Learning to Assign: Towards Fair Task Assignment in Large-Scale Ride Hailing

Dingyuan Shi[1], Yongxin Tong[1], Zimu Zhou[2],

Bingchen Song[1], Weifeng Lv[1], Qiang Yang[3,4]

1 Beihang University
2 Singapore Management University
3 The Hong Kong University of Science and Technology
4 WeBank Co., Ltd

Outline

- Background & Motivation
- Problem Statement
- Our Solutions
- Experiments
- Conclusion

Outline

- Background & Motivation
- Problem Statement
- Our Solutions
- Experiments
- Conclusion

Background & Motivation

- **Ride hailing drivers do not get paid fairly.**
- **Order dispatching algorithm largely impacts income fairness.**

When Scholars Collaborate With Tech Companies, How Reliable Are the Findings?

and noted that his team had provided analysis using alternative assumptions. He also said that the study had found a wide variation in earnings among drivers, and that driving might be a worse deal for full-timers than those who drive casually or part

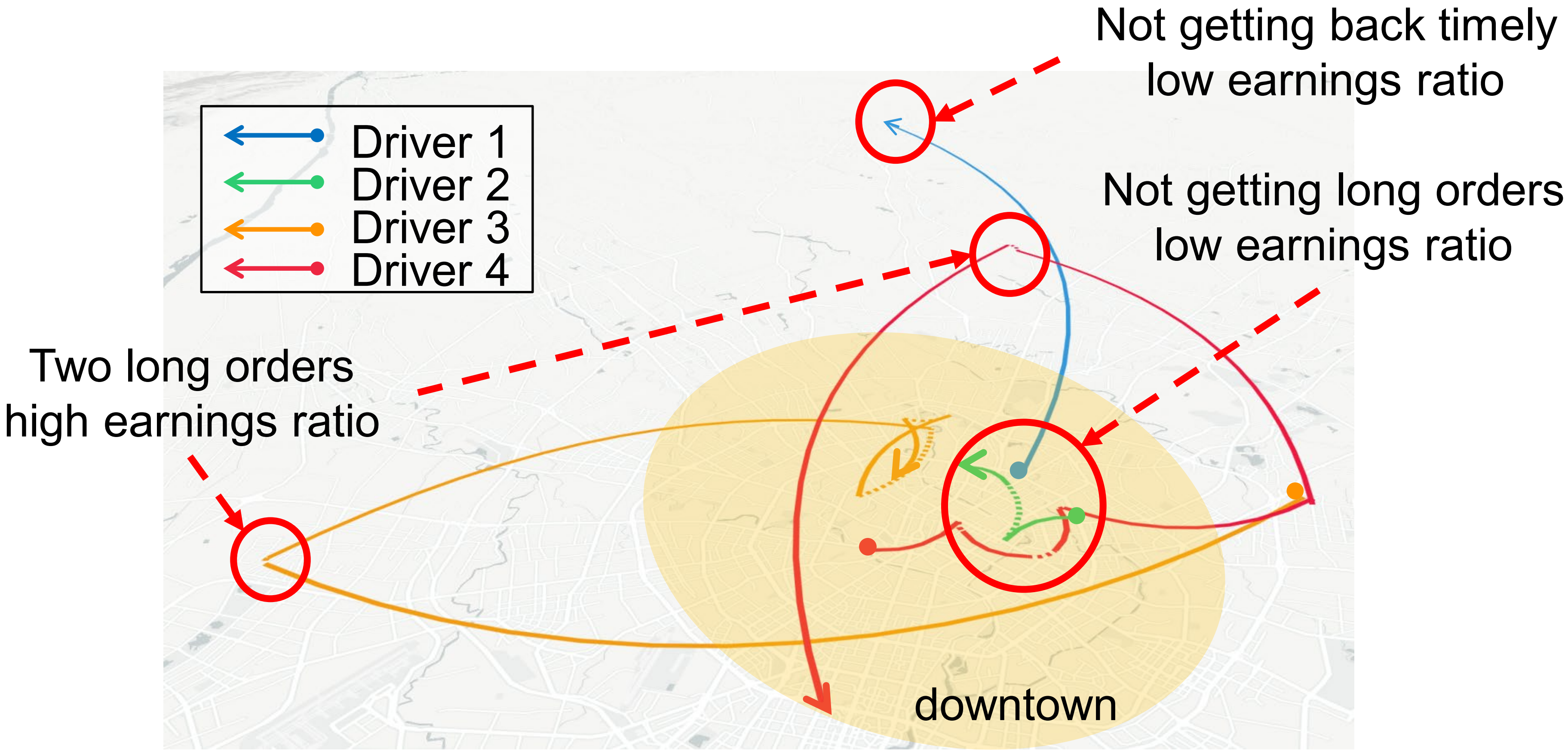
Uber and Lyft drivers strike for pay transparency — after algorithms made it harder to understand

Demonstrations took place in at least eight major cities nationwide Wednesday.

Driver income fairness has long been ignored, which deflates drivers.

Background & Motivation

- **Analyze driver income fairness from history data**



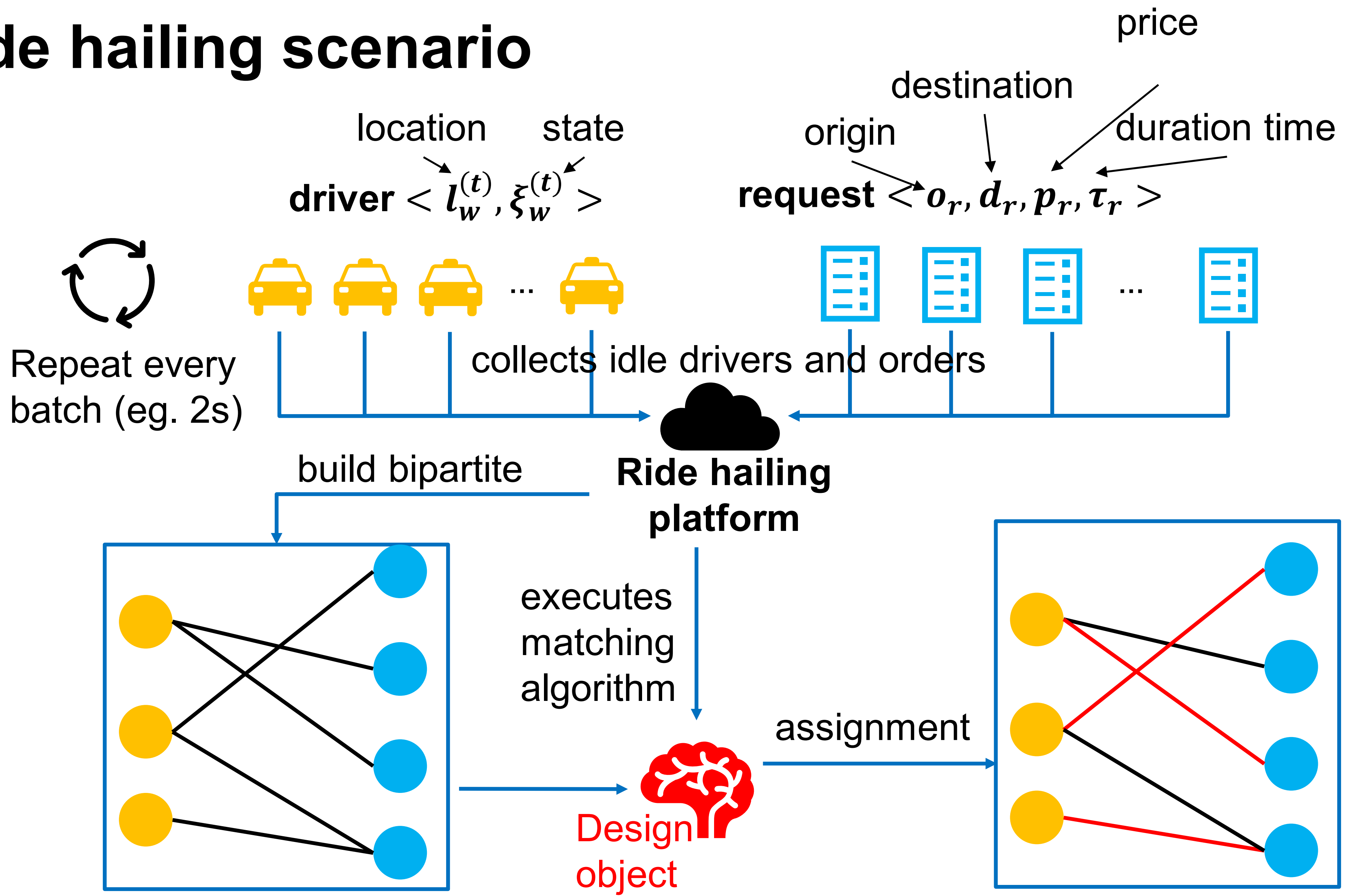
How to maximize utility while maintaining driver income fairness?

