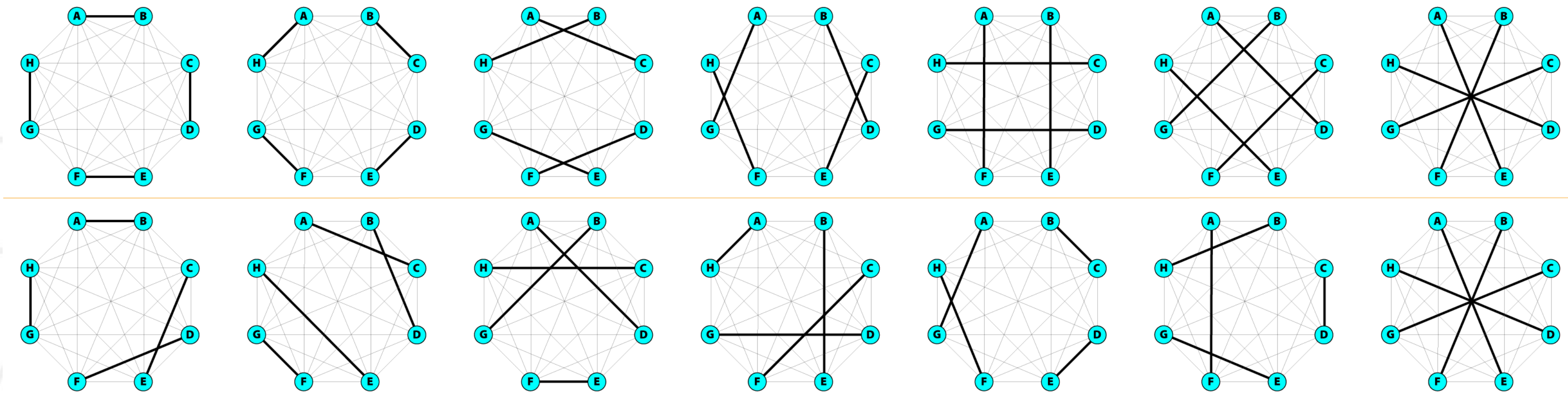
Batching and Greedy Policies in Dynamic Matching

Alejandro Toriello
joint with Myungeun Eom

University of Florida ISE
August 25, 2023

Georgia Tech

Matching



Source: math.stackexchange

- Matching: No two edges share a node.
- Max-weight matching can be solved efficiently (Edmonds, 65).
- Myriad applications…

Dynamic Matching

- Nodes, edges may arrive/depart over time.
  - Must match sequentially, with partial information.

- Bipartite graph: Models supply and demand in dynamic markets
  - E.g. ride-hailing, online ads, gig economy

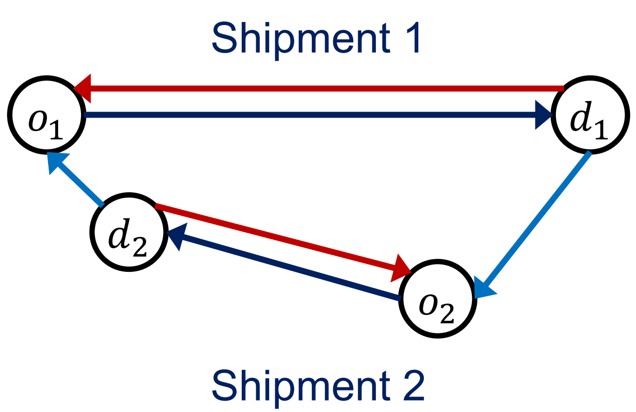
- Non-bipartite: Platform matches agents
  - E.g. organ exchanges, transportation marketplaces, ride-sharing

Georgia Tech

Motivation: Freight transportation

- Shipment requests dynamically appear over time
- A broker platform either assigns or auctions these requests to carriers
- Carriers (usually truck drivers) want to avoid deadhead

- The platform bundles requests to reduce the deadhead distance



Matching reward for a request pair :
Deadhead distance saved

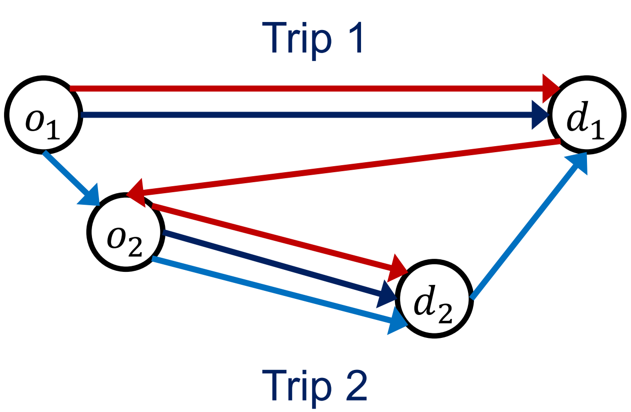
$$d(d_1, o_1) + d(d_2, o_2) - d(d_1, o_2) - d(d_2, o_1)$$

Georgia Tech

Motivation: Ride-Sharing

- Trip requests dynamically appear over time
- Customers may be willing to share rides for a lower fare

