optimization Problem (OP):**

Decision variable

$$x_i = \begin{cases} 0, & \text{the } i\text{th item is not selected} \\ 1, & \text{the } i\text{th item is selected} \end{cases}$$

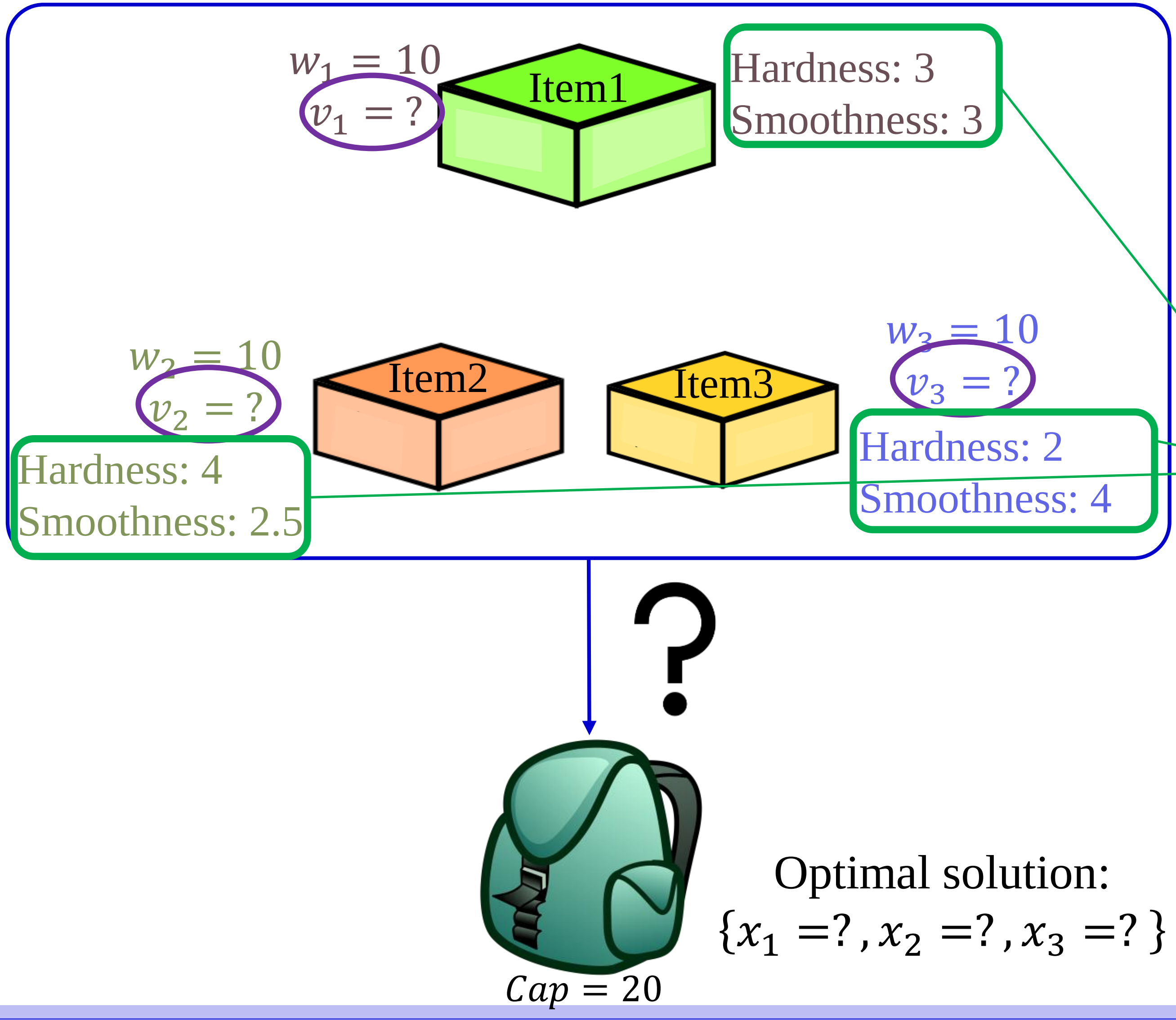
Objective function

$$\max_x x_1 + 2x_2 + 3x_3$$

Constraints

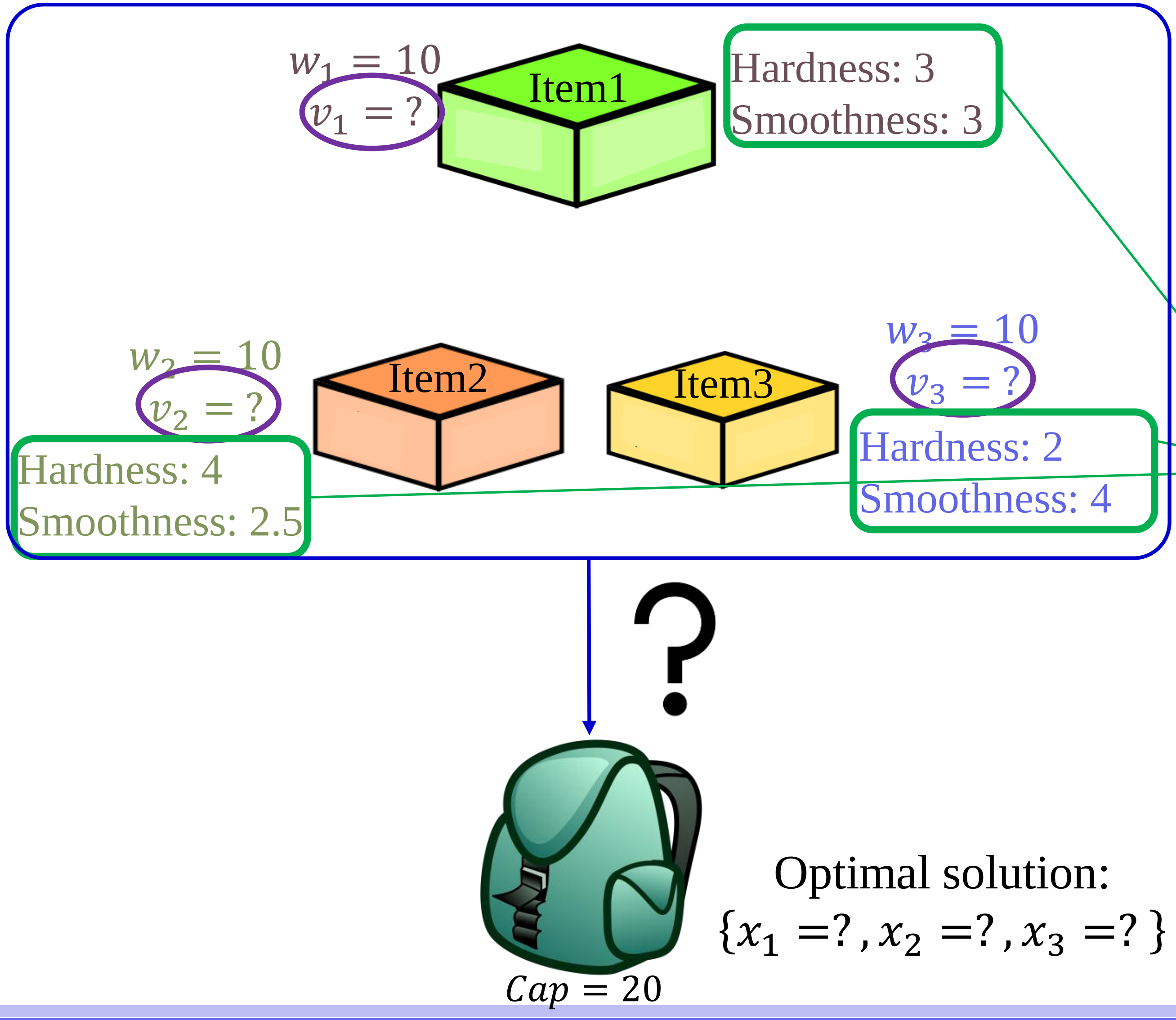
$$s.t. \; 10x_1 + 10x_2 + 10x_3 \le 20$$
$$x_i \in \{0,1\} \; \forall i \in \{1,2,3\}$$