Outline

- Background & Motivation
- Problem Statement
- Our Solutions
- Experiments
- Conclusion

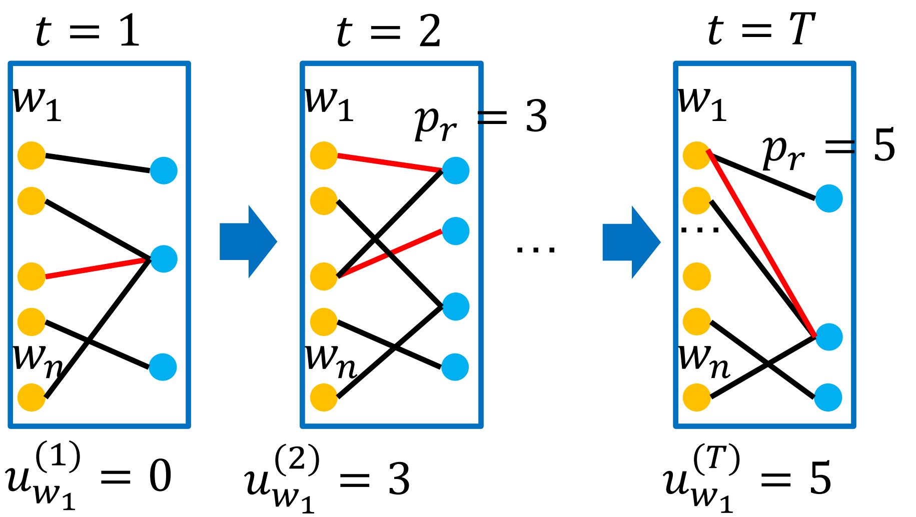
Problem Statement

- **Ride hailing scenario**



Problem Statement

- **Goal and constrain**

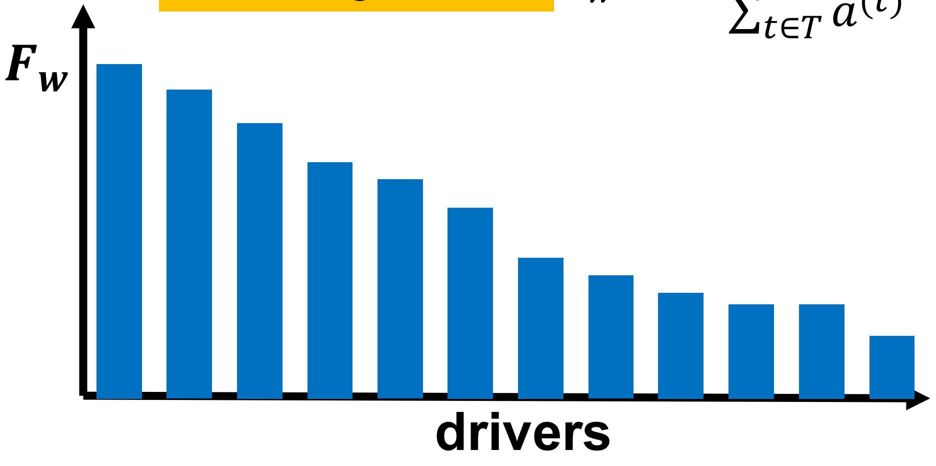


All drivers' income of the day.

$$U = \sum_{w \in W} \mathbb{E}[\sum_{t \in T} u_w^{(t)}]$$

Goal: maximize utility

Earnings ratio $F_w = \frac{\sum_{t \in T} u_w^{(t)} / \xi^{(t)}}{\sum_{t \in T} a^{(t)}}$



Use entropy to qualify fairness.

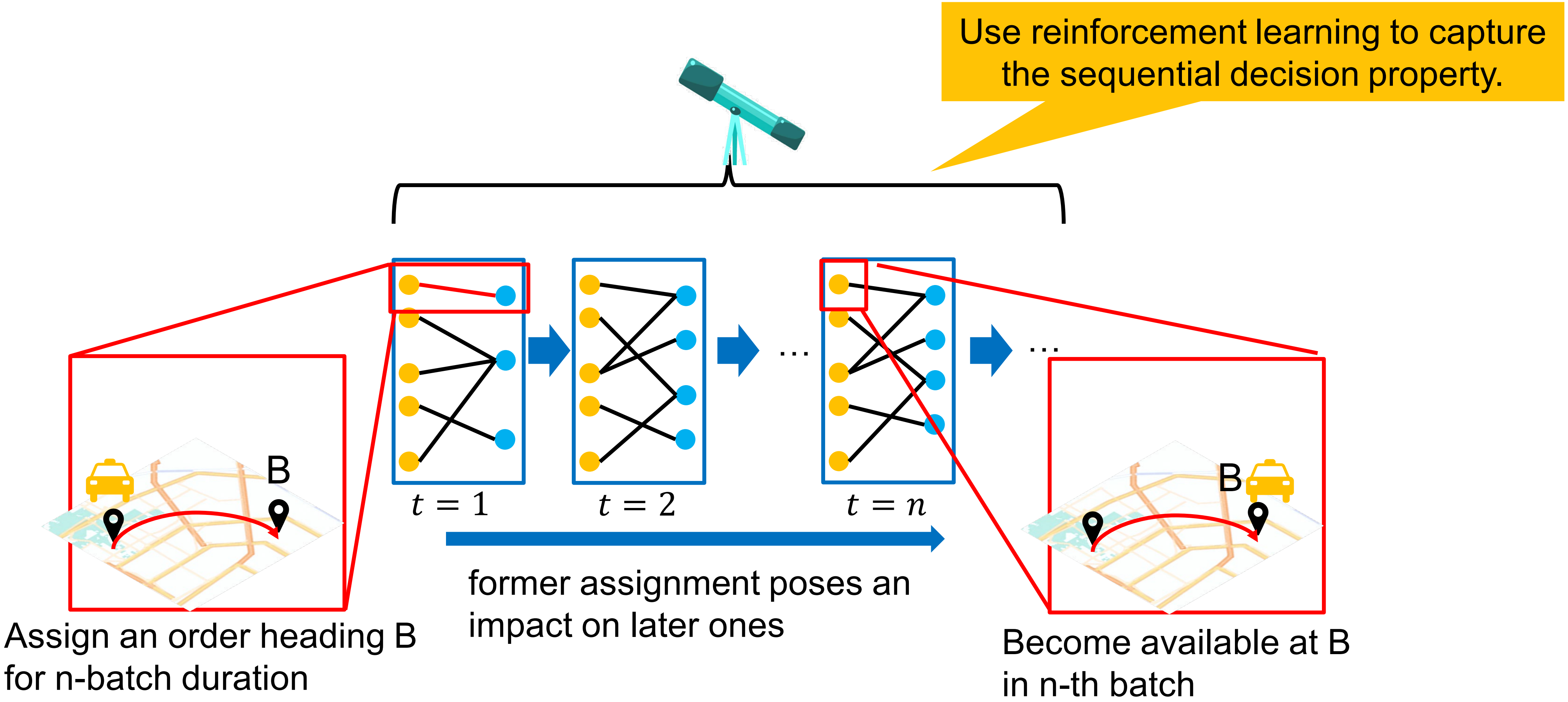
$$F = -\sum_{w \in W} \log\left(\frac{F_w}{\max_{w'} F_{w'}}\right)$$

Constrain: maintain income fairness

Design object

Problem Statement

- **Main challenges**
  - **Temporal dependency between assignments**

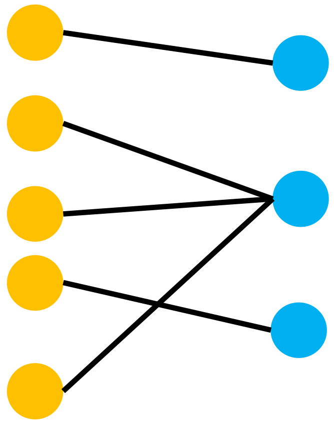


Problem Statement

- **Main challenges**
  - **Inefficient for large scale applications**
    - **Fairness constrain calls for more calculation than normal assignment, which exacerbates time consuming.**

calls for efficient design

Matching with constrain



Goal
Maximize total utility

Constrain
maintain fairness

Problem
Time consuming

Current solutions
Integer linear programming[1]
Reassign[2]

[1] T. Sühr et al, Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform KDD 2019
[2] Lesmana et al, Balancing Efficiency and Fairness in On-Demand Ridesourcing, NeurIPS 2019.