- The platform pairs requests to reduce the travel distance



Matching reward for a request pair :
Travel distance saved

$$d(o_1, d_1) + d(d_1, o_2) - d(o_1, o_2) - d(d_2, d_1)$$

Motivation: Two widely used policies

Batching policy

- Accumulate a batch of arriving nodes within a fixed period and optimize matches over each such batch

Greedy policy

- Make matching decisions as opportunities arise by optimizing over available nodes

In spite of their prevalence, there have been a limited number of studies analyzing the performance of these policies

- Anderson et al. (2017)
- Aouad and Saritac (2022)
- Ashlagi et al. (2022)

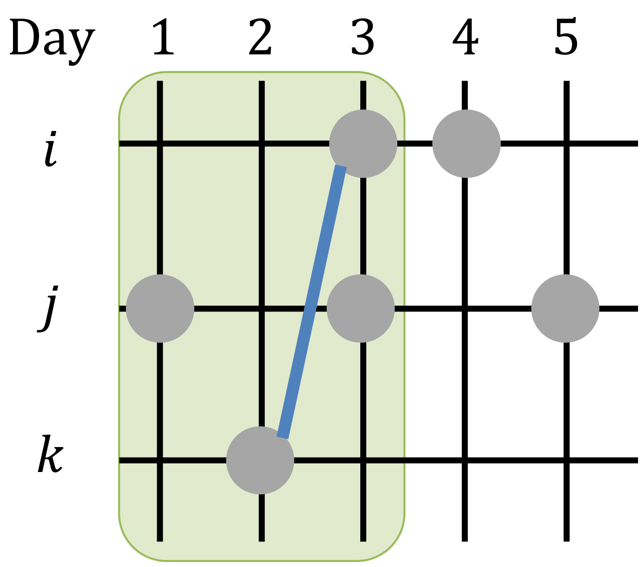
Problem description

Assumptions & Notations

- Node types $N = \{1, \ldots, n\}$
- Each period: Bernoulli arrivals, $p_i \in (0,1]$ for $i \in N$
- Sojourn time $\tau \in \mathbb{N}$
- Match reward $w_{ij} \geq 0$
- Objective : maximize long-run average reward

$N = \{i, j, k\}$

$\tau = 3$

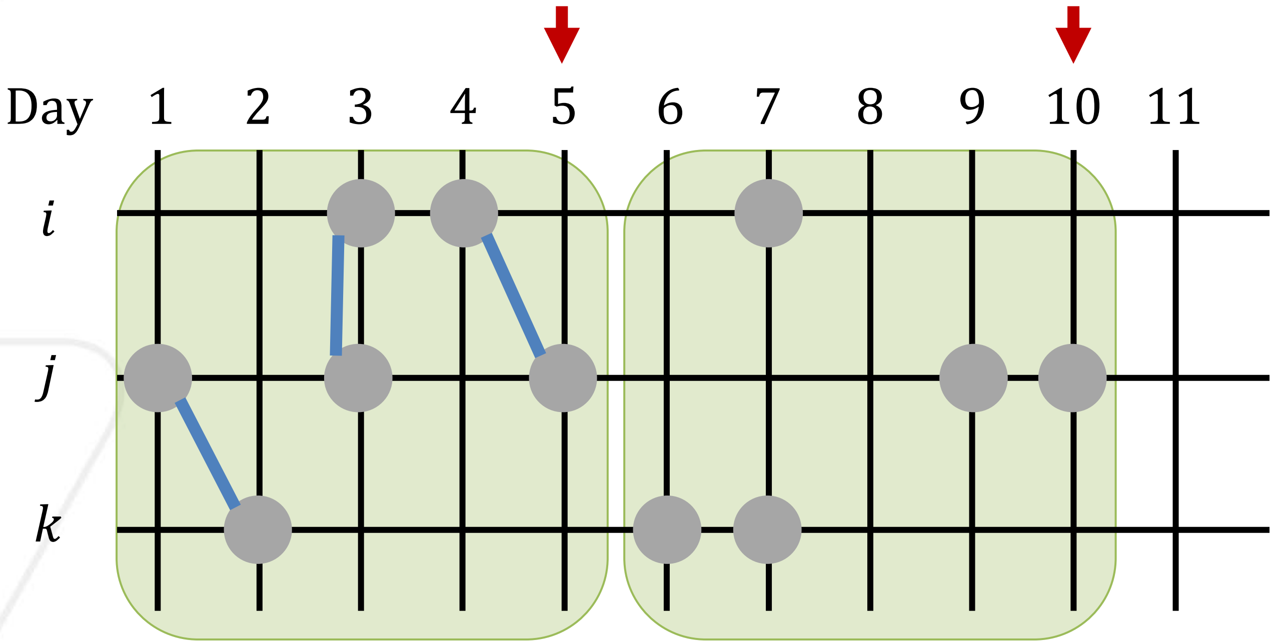


Current day
3 4

Batching and Greedy policies

Batching policy solves a max-reward matching problem and clears out the system every τ periods

(Randomized) Greedy policy solves a max-reward matching problem every period (with a restricted set of nodes)