Knapsack Problem with Unknown Values



- The value v_i is unknown.
- Select items so that
  - the total weight is no more than the capacity and
  - the total value is maximized
- **OP with Unknown Parameters:**
  - θ: unknown parameters, e.g., $\theta = \{v_1, v_2, v_3\}$
  - A: feature matrix
    - Hardness
    - Smoothness
    - $A = \begin{bmatrix} 3 & 3 \\ 4 & 2.5 \\ 2 & 4 \end{bmatrix}$

Knapsack Problem with Unknown Values



- The value v_i is unknown.
- Select items so that
  - the total weight is no more than the capacity and
  - maximize the total value
- **OP with Unknown Parameters:**
  - θ: unknown parameters, e.g., $\theta = \{v_1, v_2, v_3\}$
  - A: feature matrix
    - Hardness
    - Smoothness
  - $A = \begin{bmatrix} 3 & 3 \\ 4 & 2.5 \\ 2 & 4 \end{bmatrix}$
  - Historical data: $\{(A^1, \theta^1), (A^2, \theta^2), \dots, (A^k, \theta^k)\}$

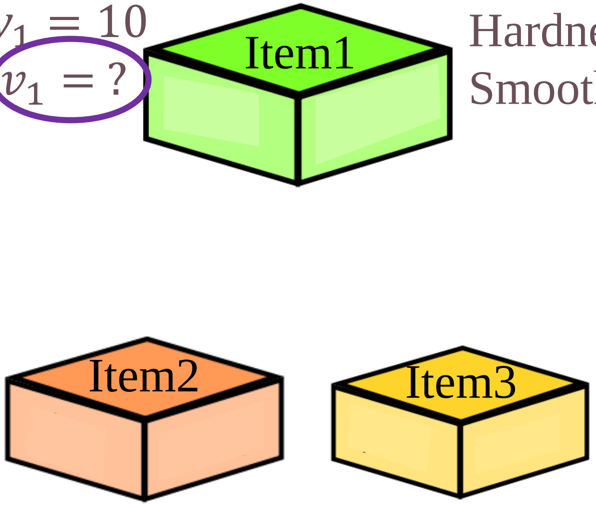
Historical features　True parameters

$$(A^i, \theta^i) = \left(\begin{bmatrix} 2 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \\ 2 \end{bmatrix} \right)$$

Knapsack Problem with Unknown Values

$w_1 = 10$
$v_1 = ?$

Hardness: 3
Smoothness: 3



$w_2 = 10$
$v_2 = ?$
Hardness: 4
Smoothness: 2.5

$w_3 = 10$
$v_3 = ?$
Hardness: 2
Smoothness: 4

Historical data:

$(A^1, \theta^1), \dots,$

$(A^i, \theta^i) = \left(\begin{bmatrix} 2 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 4 \\ 2 \end{bmatrix} \right),$

...

$Cap = 20$

Optimal solution:
$\{x_1 = ?, x_2 = ?, x_3 = ?\}$

- **OP with Unknown Parameters:**

$$x_i = \begin{cases} 0, & \text{the } i\text{th item is not selected} \\ 1, & \text{the } i\text{th item is selected} \end{cases}$$

$$\max_x v_1 x_1 + v_2 x_2 + v_3 x_3$$
$$s.t.\ 10x_1 + 10x_2 + 10x_3 \leq 20$$
$$x_i \in \{0,1\}\ \forall i \in \{1,2,3\}$$

Unknown parameters

The Pipeline

Predict

Features Historical data → Prediction model → Predicted parameters

What kind of loss function should we use for the training?

