On Algorithmic Decision Procedures in Emergency Response Systems in Smart and Connected Communities

Geoffrey Pettet[1], Ayan Mukhopadhay[2], Mykel Kochenderfer[2], Yevgeniy Voroybeychik[3], Abhishek Dubey[1]

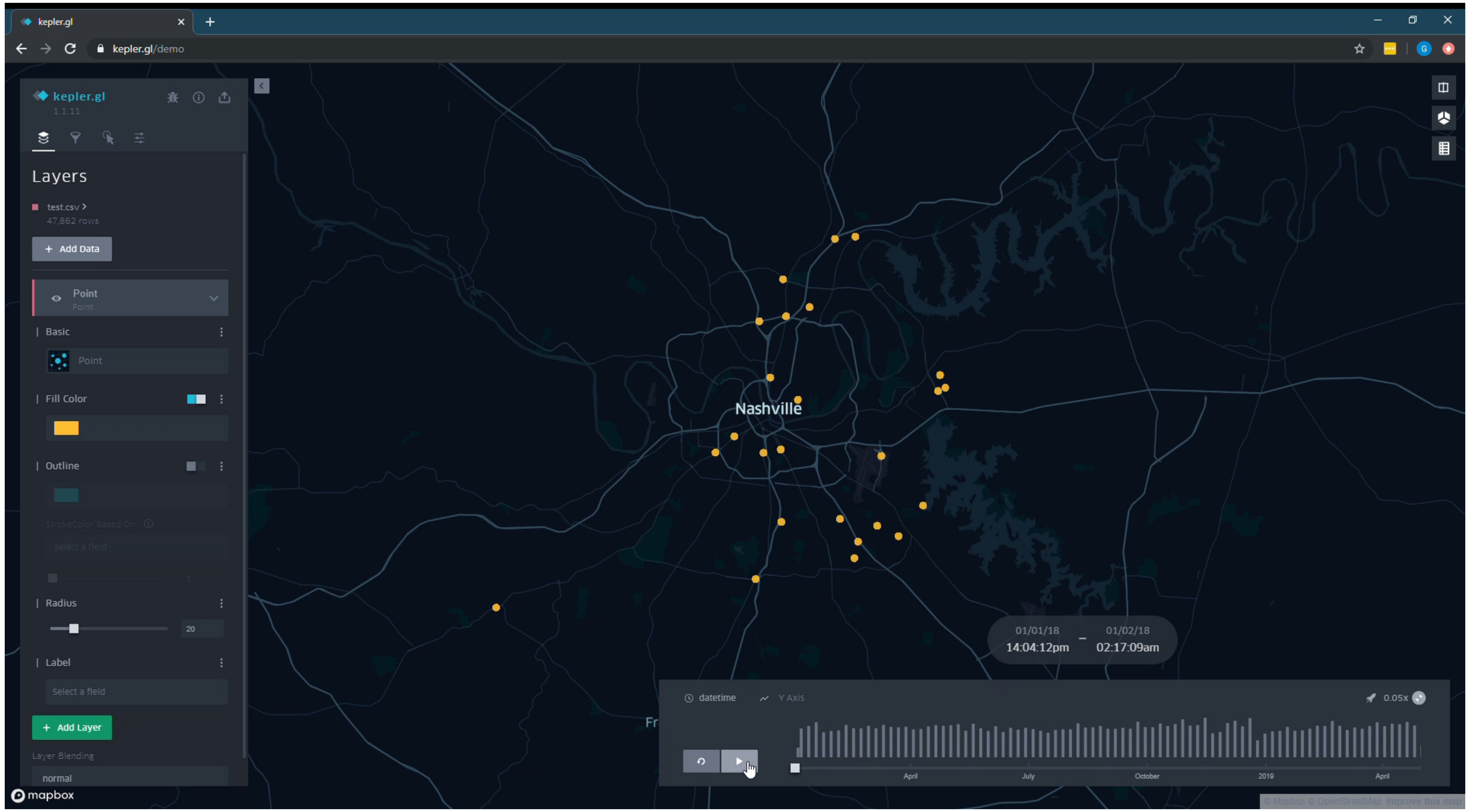
[1]Vanderbilt University, [2]Stanford University, [3]Washington University in St Louis

Sponsored by National Science Foundation, Center for Automotive Research at Stanford (CARS), and Tennessee Department of Transportation

Motivation and Background

The emergency response problem

All traffic incidents occurring in Davidson County In January 2018, with a sliding window of ~12 hours



The problem: Respond Efficiently to all incidents spread over a large geographic area with limited resources.