Outline

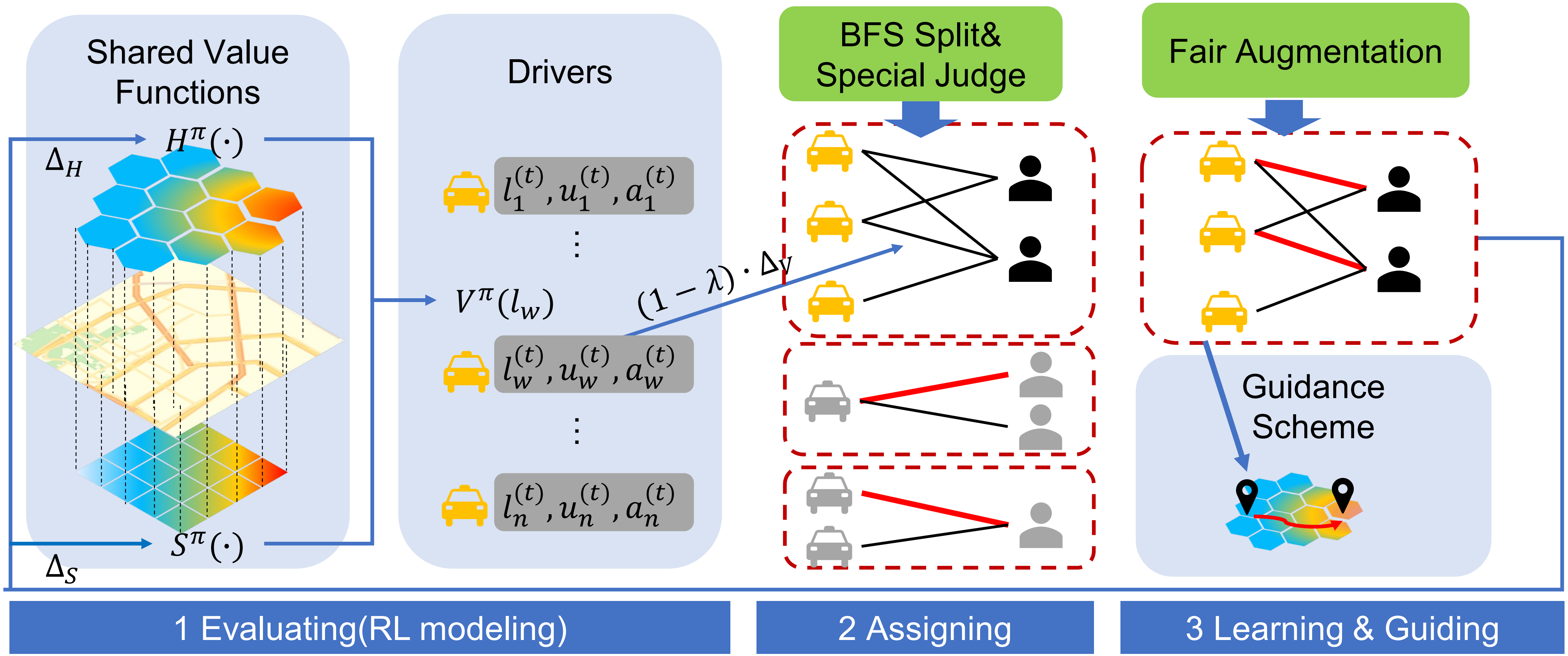
- Background & Motivation
- Problem Statement
- Our Solutions
- Experiments
- Conclusion

Our Solutions

- **LAF System overview**

LEARNING-BASED RE-WEIGHTING

EFFICIENT FAIR ASSIGNMENT

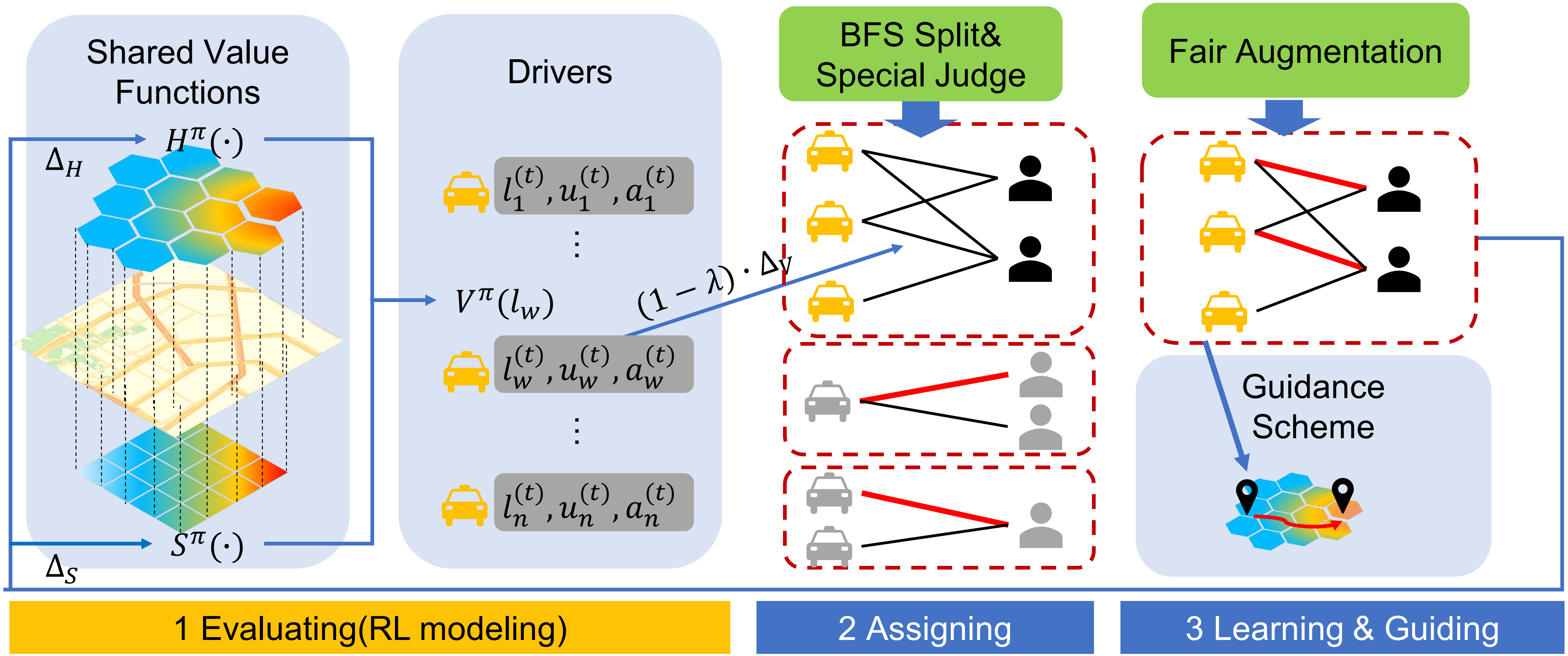


Our Solutions

- **LAF System overview**

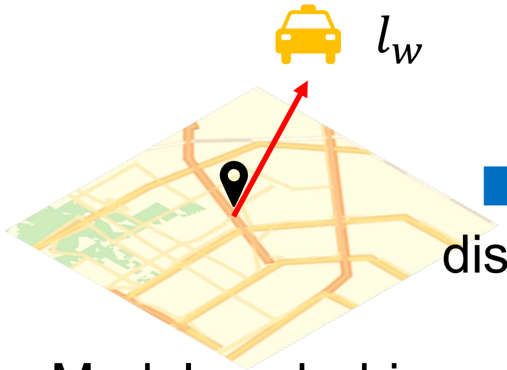
LEARNING-BASED RE-WEIGHTING

EFFICIENT FAIR ASSIGNMENT

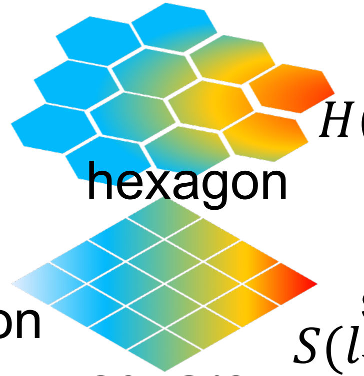


Our Solutions

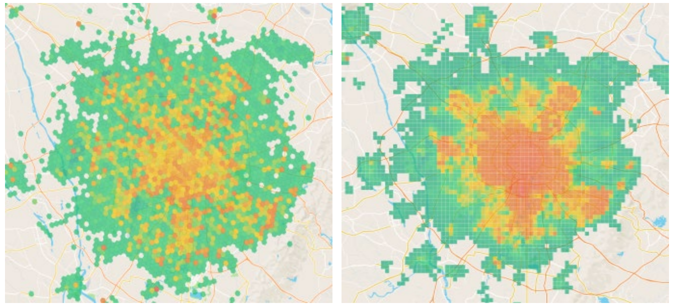
- **Step1: Evaluating(RL modeling)**



discretion



smoothing



Model each driver as an RL agent

Use location as agent state

$$V(l_w) = \sum_x H(l_w + x) + \sum_x S(l_w + x)$$

destination

origin

request $< \boldsymbol{o_r}, \boldsymbol{d_r}, \boldsymbol{p_r}, \boldsymbol{\tau_r} >$