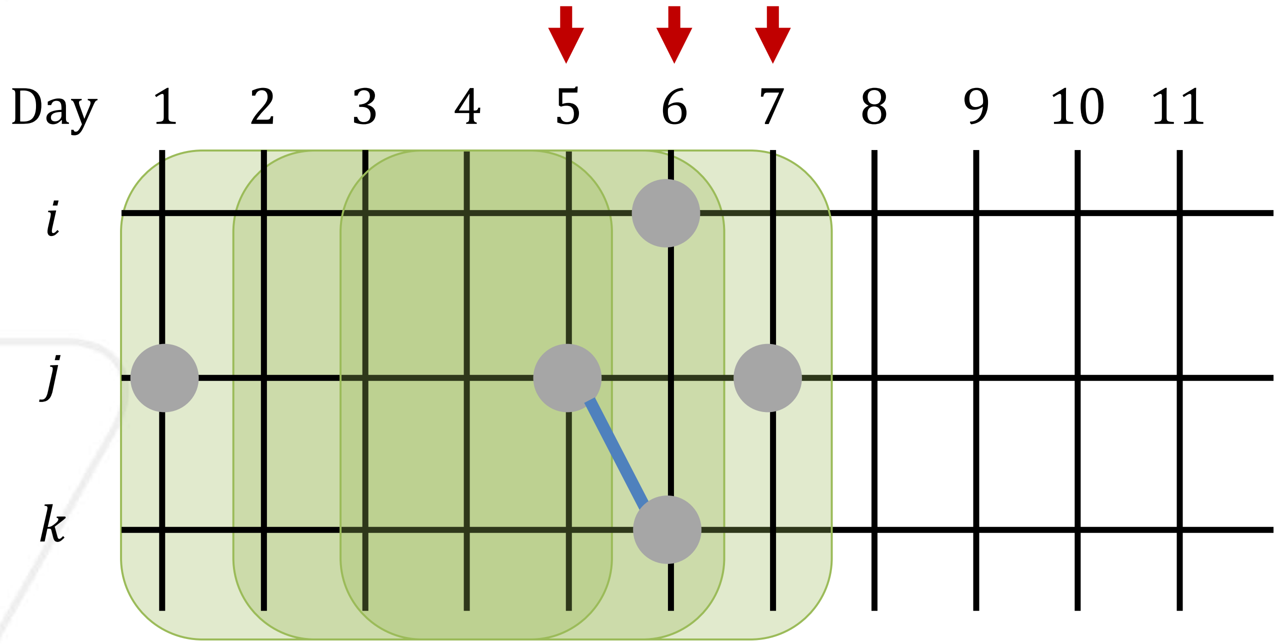
$N = \{i, j, k\}$

$\tau = 5$

Batching and Greedy policies

Batching policy solves a max-reward matching problem and clears out the system every τ periods

(Randomized) Greedy policy solves a max-reward matching problem every period (with a restricted set of nodes)



$N = \{i, j, k\}$

$\tau = 5$

Georgia Tech

Batching and Greedy policies

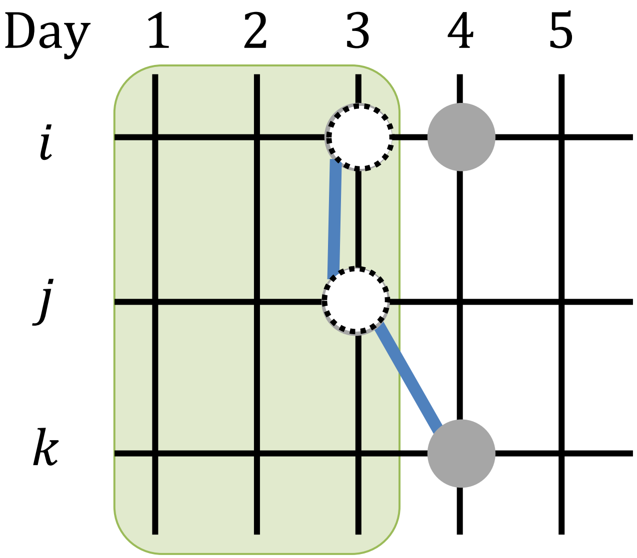
Batching policy solves a max-reward matching problem and clears out the system every τ periods

(Randomized) Greedy policy solves a max-reward matching problem every period (with a restricted set of nodes)

$$N = \{i, j, k\}, \tau = 3$$

$$w_{ij} = w_{ik} = \delta, w_{jk} = 1$$

$$p_j = p_k = p < 1, p_i = 1$$



Main Results

Asymptotic optimality

- $H(\tau)$: average reward of policy H with sojourn τ

$B(\tau) \geq OPT(\tau) - \mathcal{O}(1/\sqrt{\tau})$ $B(\tau) \geq (1-\epsilon)OPT(\tau) - \mathcal{O}(e^{-\tau\epsilon}), \quad \epsilon > 0$ $G(\tau) \geq OPT(\tau) - \mathcal{O}(1/\tau)$ $G(\tau) \geq (1-\epsilon)OPT(\tau) - \mathcal{O}((1+\epsilon)^{-\tau}), \quad \epsilon > 0$
$B(\tau, I) \leq G(\tau, I) - \Omega(1/\sqrt{\tau})$