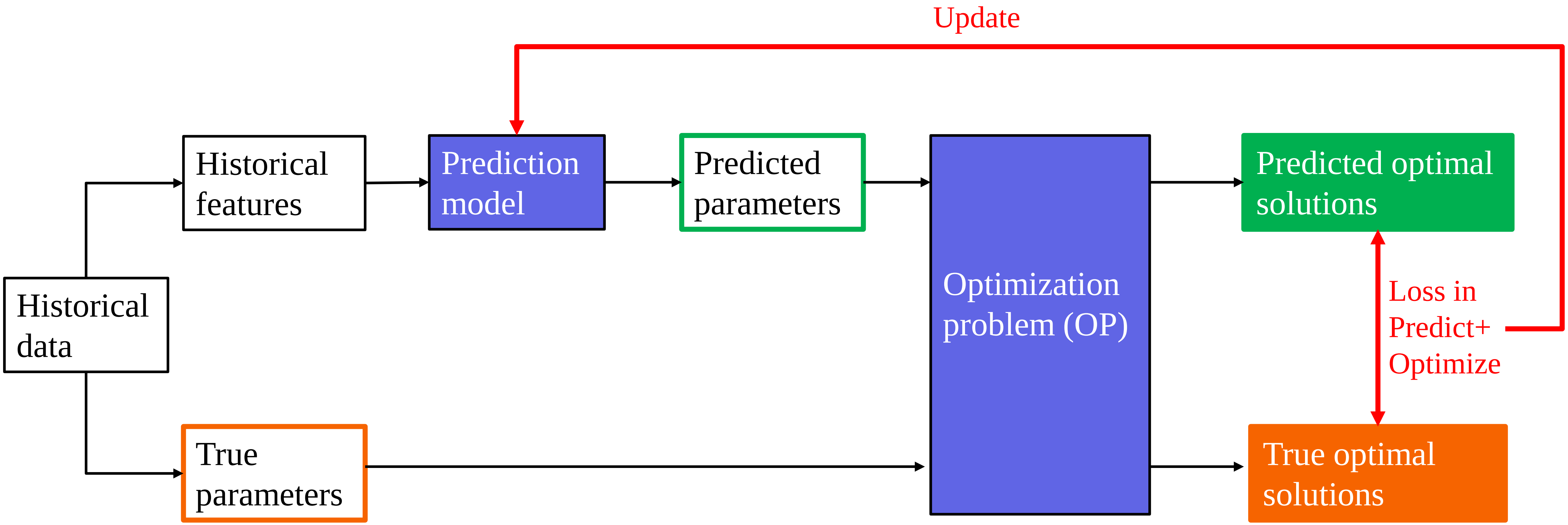
Optimize

→ Optimization problem (OP) → Predicted solutions

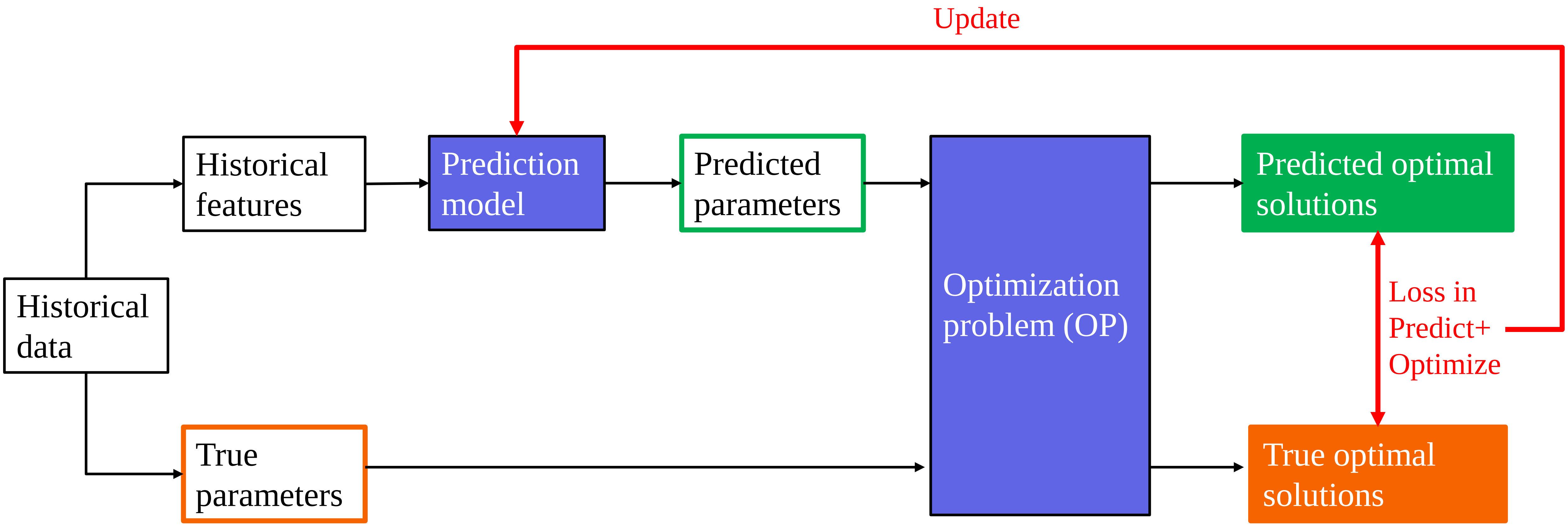
Goal
good predicted solutions under true parameters

Predict+Optimize

Aims to incorporate optimization problems into the loss function



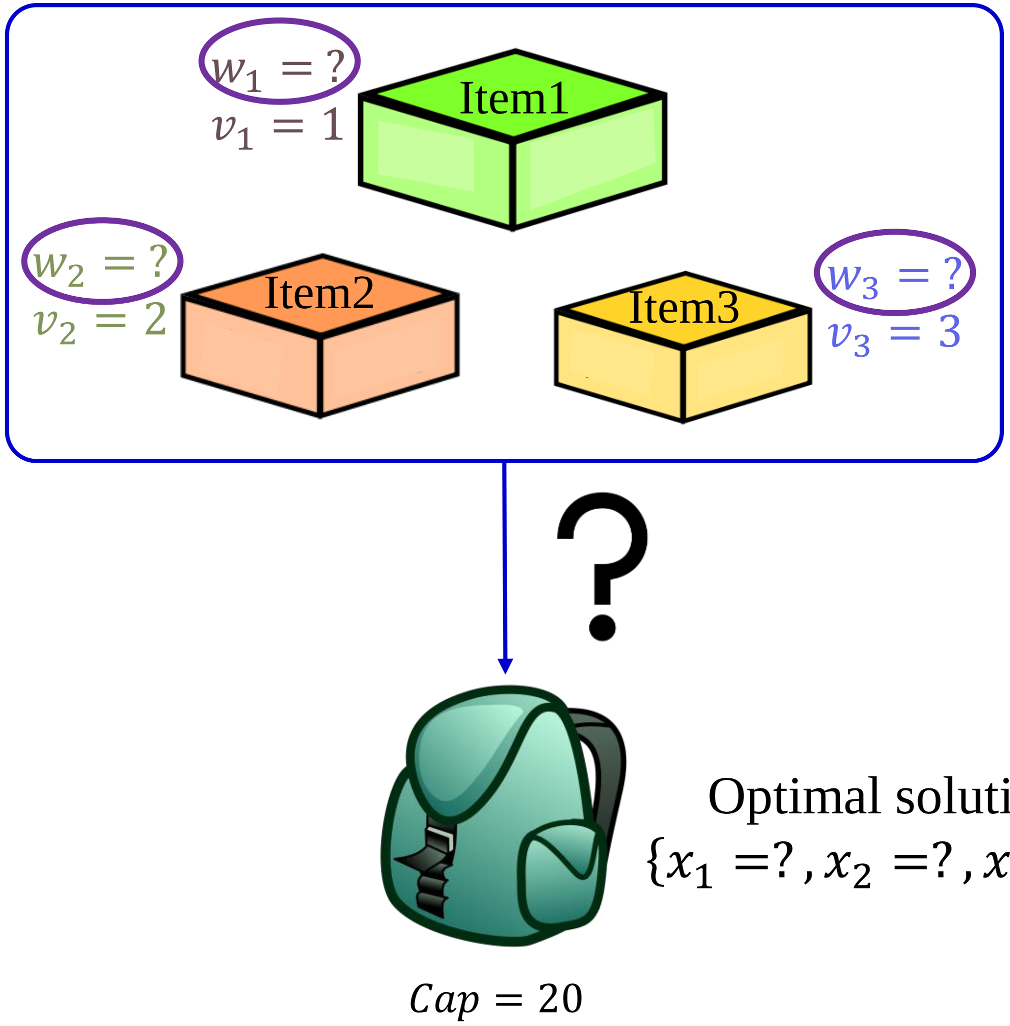
Unknown Parameters in Objectives



- Regret ([Demirović et al., 2019a], [Elmachtoub and Grigas, Management Science 2022]):