price

duration time

re-weighting

original edge weight: p_r

new edge weight: $p_r + \boxed{\gamma^{\tau_r} V(d_r) - V(l_w)}$

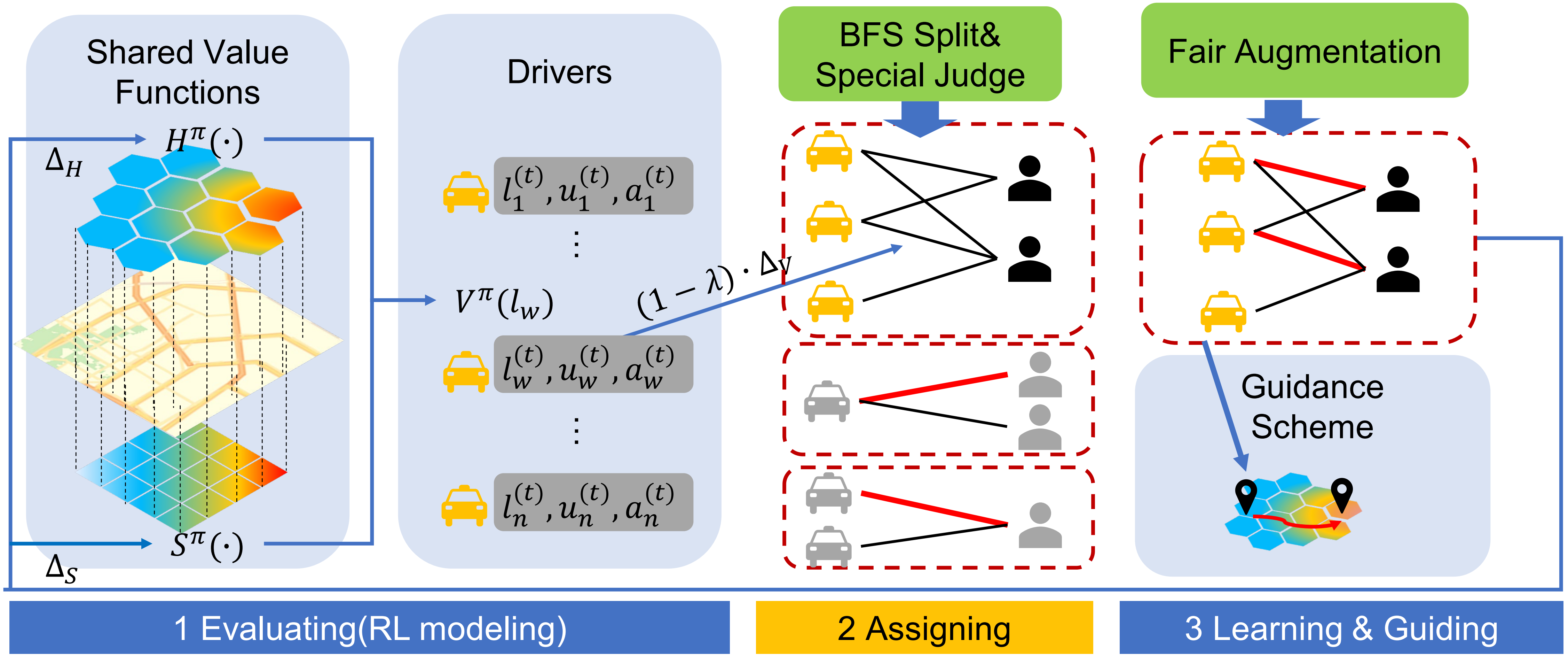
TD Error

TD Error term indicates the future expected reward.