- Assumption: bounded support

Main Results

Impatient nodes

- Unmatched node i abandons the system w.p. $d_i(\tau)$ in each period

Policy	Assumption
Batching	$d_i(\tau) = \mathcal{O}(\tau^{-\beta})$ for $\beta > 1$, $i \in N$
Randomized Greedy	$d_i(\tau) = o(1)$, $i \in N$

Batching vs. Randomized Greedy

- Batching – oblivious to distributions, more vulnerable to impatience
- Randomized Greedy – requires expectations, allows more impatience, better asymptotic performance

Analysis - Roadmap

- Analysis for a single pair
- Analysis for the general case
  - Randomization
  - Policy comparison
- Extension to impatient setting

Analysis for a single pair $\{i, j\}$

Batching policy

- Matching frequency is $\mathbb{E}[\min\{A_i, A_j\}]/\tau$
- A_i is number of type i arrivals in τ periods, $(A_i \sim Bin(\tau, p_i))$

$$\underbrace{\min\{p_i, p_j\}}_{\text{Fluid matching frequency}} - \underbrace{\mathbb{E}[\min\{A_i, A_j\}]/\tau}_{\text{Batching policy frequency}} = \begin{cases} \mathcal{O}(1/\sqrt{\tau}) & \text{if } p_i = p_j \\ \mathcal{O}(e^{-\tau(p_j - p_i)}) & \text{if } p_i < p_j \end{cases}$$

- Proof uses basic probability inequalities
- If $p_i < (1-\epsilon)p_j$, then $\mathcal{O}(e^{-\tau\epsilon})$, else, discard type i arrivals w.p. ϵ