‖ True optimal value − Predicted optimal value ‖

Unknown Parameters in Constraints



Optimal solution:
$\{x_1 =?, x_2 =?, x_3 =?\}$

- Knapsack with unknown weights

$$\max_x x_1 + 2x_2 + 3x_3$$
$$s.t. \; w_1 x_1 + w_2 x_2 + w_3 x_3 \le 20$$
$$x_i \in \{0,1\} \; \forall i \in \{1,2,3\}$$

Unknown parameters in constraints

E.g.,

- Predicted weights:

$$\{\widehat{w_1} = \widehat{w_2} = \widehat{w_3} = 5\}$$

  - Predicted optimal solution:

$$\{x_1 = x_2 = x_3 = 1\}$$

infeasible

- True weights:

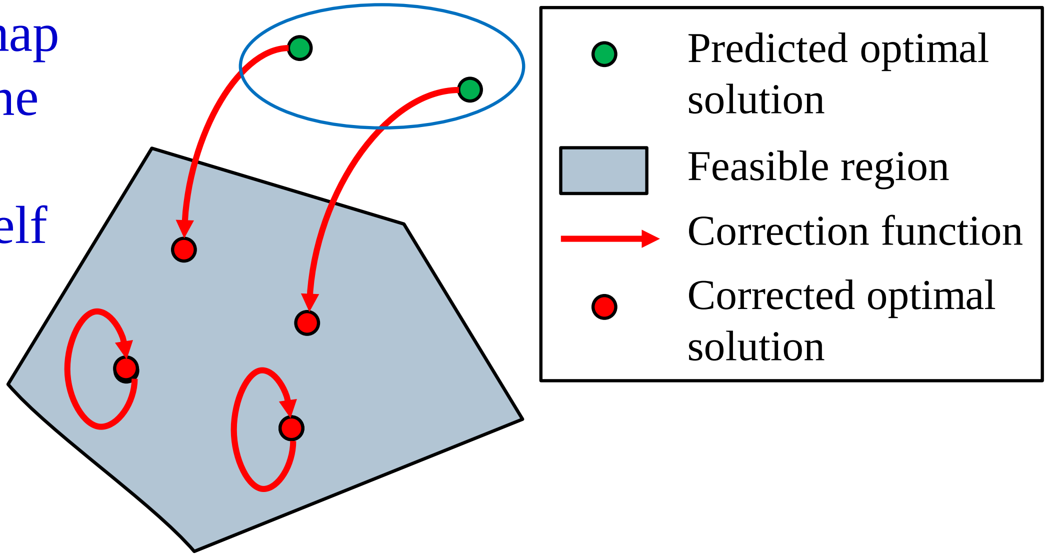
$$\{w_1 = w_2 = w_3 = 10\}$$

Unknown Parameters in Constraints

Some applications allow solution modification after true parameters are revealed

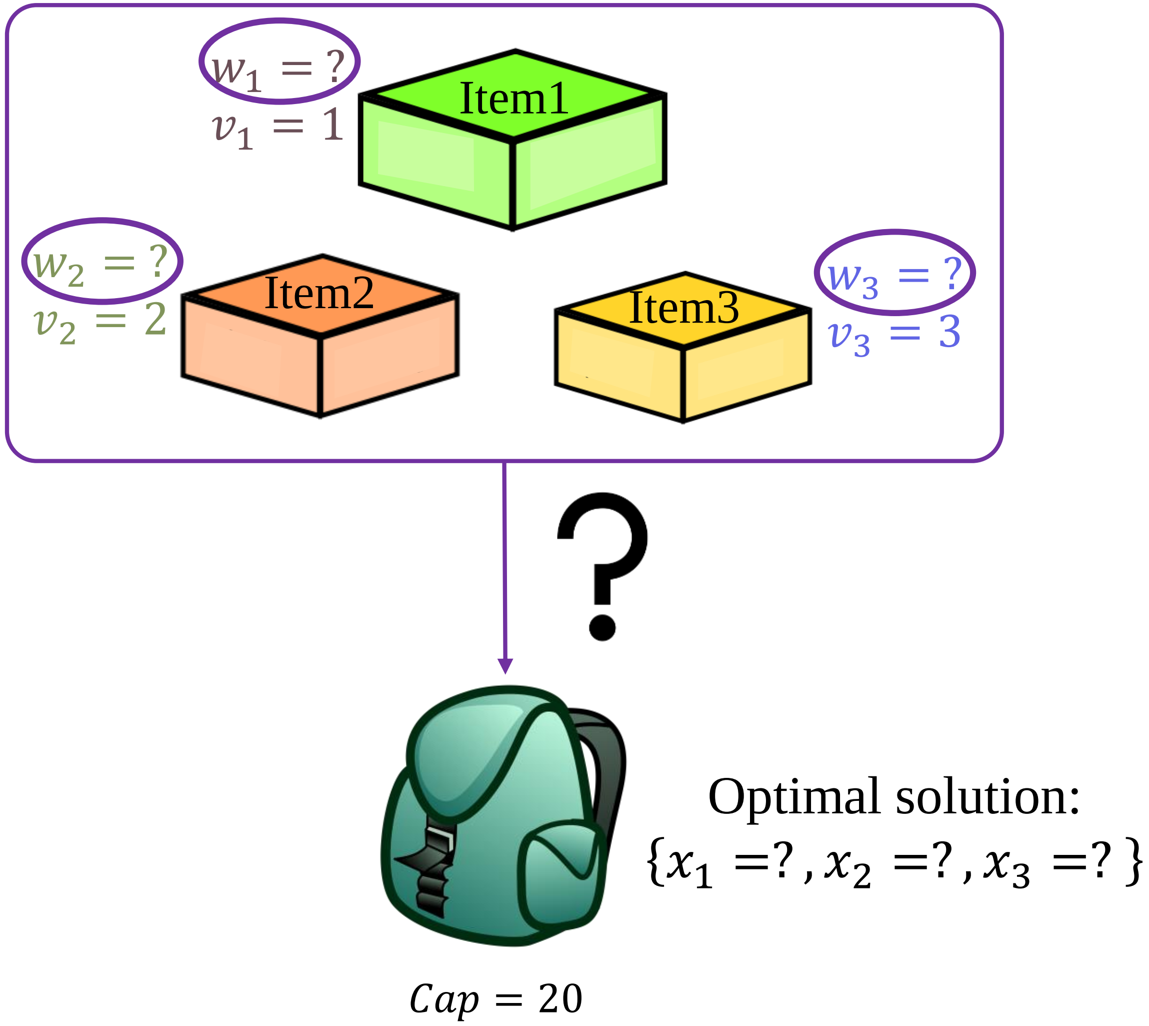
[Hu et al., AAAI 2023]

- **Correction function** should map
  - (a) each infeasible solution to one in the feasible region
  - (b) every feasible solution to itself