Our Solutions

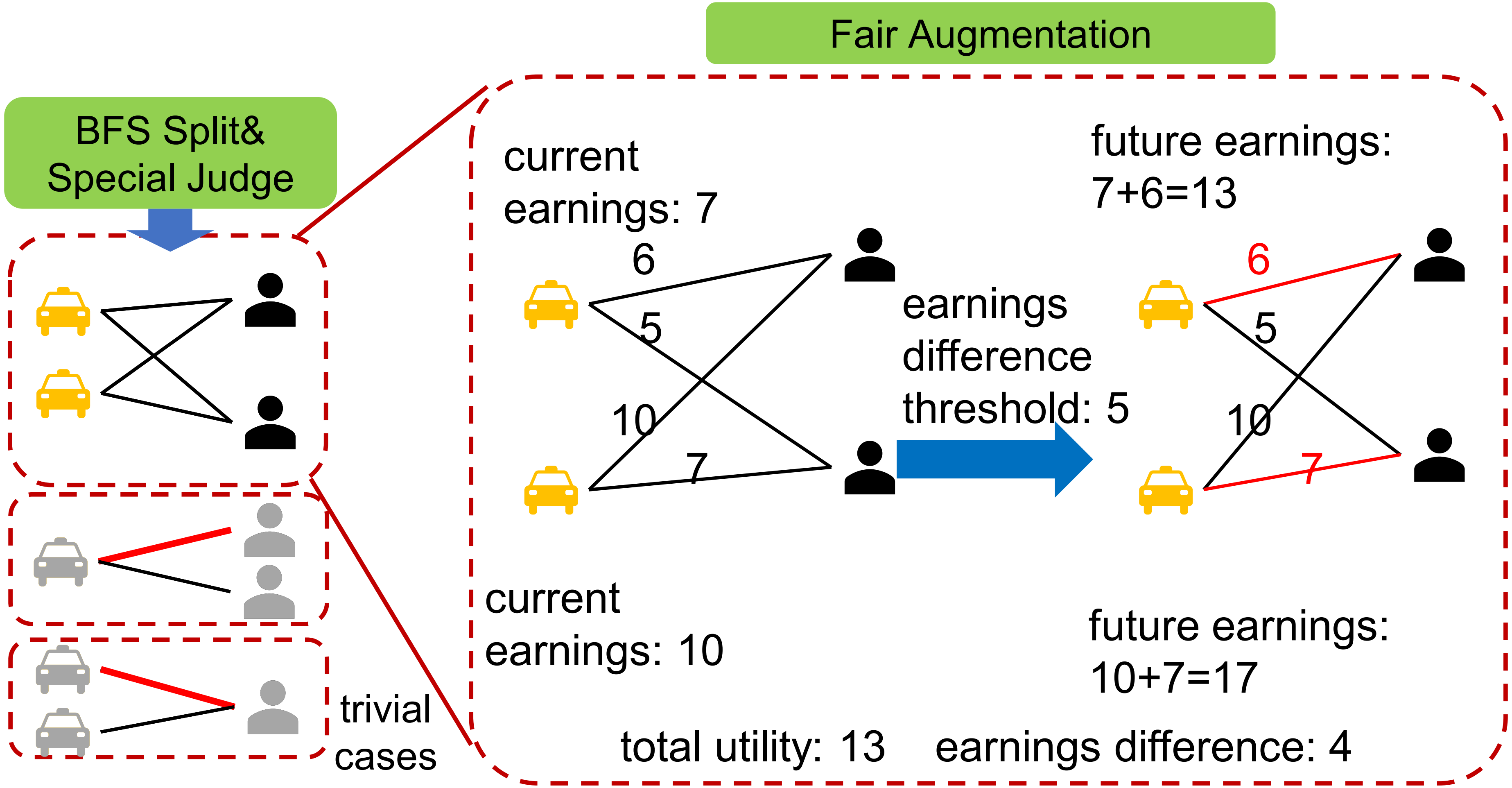
- **LAF System overview**

LEARNING-BASED RE-WEIGHTING   EFFICIENT FAIR ASSIGNMENT



Our Solutions

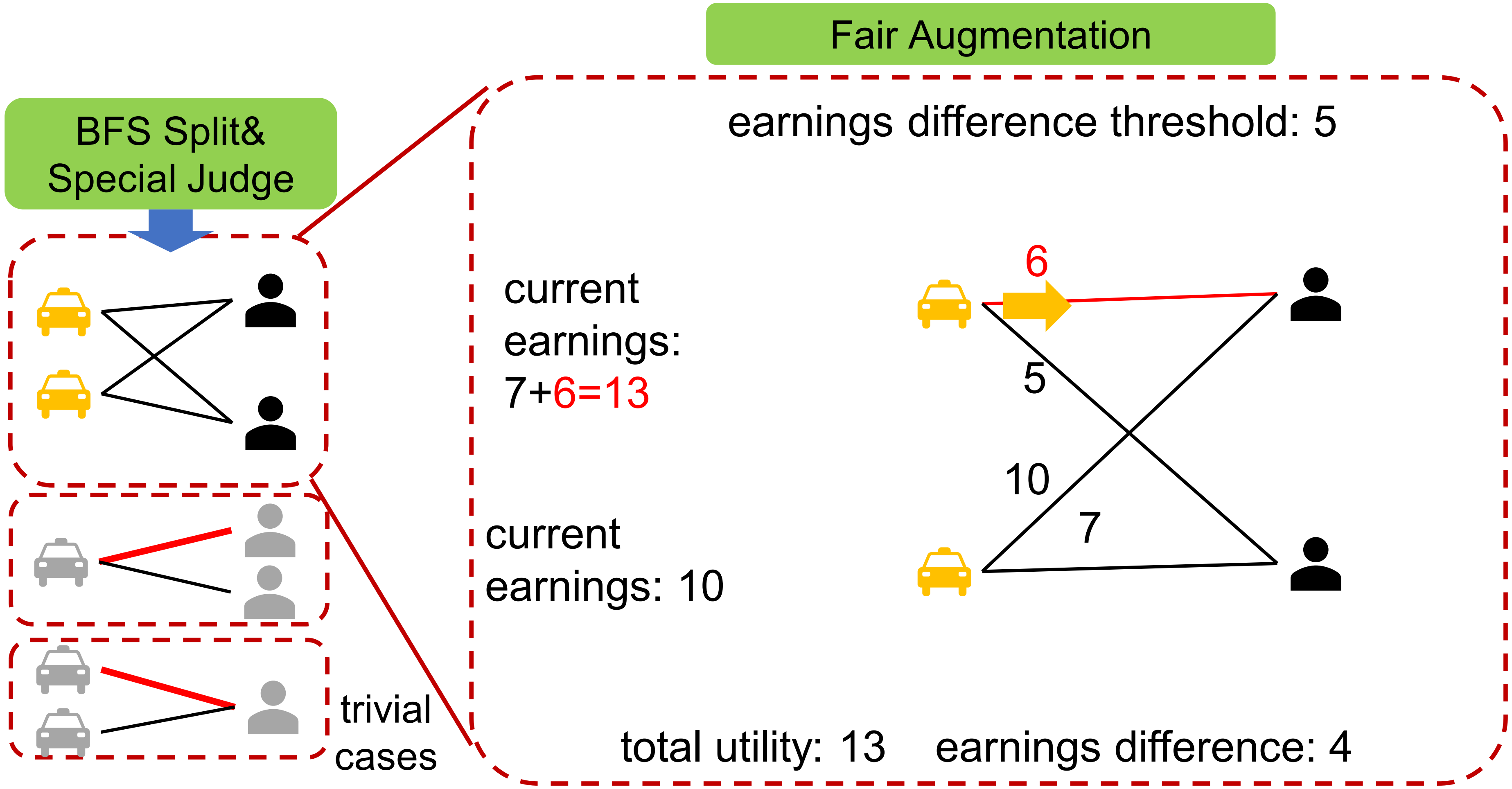
- **Step2: Assigning**



Check earnings difference within threshold while augmentation.

Our Solutions

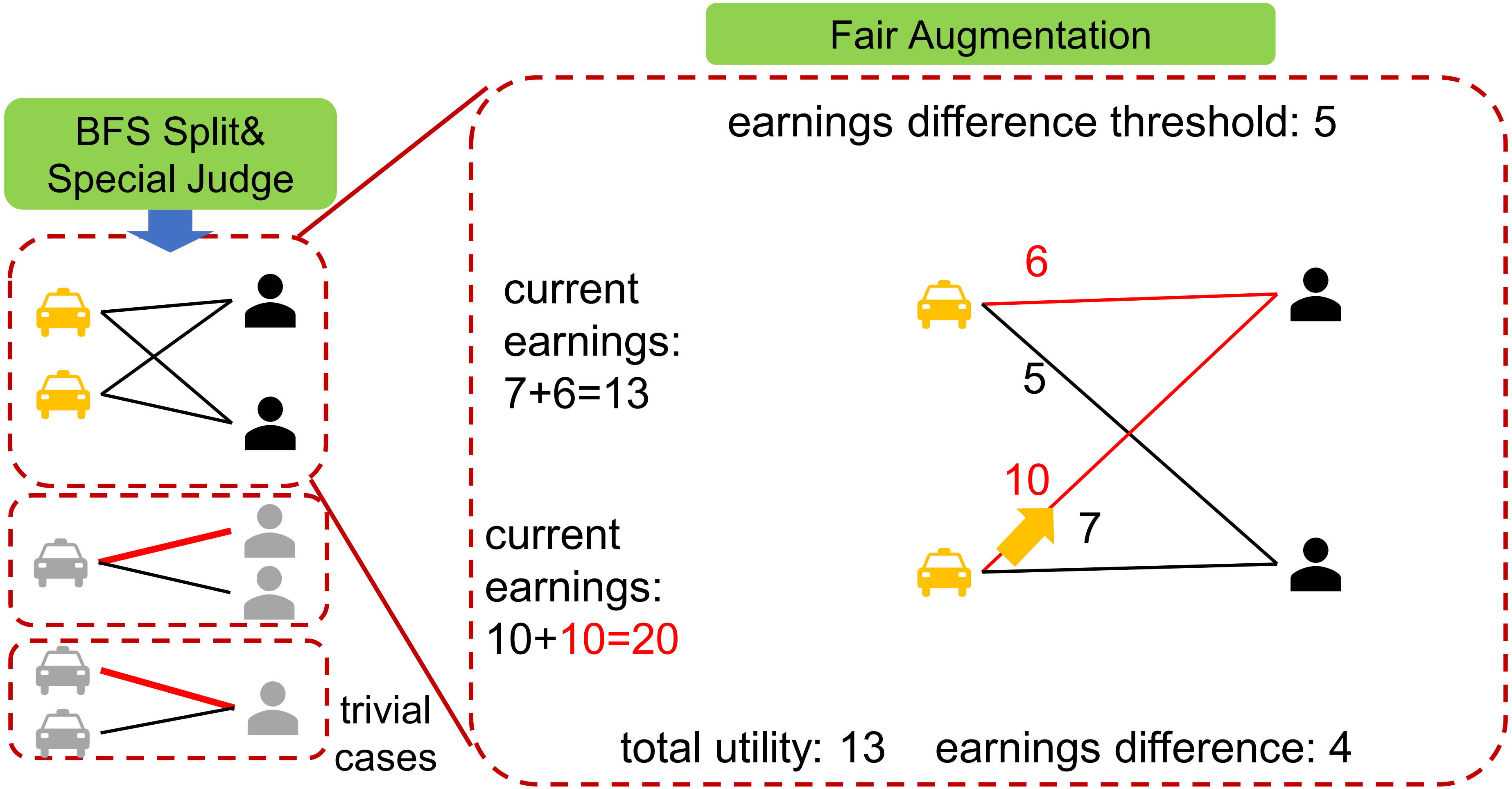
- **Step2: Assigning**



Check earnings difference within threshold while augmentation.

Our Solutions

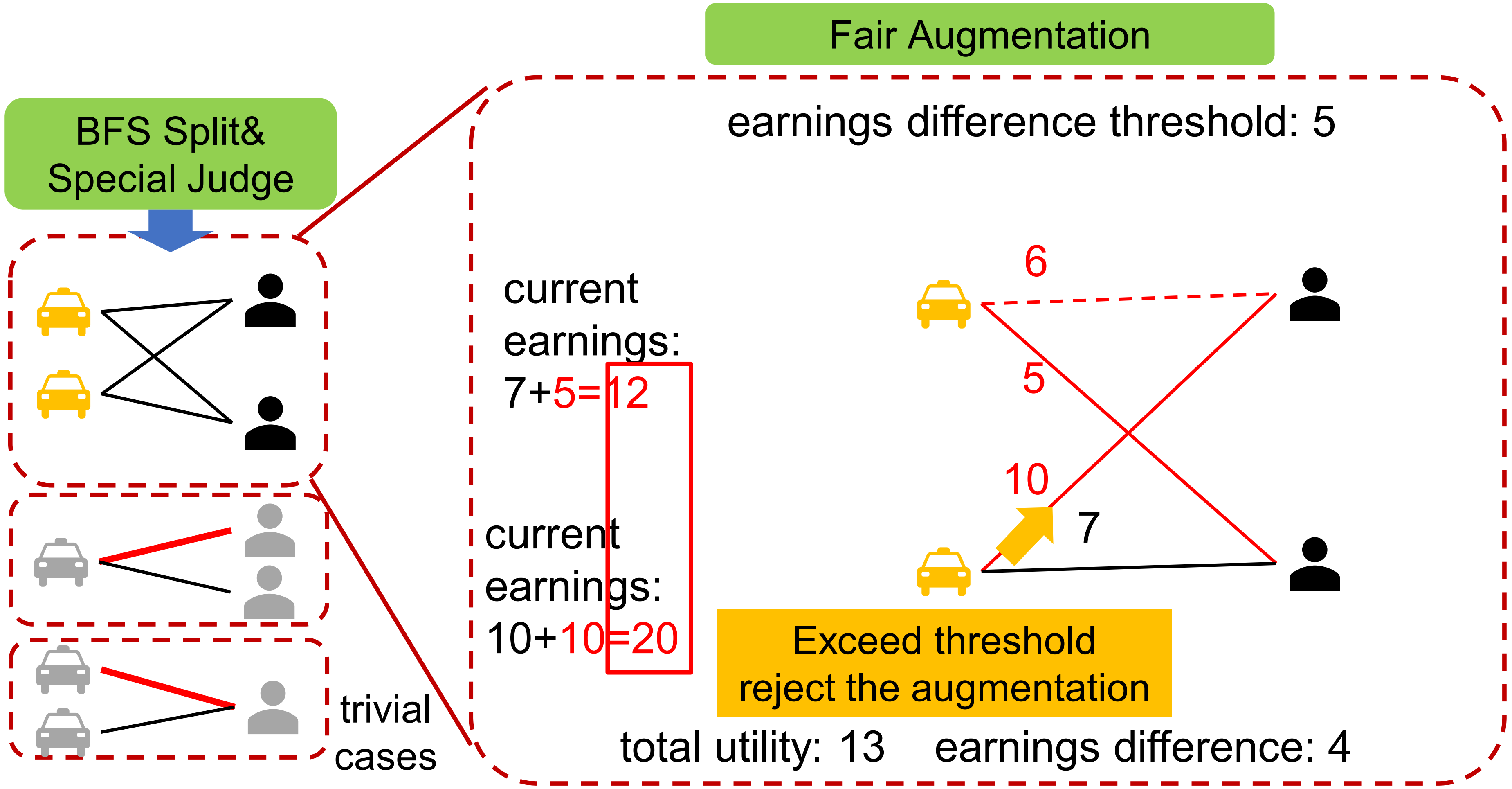
- **Step2: Assigning**



Check earnings difference within threshold while augmentation.