Analysis for a single pair $\{i, j\}$

Greedy policy

- Discrete-time Markov chain

State S

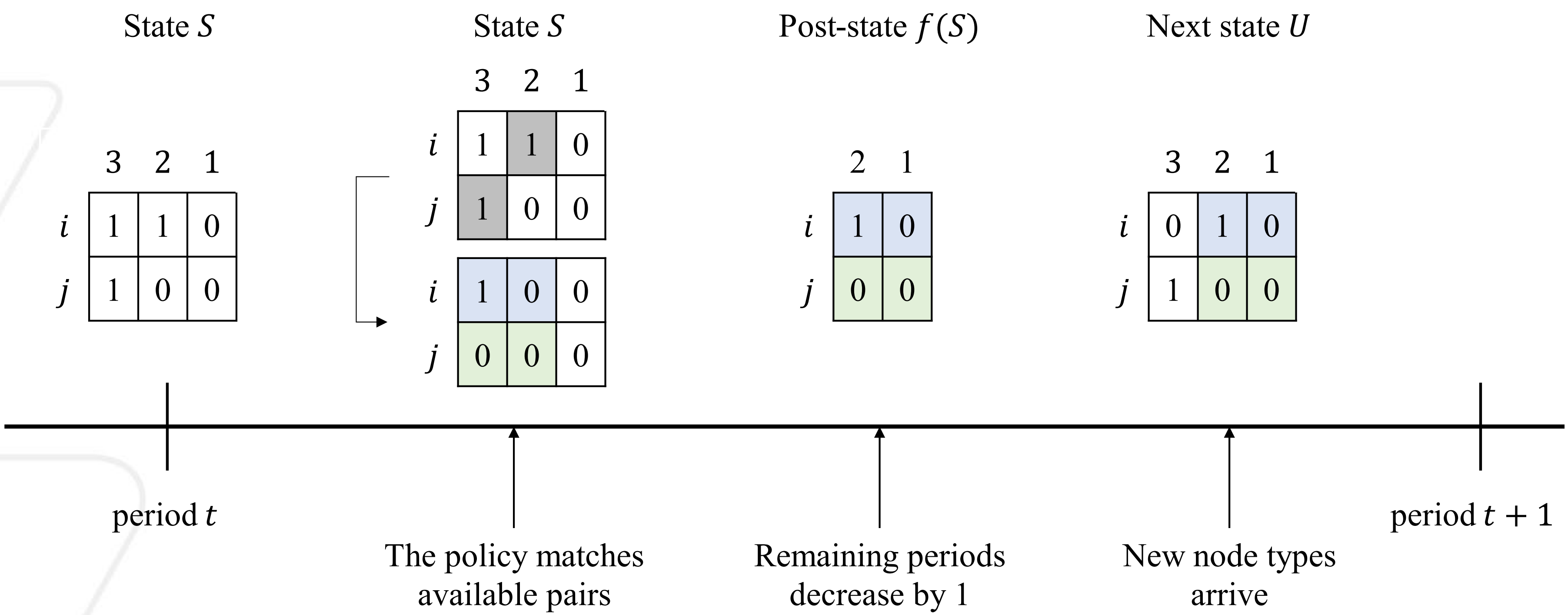
remaining day	3	2	1
i	1	1	0
j	1	0	0

$\alpha(S) = 1$

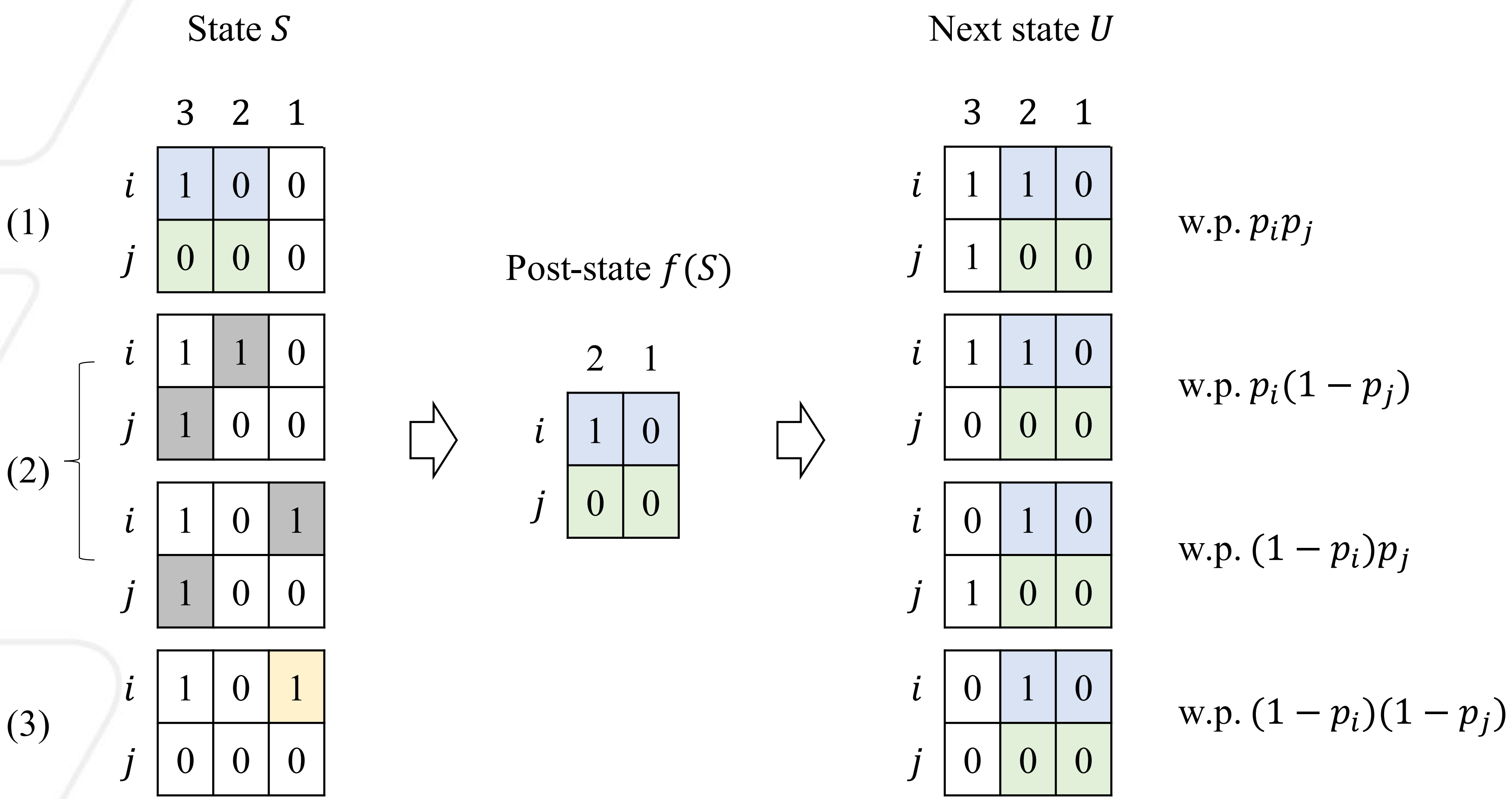
- Chain is ergodic with unique stationary distribution π
- Matching frequency is $\sum_S \pi(S)\alpha(S)$
- $\alpha(S)$ is number of matches in state S following greedy policy

Analysis for a single pair $\{i, j\}$

Transition process



Analysis for a single pair $\{i, j\}$



Analysis for a single pair $\{i, j\}$

The long-run average matching frequency is

$$\begin{cases} p - \frac{p(1-p)}{1+2(\tau-1)p} & if\ p_i = p_j = p \\ p_i - \frac{(p_j-p_i)p_i(1-p_i)}{p_j(1-p_j)q^{-2\tau}-p_i(1-p_i)} & if\ p_i < p_j, q = \frac{1-p_j}{1-p_i} < 1 \end{cases}$$

- Proof relies on solving balance equations in Markov chain
- If $p_i < (1-\epsilon)p_j$, then $\mathcal{O}((1+\epsilon)^{-\tau})$, else, discard type i arrivals w.p. ϵ

Fluid LP Relaxation

$$\max_{z \geq 0} \sum_{i,j \in N} w_{ij} z_{ij} : \sum_{j \in N \setminus i} z_{ij} \leq p_i \,, \forall i \in N$$