The space of possible correction functions: problem and application specific

Post-hoc Correction: Knapsack Example



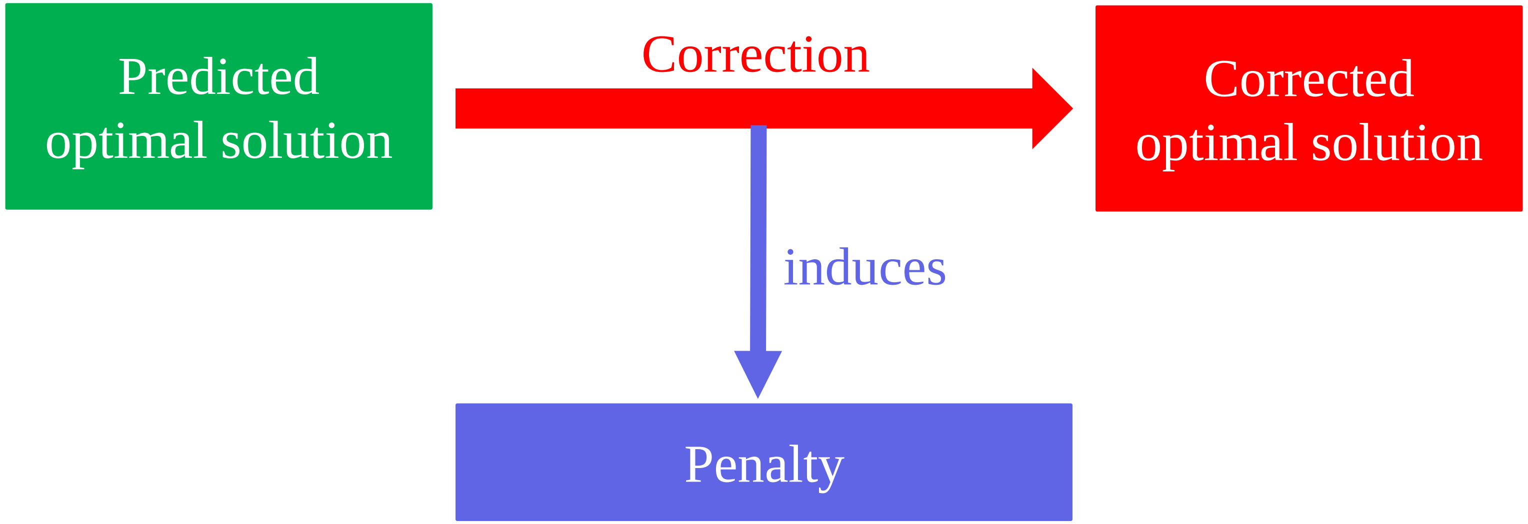
- If the weights are unknown?

$$\max_x \sum_i v_i x_i$$
$$s.t. \sum_i w_i x_i \leq Cap$$
$$x_i \in \{0,1\} \; \forall i \in \{1,2,3\}$$

- When the total weight of the selected items exceeds the capacity:
  - Correction function: remove the items one by one in increasing order of the ratios of value/weight

- Penalty function: removal fee
  - problem and application specific

Unknown Parameters in Constraints



- Post-hoc regret ([Hu et al., AAAI 2023]):

‖ True optimal value – Corrected optimal value ‖ + Penalty

- When only the objective contains unknown parameters, degenerates into Regret

Our Contributions

- Handles optimization problems with unknown parameters in both the objective and constraints

Prior works

- Most focus on unknown parameters **only in objective**
  - [Mandi et al., ICML 2022]
  - [Jeong et al., ICML 2022]
  - [Guler et al., AAAI 2022]
  - [Hu et al., NeurIPS 2022]
  - …
- Focus on unknown parameters **in both objective and constraints**
  - [Hu et al., AAAI 2023]
    - New loss: post-hoc regret (non-differentiable)
    - An approximation method for covering and packing LPs
      - Use an approximation of post-hoc regret

Our work

- Focus on unknown parameters **in both objective and constraints**
  - An exact method for recursively and iteratively solvable problems
    - Use post-hoc regret
  - Experimentally compare the proposed exact method with the prior approximation method
  - Empirically study different combinations of the 2 key components of the framework

Branch & Learn with Post-hoc Correction

- Assumption: the prediction model is linear
- To train models without computing gradients
  - Adopt the coordinate descent based method proposed by previous work [Hu et al., NeurIPS 2022]

Previous work [Hu et al., NeurIPS 2022] :

- For unknown parameters only in **objectives**
- Use **Regret** as the loss function

Algorithm 1: Branch & Learn

Input: A Para-OP $P(\theta)$ and a training data set $\{(A^1, \theta^1), \ldots, (A^n, \theta^n)\}$
Output: a coefficient vector $\alpha \in \mathbb{R}^m$

Initialize α arbitrarily and $k \leftarrow 0$;
while *not converged* $\wedge$ *resources remain* **do**
  $k \leftarrow (k \mod m) + 1$;
  Initialize L to be the zero constant function;
  for $i \in [1, 2, \ldots, n]$ **do**
    $(P^i_\gamma, I_0) \leftarrow \texttt{Construct}(P(\theta), k, A^i)$;
    $E^i(\gamma) \leftarrow \texttt{Convert}(P^i_\gamma, I_0)$;
    $L^i(\gamma) \leftarrow \texttt{Evaluate}(\mathbb{I}(E^i), \theta^i, I_0)$;
    $L(\gamma) \leftarrow L(\gamma) + L^i(\gamma)$;
  $\alpha_k \leftarrow \arg\min_{\gamma \in \mathbb{R}} L(\gamma)$;
return α;

- Update prediction model coefficients via coordinate descent
- Each iteration contains 3 functions:
  - Construct(): construct an OP with unknown parameters
  - Convert(): solve the OP with unknown parameters and get predicted optimal solution
  - Evaluate(): compute the Regret

Branch & Learn with Post-hoc Correction

- Assumption: the prediction model is linear
- To train models without computing gradients
  - Adopt the coordinate descent based method proposed by previous work [Hu et al., NeurIPS 2022]

Previous work [Hu et al., NeurIPS 2022] :

- For unknown parameters only in **objectives**
- Use **Regret** as the loss function

Algorithm 1: Branch & Learn

Input: A Para-OP $P(\theta)$ and a training data set $\{(A^1, \theta^1), \ldots, (A^n, \theta^n)\}$
Output: a coefficient vector $\alpha \in \mathbb{R}^m$

Initialize α arbitrarily and $k \leftarrow 0$;
while *not converged* $\wedge$ *resources remain* **do**
  $k \leftarrow (k \mod m) + 1$;
  Initialize L to be the zero constant function;
  for $i \in [1, 2, \ldots, n]$ **do**
    $(P^i_\gamma, I_0) \leftarrow \texttt{Construct}(P(\theta), k, A^i)$;
    $E^i(\gamma) \leftarrow \texttt{Convert}(P^i_\gamma, I_0)$;
    $L^i(\gamma) \leftarrow \texttt{Evaluate}(\mathbb{I}(E^i), \theta^i, I_0)$;
    $L(\gamma) \leftarrow L(\gamma) + L^i(\gamma)$;
  $\alpha_k \leftarrow \arg\min_{\gamma \in \mathbb{R}} L(\gamma)$;
return α;

2[nd] change: change Evaluate()