Our Solutions

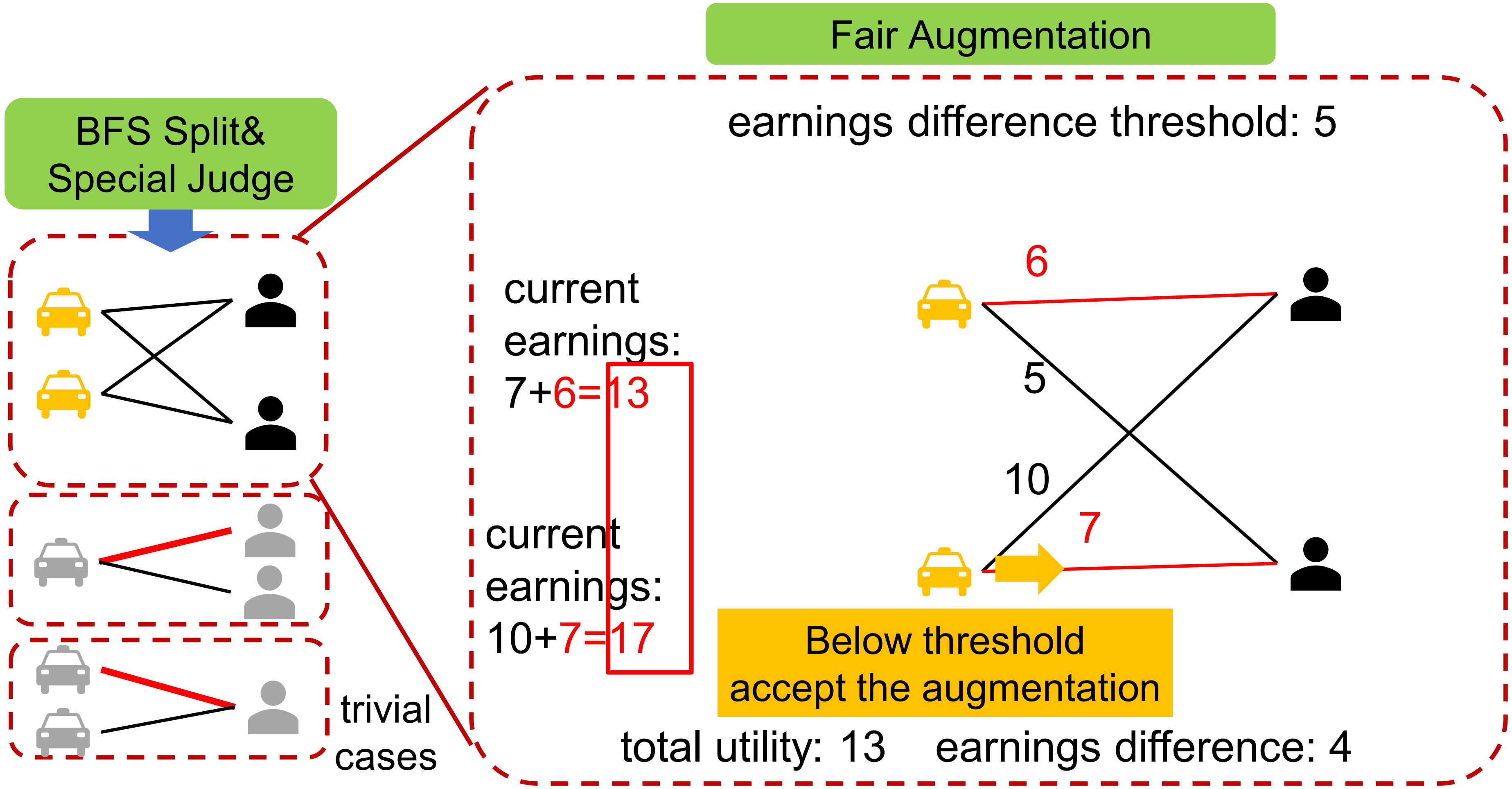
- **Step2: Assigning**



Check earnings difference within threshold while augmentation.

Our Solutions

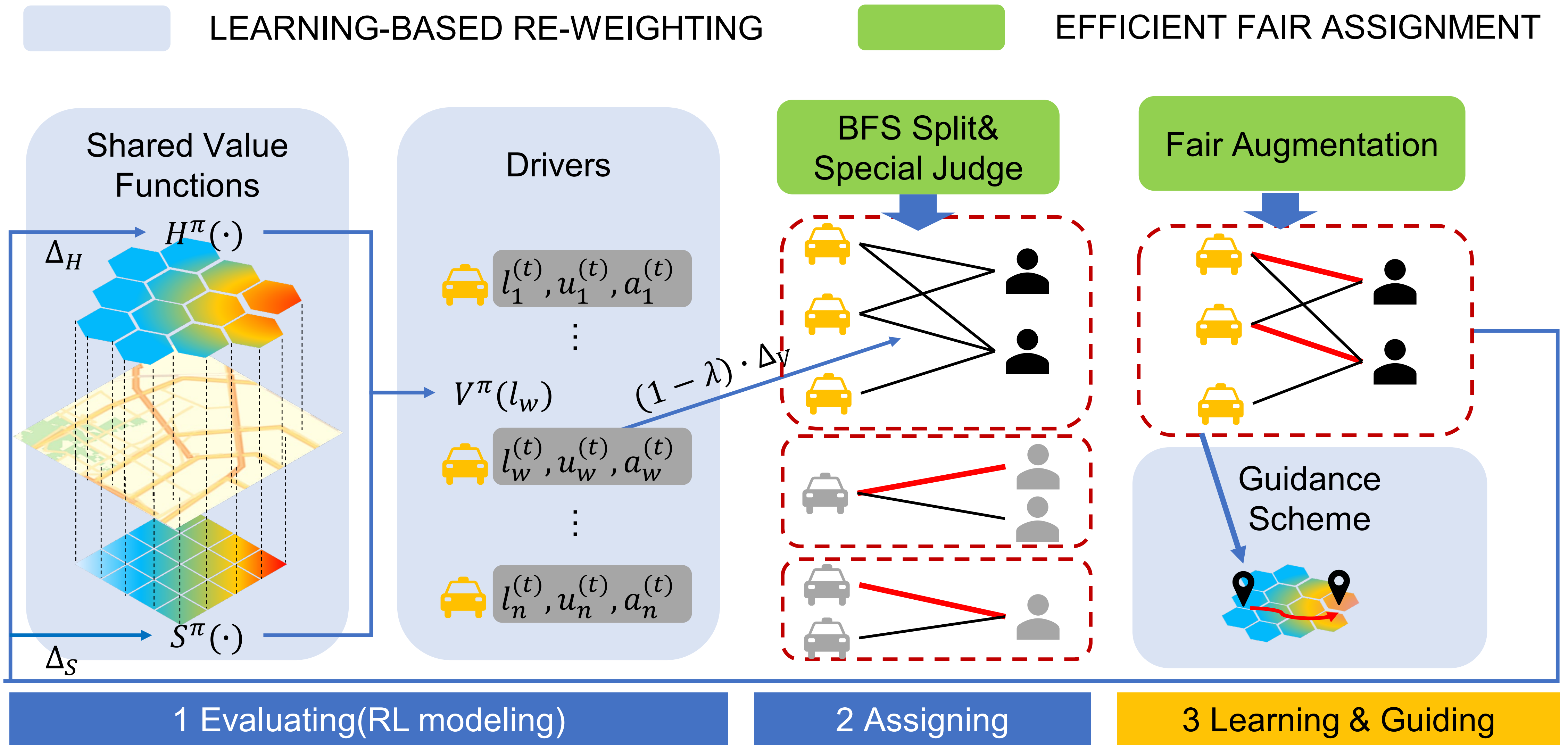
- **Step2: Assigning**



Check earnings difference within threshold while augmentation.