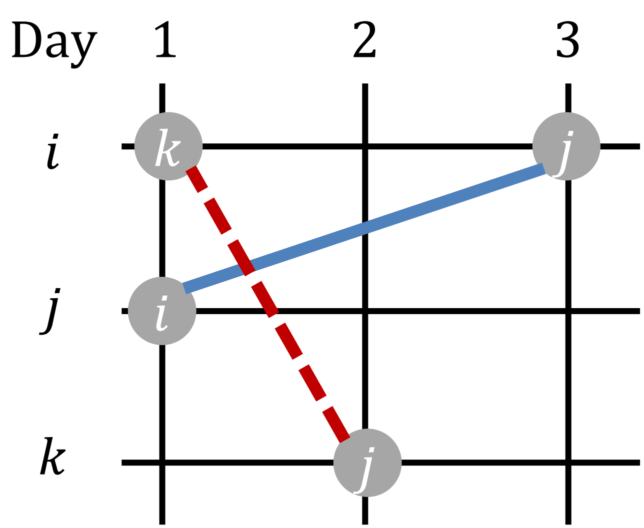
z_{ij} : average matching frequency of a pair $\{i, j\}$

Analysis for the general case

Randomization

$$\max_{z\geq 0} \sum_{i,j\in N} w_{ij} z_{ij} : \sum_{j\in N\setminus i} z_{ij} \leq p_i, \forall i \in N$$

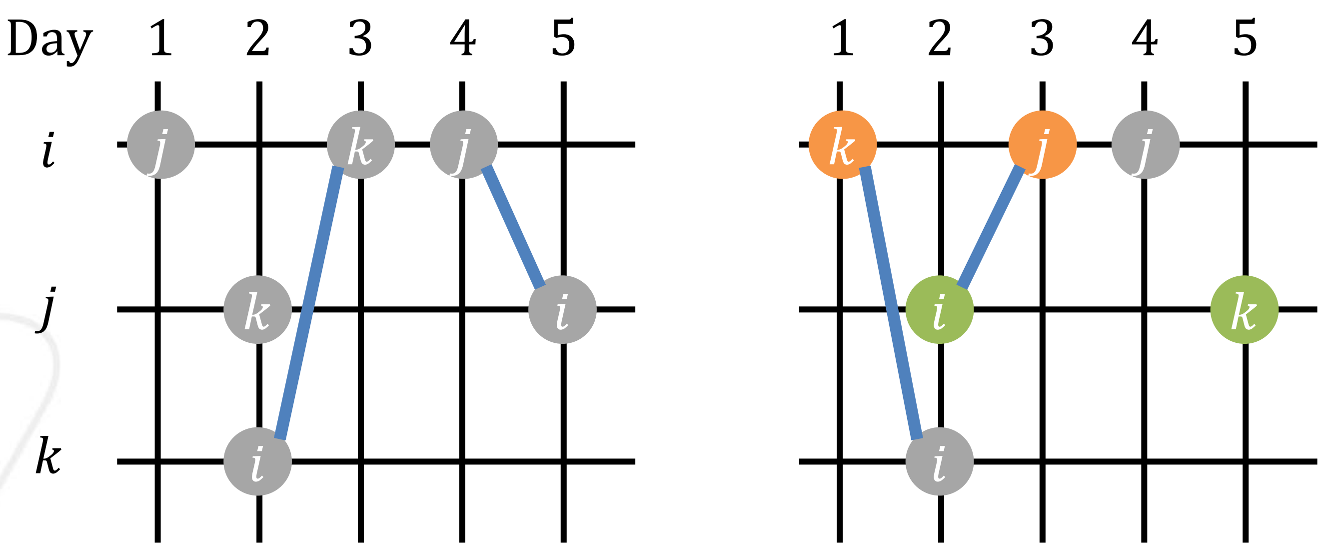
- z_{ij}^* : optimal solution of the fluid relaxation
- $\bar{z}_{ij} = z_{ij}^* / \sum_{k\in N\setminus i} z_{ik}^*$ (normalized, asymmetric)
- Assign arriving type i node to sub-type (i,j) w.p. $\bar{z}_{ij}$



Analysis for the general case

Policy comparison

- Batching dominates its randomized counterpart
- **Modified Greedy policy**: only optimize among nodes that randomized policy would have matched, delay executing matches



Impatient nodes

Unmatched node i abandons the system w.p. $d_i(\tau)$ in each period

Policy	Assumption
Batching	$d_i(\tau) = \mathcal{O}(\tau^{-\beta})$ for $\beta > 1,\ i \in N$
Randomized Greedy	$d_i(\tau) = o(1),\ i \in N$