Our work:

- For unknown parameters in **objectives and constraints**
- Use **Post-hoc Regret** as the loss function

Algorithm 2: Branch & Learn with Post-hoc Correction

Input: A Para-COP $P(\theta)$ and a training data set $\{(A^1, \theta^1), \ldots, (A^n, \theta^n)\}$
Output: a coefficient vector $\alpha \in \mathbb{R}^m$

Initialize α arbitrarily and $k \leftarrow 0$;
while *not converged* $\wedge$ *resources remain* **do**
  $k \leftarrow (k \bmod m) + 1$;
  Initialize L to be the zero constant function;
  for $i \in [1, 2, \ldots, n]$ **do**
    $(P^i_\gamma, I_0) \leftarrow \texttt{Construct}(P(\theta), k, A^i)$;
    $E^i(\gamma) \leftarrow \texttt{Convert}(P^i_\gamma, I_0)$;
    $C^i(\gamma) \leftarrow \texttt{Correct}(E^i, \theta^i, I_0)$;
    $L^i(\gamma)^* \leftarrow \texttt{Evaluate}(\mathbb{I}(E^i), \mathbb{I}(C^i), \theta^i, I_0)$;
    $L(\gamma)^* \leftarrow L(\gamma)^* + L^i(\gamma)^*$;
  $\alpha_k \leftarrow \arg\min_{\gamma \in \mathbb{R}} L(\gamma)^*$;
return α;

1[st] change: add one function

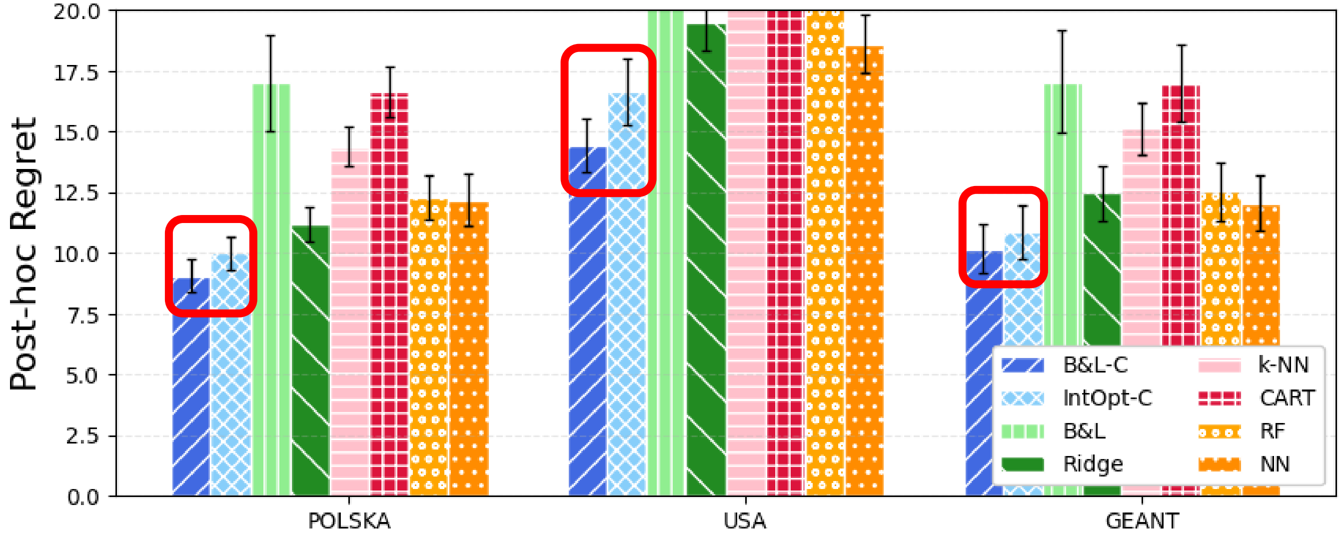
Experimental Evaluation

- Experimentally compare the exact method with the prior approximation method

- E.g., Maximum Flow with Unknown Edge Capacities
  - Packing LP
  - Aim: find the largest flow sent from a source to a terminal in a directed graph
  - Constraint: the flow sent on each edge cannot exceed the edge capacity

- B&L: the prior exact method using Regret
- IntOpt-C: the prior approximation method using an approximation of Post-hoc Regret
- B&L-C: the proposed exact method using Post-hoc Regret