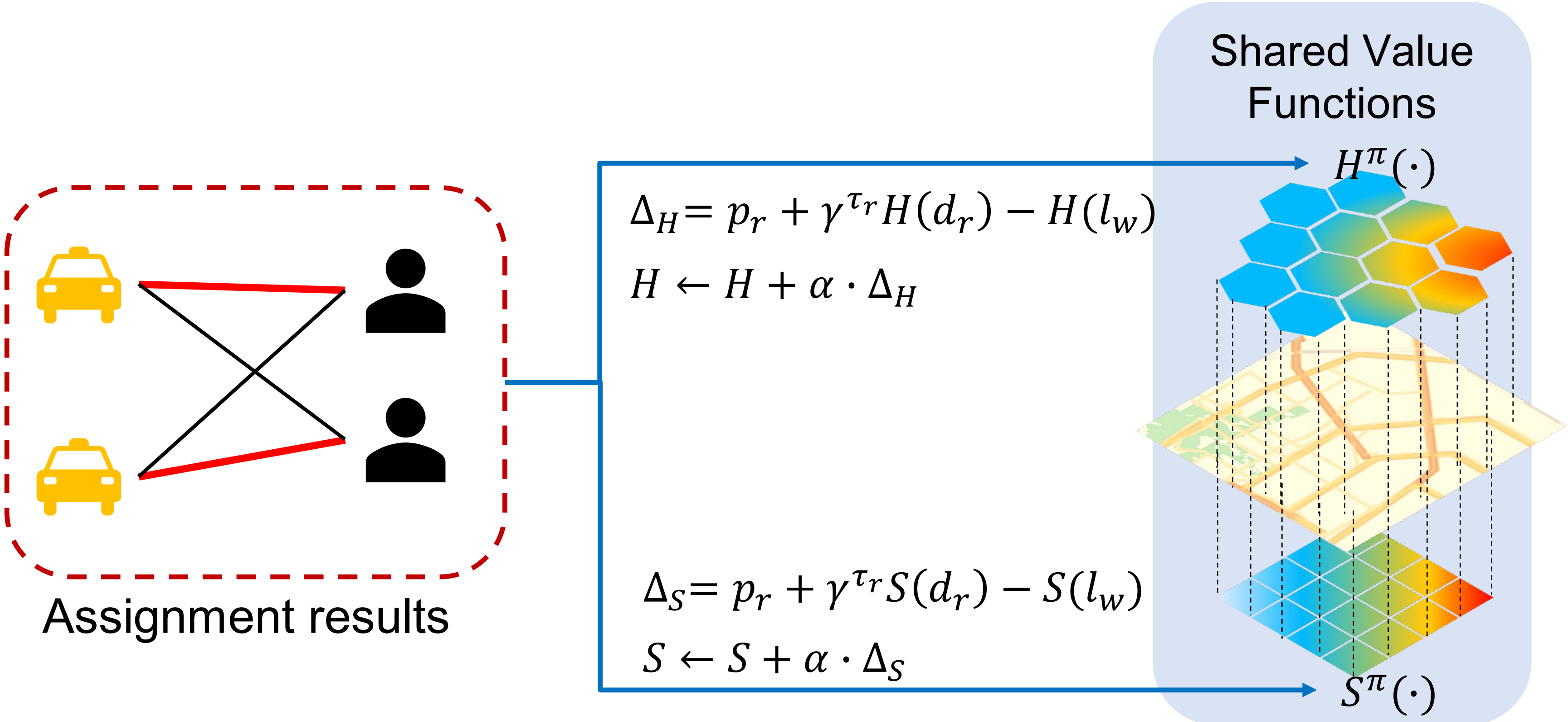
Our Solutions

- **LAF System overview**



Our Solutions

- **Step3: Learning & Guiding**
  - **Adopt online learning manner to avoid discrepancy between offline policy and online evaluation**



TD(0) Learning: update value function by TD error.

Our Solutions

- **Step3: Learning & Guiding**
  - **Online learning manner brings cold start issue**
  - **Assignment algorithm will not work without requests**

can achieve fairness by task assignment

cannot achieve fairness by task assignment

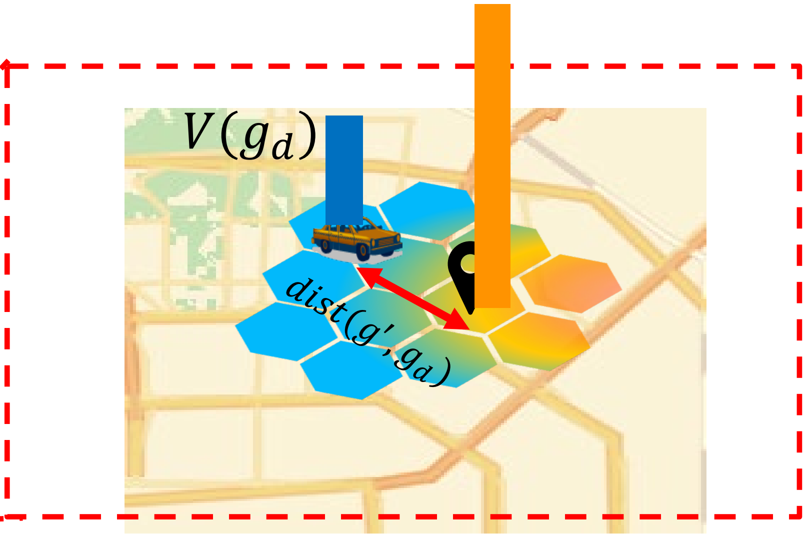
Our Solutions

- **Step3: Learning & Guiding**
  - **Online learning manner brings cold start correction**
  - **Assignment algorithm will not work without requests**

$$g \leftarrow \arg\max_{g'} \frac{V(g') - V(g_d)}{dist(g', g_d)}$$

Value increment

Guiding distance



Guide drivers to suitable area based on both value and distance.

Outline

- Background & Motivation
- Problem Statement
- Our Solutions
- Experiments
- Conclusion

Experiments

- **Validation Environment**
  - **Dataset**
    - **Real order and driver data on three cities in China**
  - **Simulator**
    - **Developed by a major ride hailing platform**
    - **Builds bipartite, simulates drivers moving, executes algorithm**
- **Running Information**
  - **CPU: Intel Xeon E5-2630 v4 @ 2.20 GHz**
  - **Memory: 12GB**
- **Parameter Setting**
  - **Discount factor:** $\boldsymbol{\gamma = 0.9}$
  - **Learning rate:** $\boldsymbol{\beta = 0.025}$
  - **Batch size: 2s**

Experiments

- **Comparing methods**
  - DG: assigns driver the nearest request regardless of fairness
  - ERG: assigns lowest earnings ratio driver the best request
  - ILP[1]: integer linear programming based solution
  - REA[2]: reassigns matchings to make trade-off

- **Evaluation metrics**
  - Temporal Fairness
  - Total Utility
  - Time Consuming

[1] T. Sühr et al, Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform KDD 2019
[2] Lesmana et al, Balancing Efficiency and Fairness in On-Demand Ridesourcing, NeurIPS 2019.

Experiments

- **Fairness Comparison**

$$F = -\sum_{w \in W} \log\left(\frac{F_w}{\max_{w'} F_{w'}}\right)$$

- **Lower values mean better fairness**

	City A		City B		City C	
	weekday	weekend	weekday	weekend	weekday	weekend
DG	41,767	31,962	31,217	28,563	12,897	9,044
ERG	48,549	35,903	38,594	37,464	13,704	8,793
ILP	44,072	30,573	29,496	25,077	12,377	8,439
REA	44,251	27,865	31,744	33,671	13,491	8,684
LAF	22,656	13,384	7,420	4,976	2,553	2,392

Our LAF performs best among all cities on both weekdays and weekends.