<u>Batching policy</u>

- If $d_i(\tau) = 1/\tau$ and $p_i = p_j = p$,

  matching frequency $\rightarrow p(1 - 1/e) < p$ = greedy matching frequency

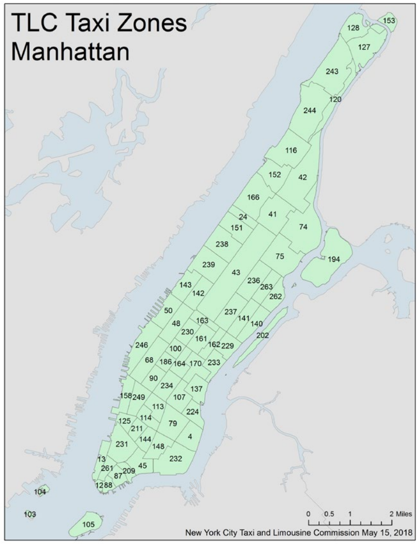
<u>Greedy policy</u>

- If $d_i(\tau) = 1 - \epsilon, \epsilon > 0$ and $p_i = p_j = p < 1$,

  matching frequency $\rightarrow p^2 < p$ = fluid matching frequency

Georgia Tech

Computational study



Ridesharing marketplace

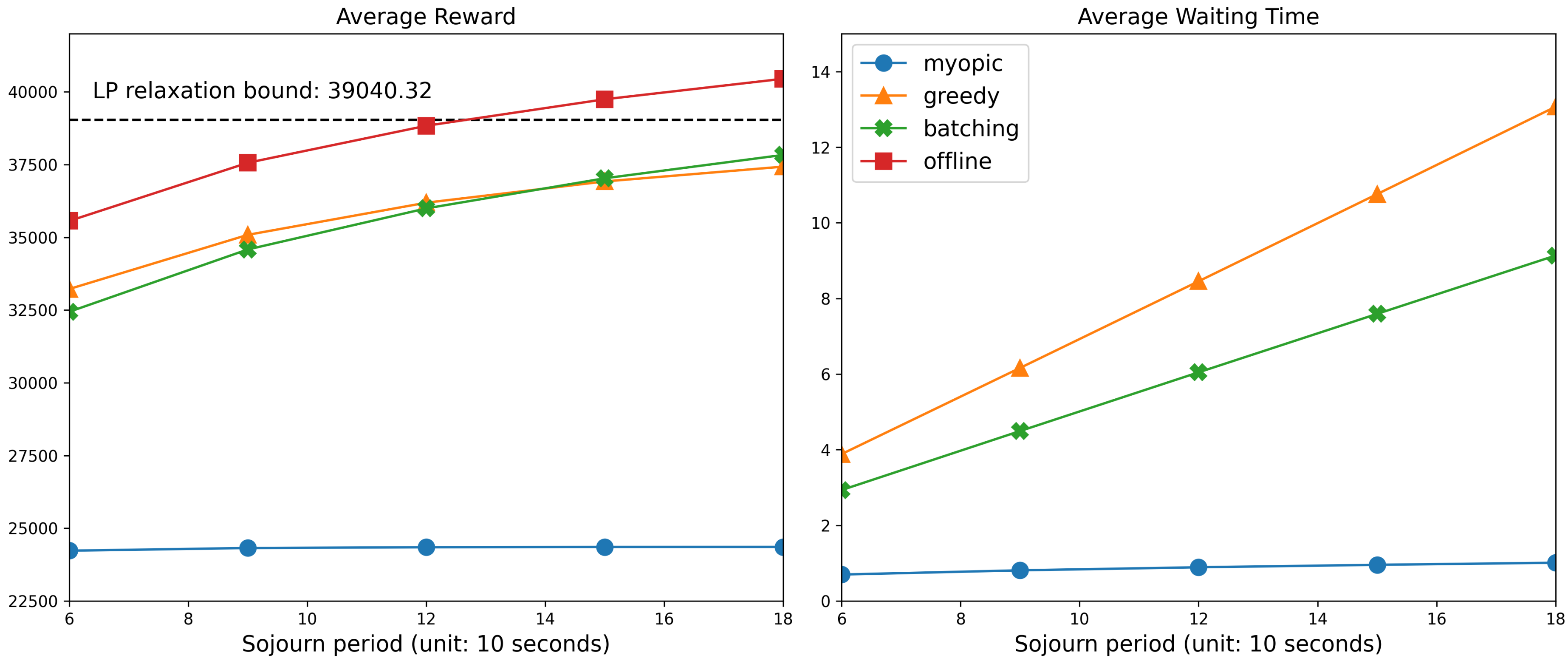
- Manhattan, New York City (69 taxi zones)
- For-hire vehicle record, NYC Open Data platform (February 2020, 6-10 am, Mon to Fri except holidays)
- Matching reward for a request pair : travel miles saved