Proactive Emergency Response

Active Learning and Improvement Mechanisms

Online Demand Estimation Models[1] → Anticipatory Stationing of Resources → Optimal Dispatch[1]

[1] Ayan Mukhopadhyay, Geoffrey Pettet, Chinmaya Samal, Abhishek Dubey, and Yevgeniy Vorobeychik. 2019. An online decision-theoretic pipeline for responder dispatch. In Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS ’19). Association for Computing Machinery, New York, NY, USA, 185–196. DOI:https://doi.org/10.1145/3302509.3311055

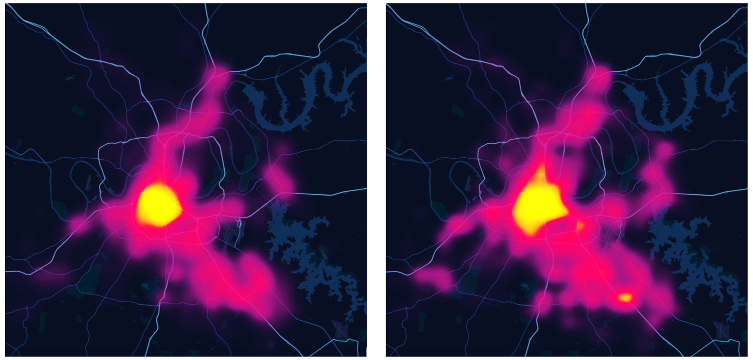
Proactive Emergency Response

Previous Work

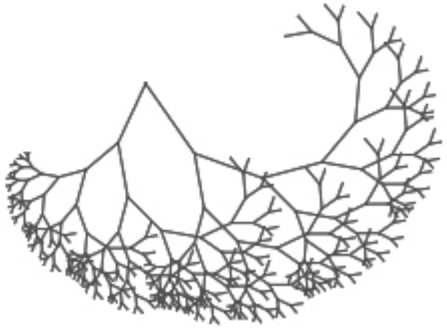
Active Learning and Improvement Mechanisms

Online Demand Estimation Models[1] → Anticipatory Stationing of Resources → Optimal Dispatch[1]

[1]



Prediction Actual



Tree Search (MCTS)

[1] Ayan Mukhopadhyay, Geoffrey Pettet, Chinmaya Samal, Abhishek Dubey, and Yevgeniy Vorobeychik. 2019. An online decision-theoretic pipeline for responder dispatch. In Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS '19). Association for Computing Machinery, New York, NY, USA, 185–196. DOI:https://doi.org/10.1145/3302509.3311055

Proactive Emergency Response

Focus of Paper

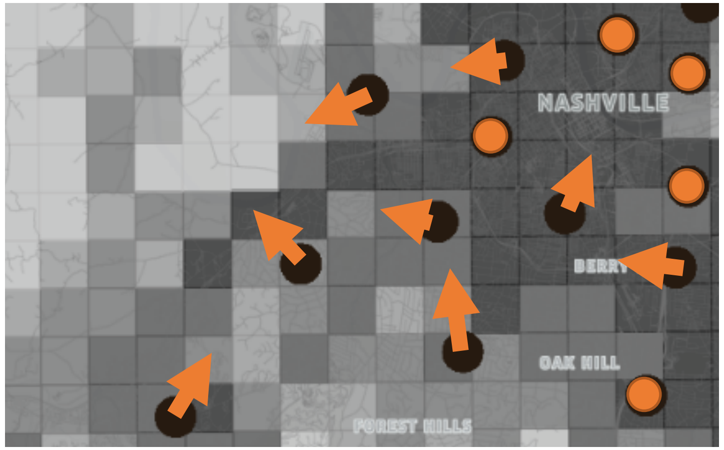
Active Learning and Improvement Mechanisms