Experiments

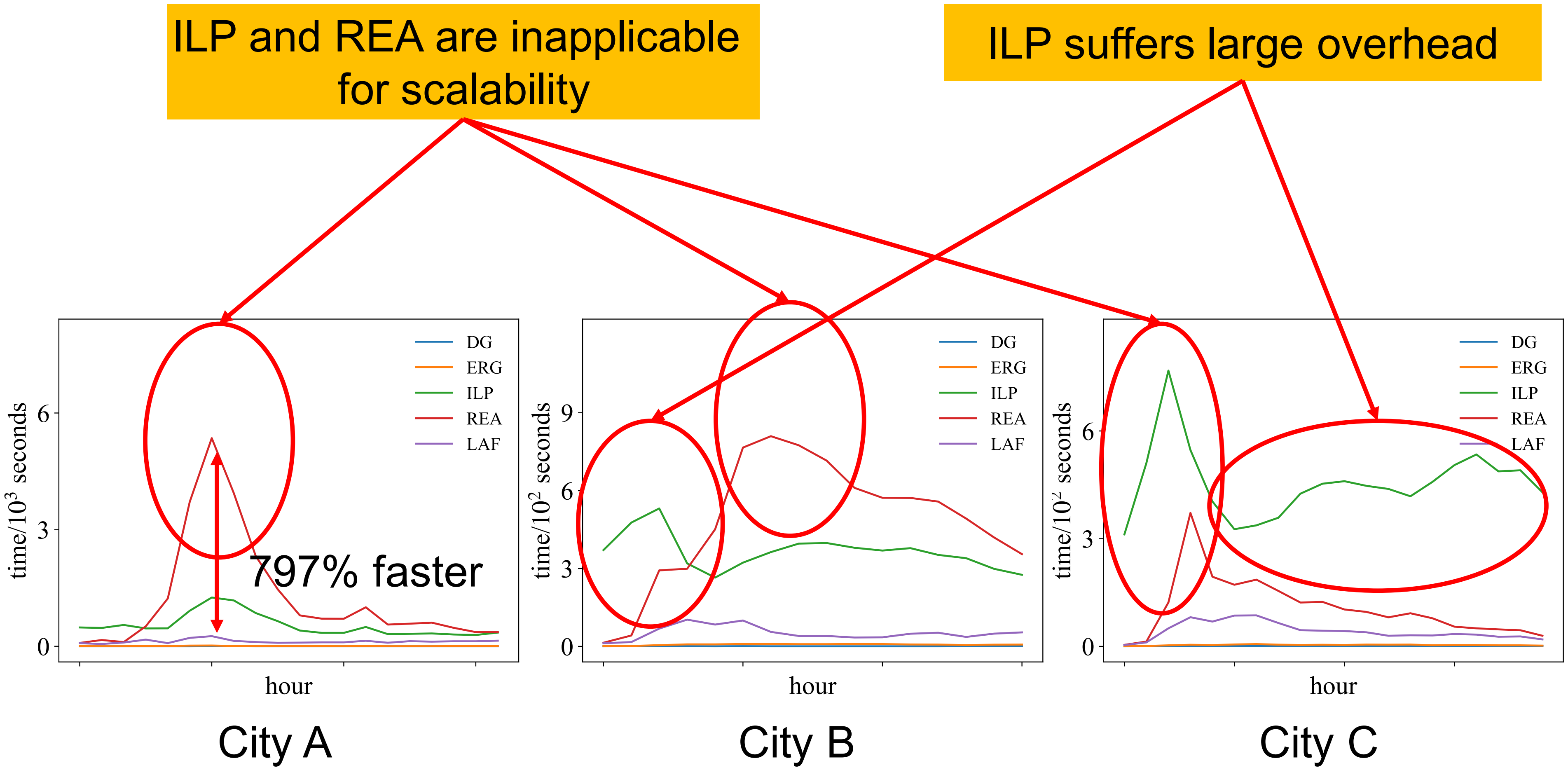
- **Utility Comparison**
- **Higher values mean better utility**

	City A		City B		City C	
	weekday	weekend	weekday	weekend	weekday	weekend
DG	2,242,977	2,131,996	1,297,735	1,360,659	872,693	865,031
ERG	2,256,277	2,172,489	1,373,478	1,452,488	885,558	882,573
ILP	2,246,077	2,122,834	1,281,273	1,336,712	862,588	863,245
REA	2,250,118	2,134,534	1,319,958	1,373,691	880,847	881,519
LAF	2,656,773	2,565,060	1,479,348	1,616,385	1,114,000	1,109,474

Our LAF performs best among all cities on both weekdays and weekends.

Experiments

- **Efficiency Comparison**

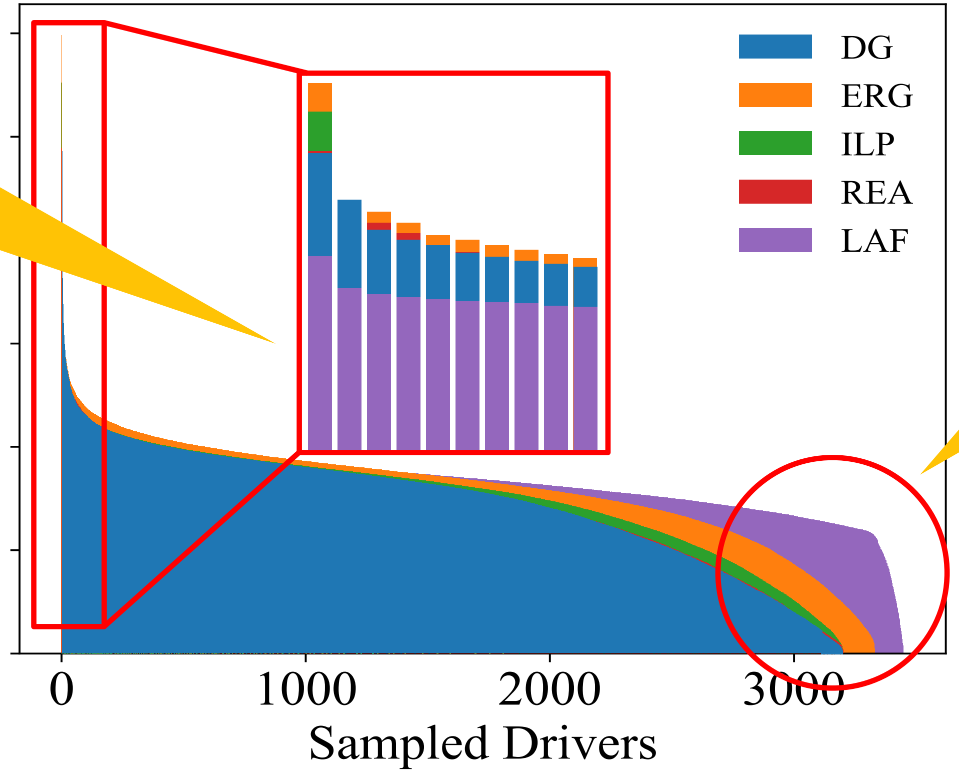


Our LAF shows a high execution efficiency.

Experiments

- **Case Study**
  - **sample and sort drivers by earnings ratio**

1. Compared with other algorithms, our algorithm avoid exceedingly high earnings ratio

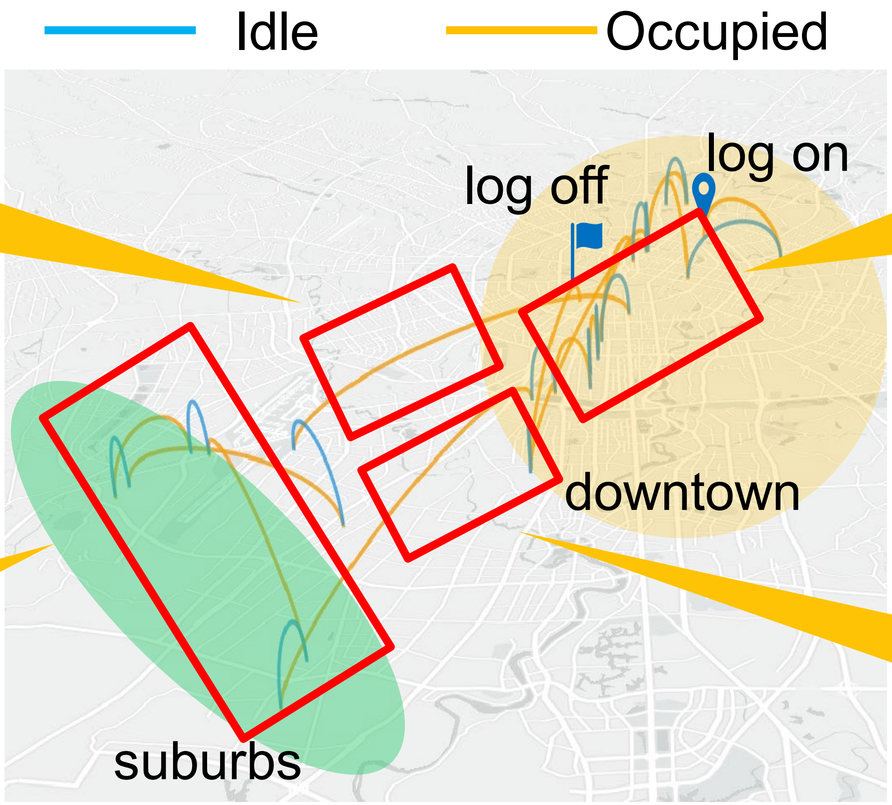


2. Compared with other algorithms, our algorithm avoid exceedingly low earnings ratio

Avoid exceedingly high or low earnings ratio

Experiments

- **Case Study**
  - **Analysis on driver trajectories**



4. Guide driver back to downtown timely by assignment and scheduling

1. Initially log on, serve within downtown to improve earnings ratio

3. Improve serving rate by scheduling

2. When earnings ratio improved, assign suburban orders

Improve earnings ratio via scheduling

Outline

- Background & Motivation
- Problem Statement
- Our Solutions
- Experiments
- Conclusion

Conclusion

- We propose a reinforcement learning based approach for task assignment, aiming to achieve both high utility and temporal fairness.
- We devise an efficient matching algorithm which executes augmentation while checking fairness.
- Experiments on real history data validate the performances on fairness, utility and efficiency.

Q & A

Thank You