Simulation parameters

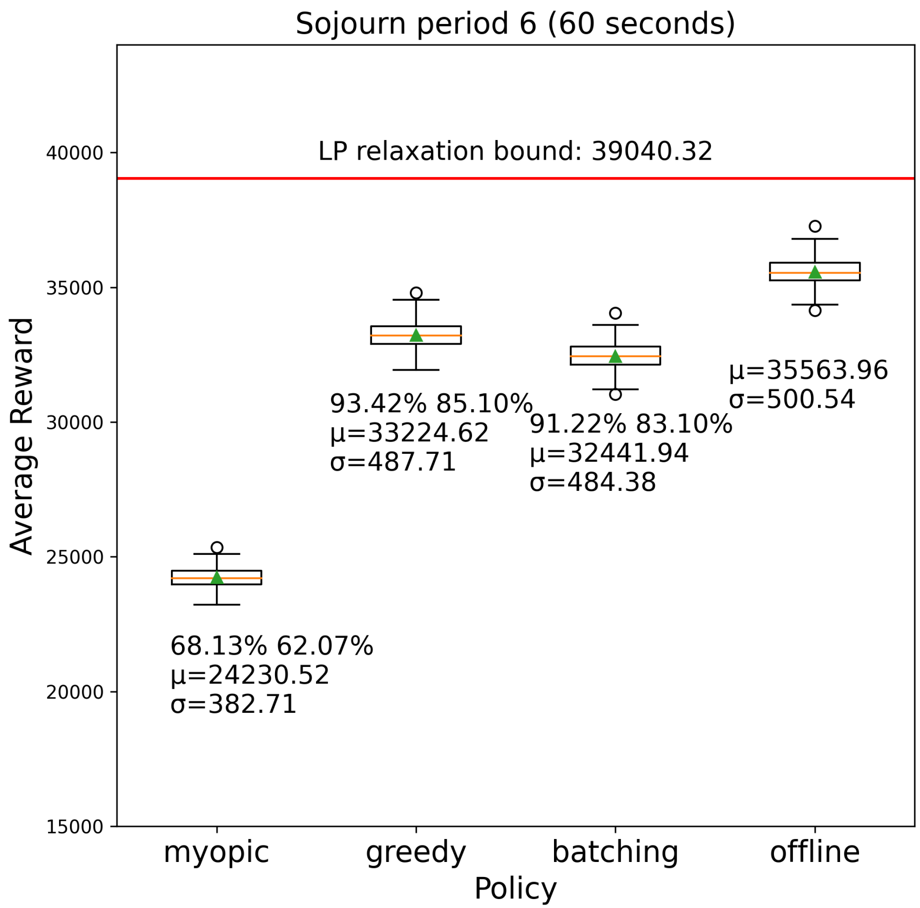
- Policies : myopic, (delayed) greedy, batching, offline benchmark
- Time unit: 10 seconds
- Time horizon length : 1 hour (360 periods)
- $\tau \in \{6, 9, 12, 15, 18\}$
- Simulation replication : 200

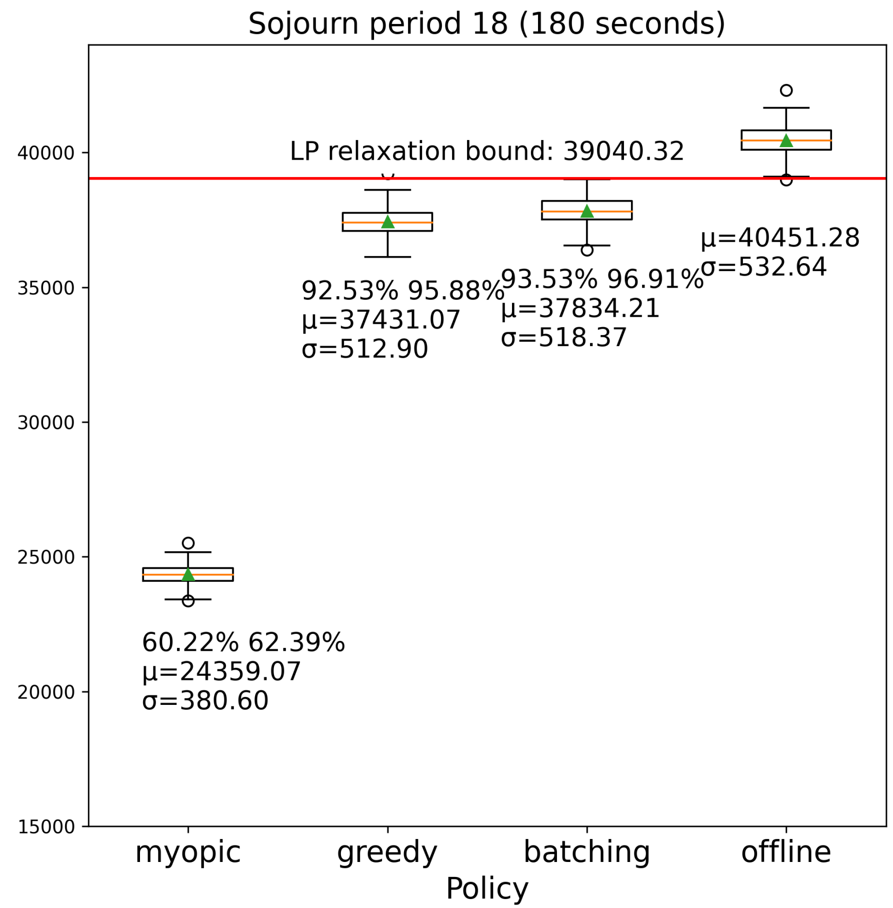
Image source: NYC Taxi & Limousine Commission website

Computational study



Computational Study





Conclusions

- Prove asymptotic optimality of two widely used policies
  - batching and greedy policies
  - dynamic stochastic non-bipartite matching
  - freight transportation and ridesharing marketplaces

- Extend to impatient setting

- Show these policies perform well in practice with reasonably small sojourn periods

- Motivate models in which
  - matching reward depends on its nodes' waiting times
  - nodes' sojourn periods are random

sites.gatech.edu/alejandro-toriello