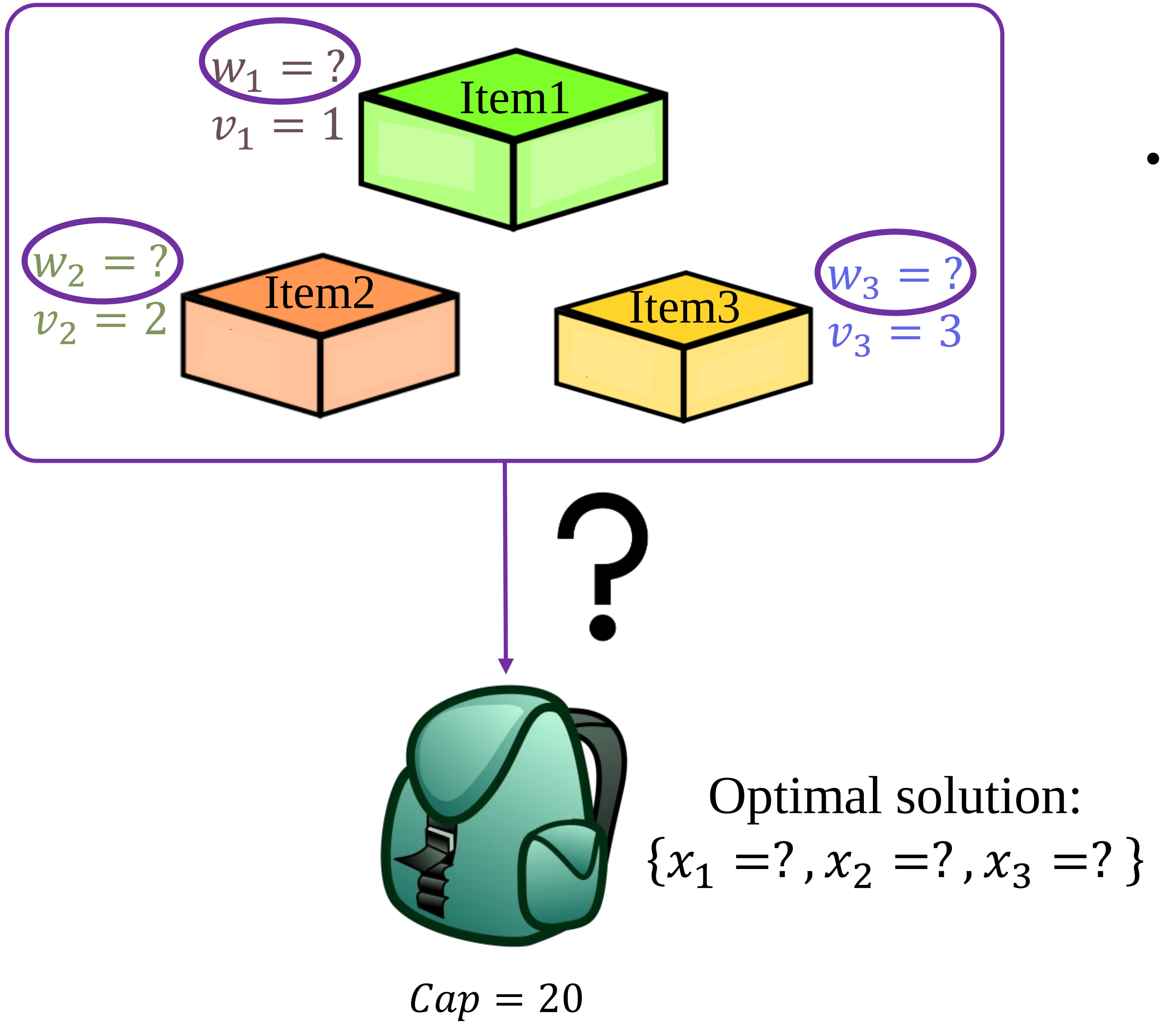
better solution quality



longer runtime

Runtime(s)	POLSKA	USANet	GÉANT
B&L-C	66.54	411.67	48.32
IntOpt-C	18.65	132.22	15.48
B&L	40.30	288.43	29.90
RF	4.11	11.00	11.89
NN	10.33	12.82	13.89

Experimental Evaluation



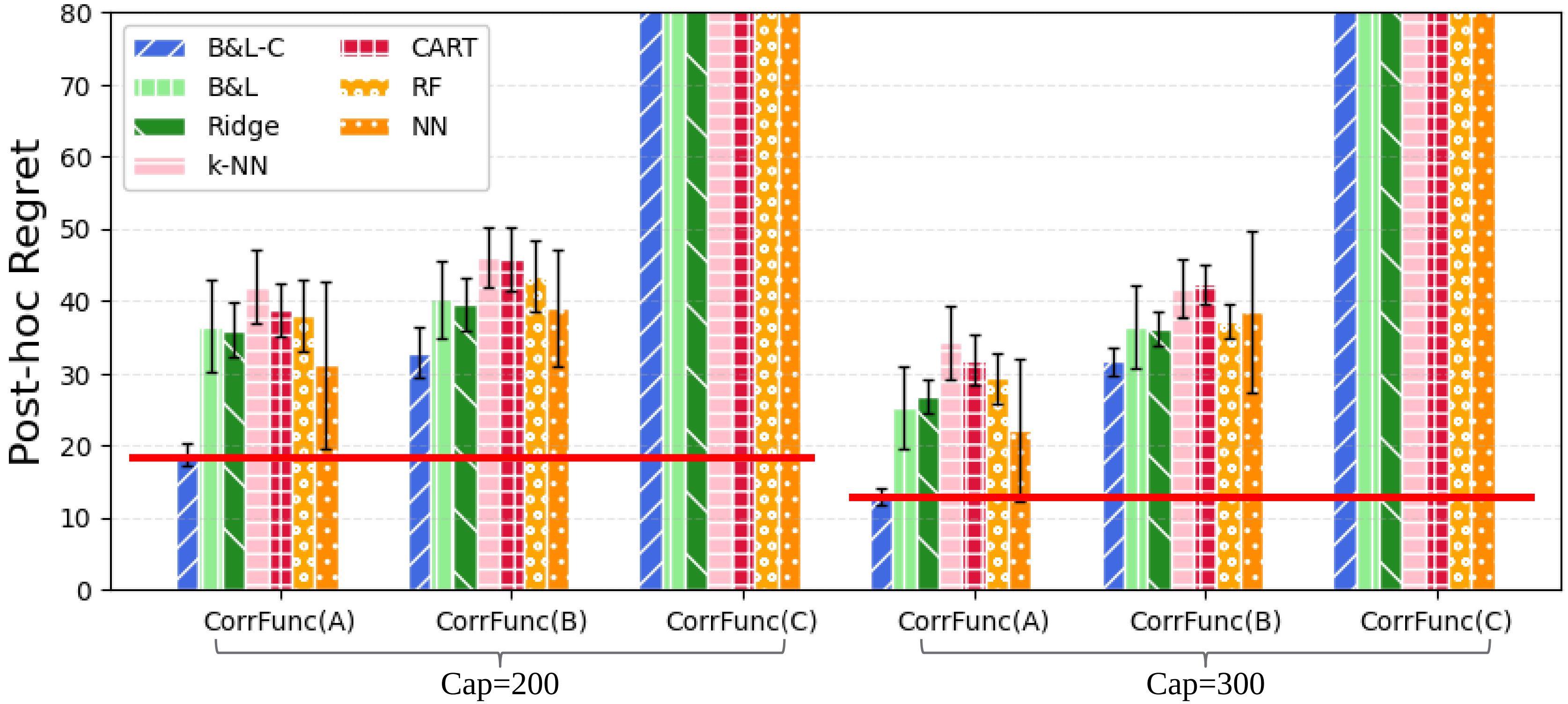
- Empirically study different combinations of the correction function and the penalty function on 3 OPs

- E.g., 0-1 Knapsack with Unknown Weights

  - 3 correction functions
    - A: remove the items one by one in increasing order of the ratios of value/weight
    - B: remove the items one by one in decreasing order of the weights
    - C: remove all items

  - 2 penalty functions
    - I: when the i-th item is removed, $\sigma_i v_i$ units of value is deducted
    - II: whenever a selected item is removed, K (a constant) units of value is deducted

Experimental Evaluation



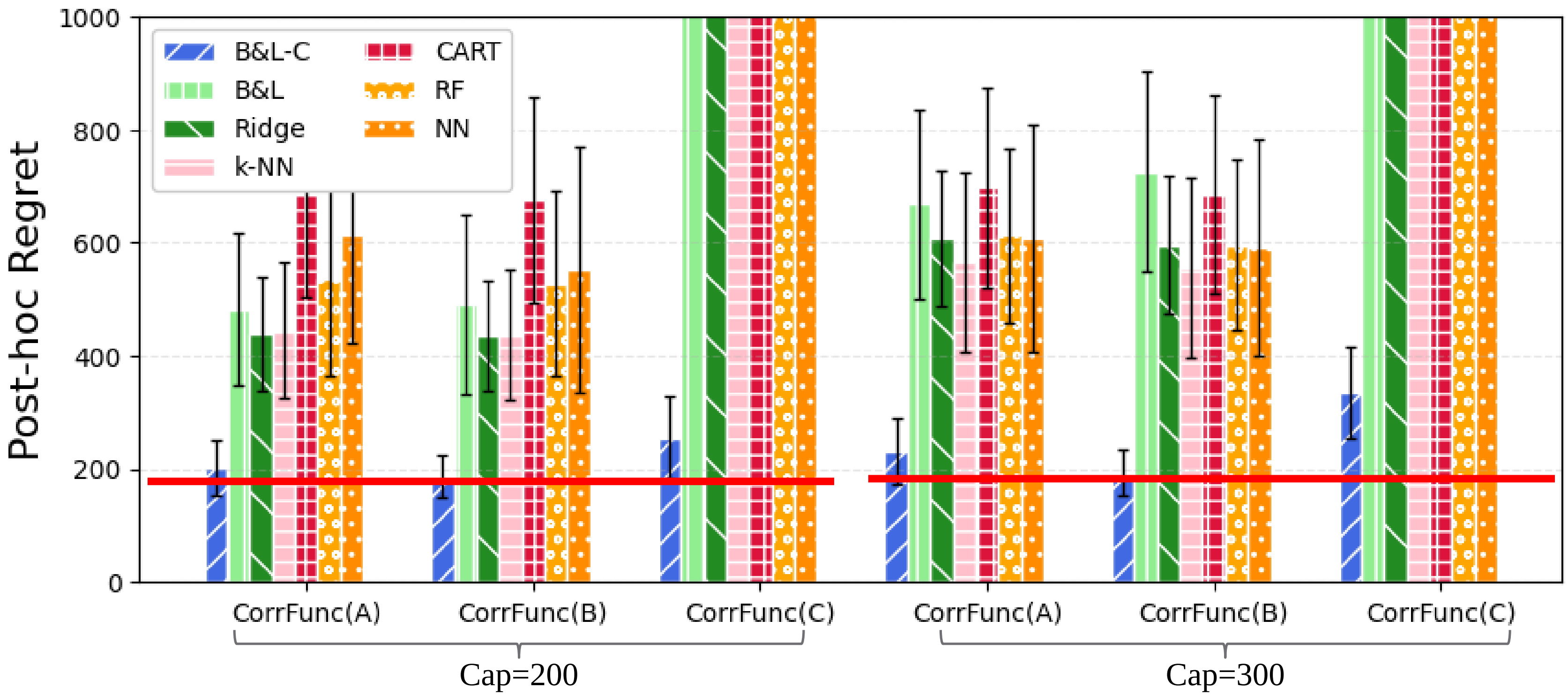
Post-hoc Regrets for 0-1 knapsack with unknown weights using different correction functions with Penalty Function I.

Penalty Function I:

Post-hoc Regrets achieved by using Correction Function A are much smaller than Post-hoc Regrets achieved by using Correction Function B or C

➔ Correction Function A is more suitable to use than Correction Functions B or C

Experimental Evaluation



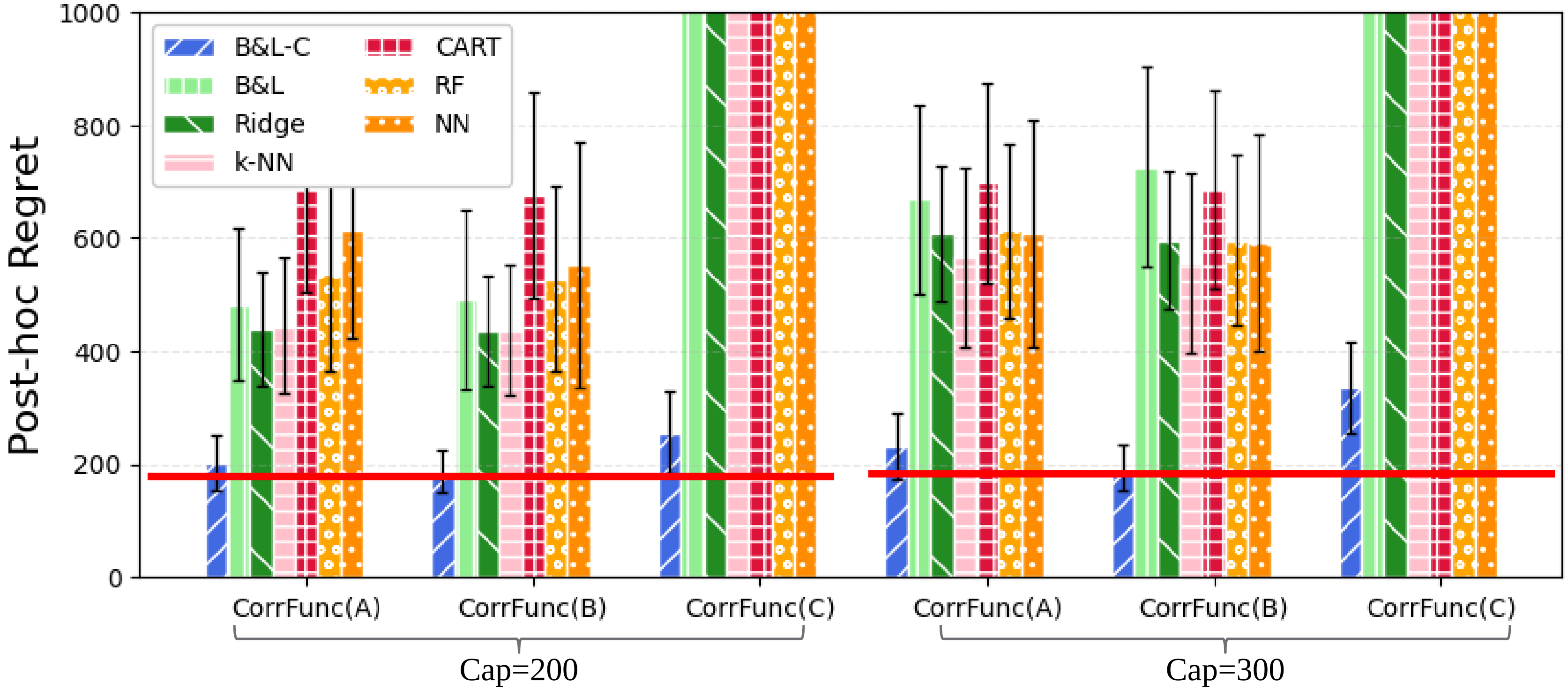
Post-hoc Regrets for 0-1 knapsack with unknown weights using different correction functions with Penalty Function II.

Penalty Function II:

Post-hoc Regrets achieved by using Correction Function B are smaller than Post-hoc Regrets achieved by using Correction Function A or C

➔ Correction Function B is more suitable to use than Correction Functions A or C

Experimental Evaluation



Post-hoc Regrets for 0-1 knapsack with unknown weights using different correction functions with Penalty Function II.

- Correction Function A: remove the items one by one in increasing order of the ratios of value/weight
- Correction Function B: remove the items one by one in decreasing order of the weights
- Correction Function C: remove all items
- Penalty Function II: whenever a selected item is removed, K (a constant) units of value is deducted

Contributions

- An exact method for recursively and iteratively solvable problems with unknown parameters in both objectives and constraints
- Experimentally compare the proposed exact method with the prior approximation method
- Empirically study different combinations of the 2 key components of the framework

Questions? xyhu@cse.cuhk.edu.hk