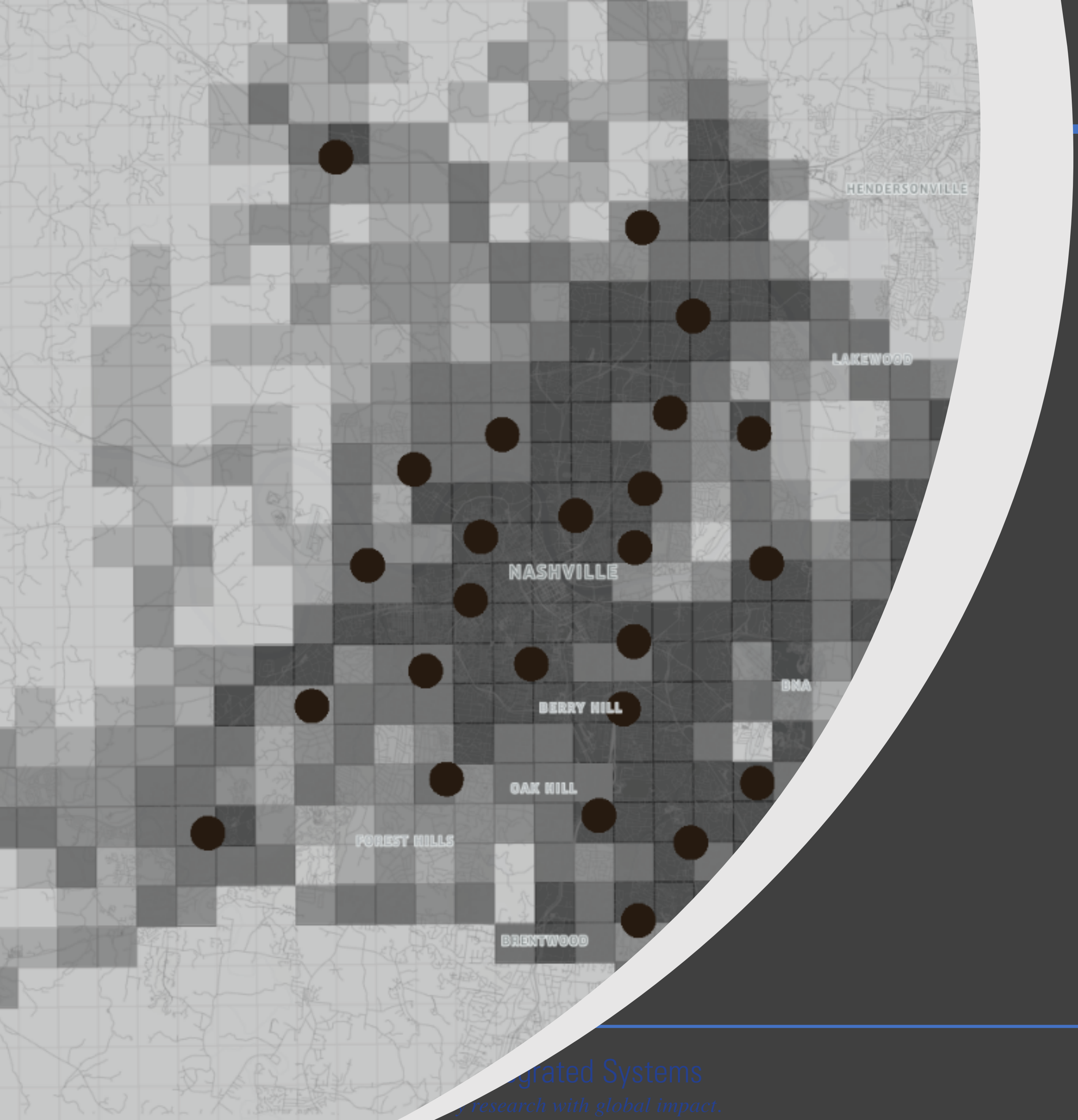
Online Demand Estimation Models[1] → **Anticipatory Stationing of Resources** → Optimal Dispatch[1]

("Rebalancing")

Advantages over decision making at time of dispatch:

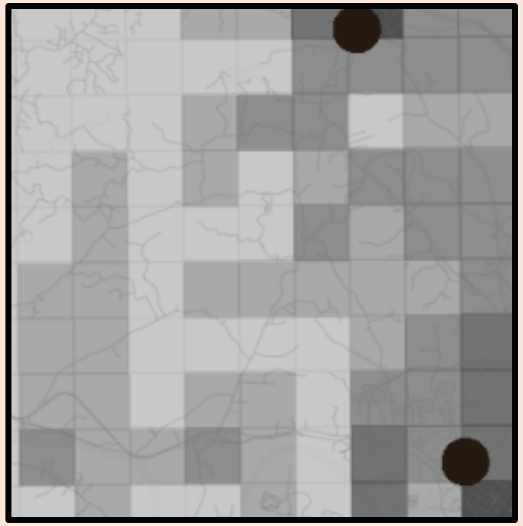
- Ample time to make decision
- Avoids legal and moral questions
- Proactive
- Larger decision space => more room for gains



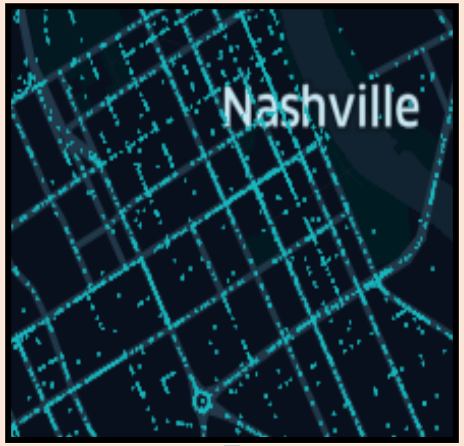


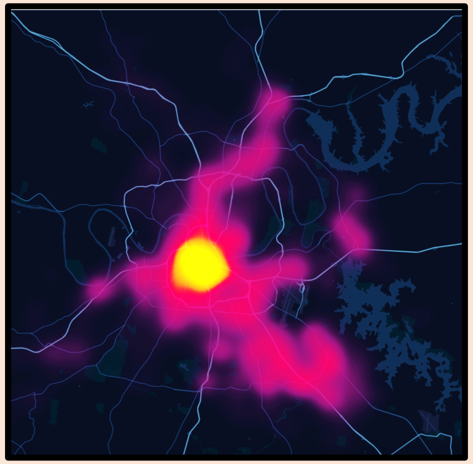
System Model and Assumptions

System Model – Assumptions



Region segmented into a grid with equally sized cells





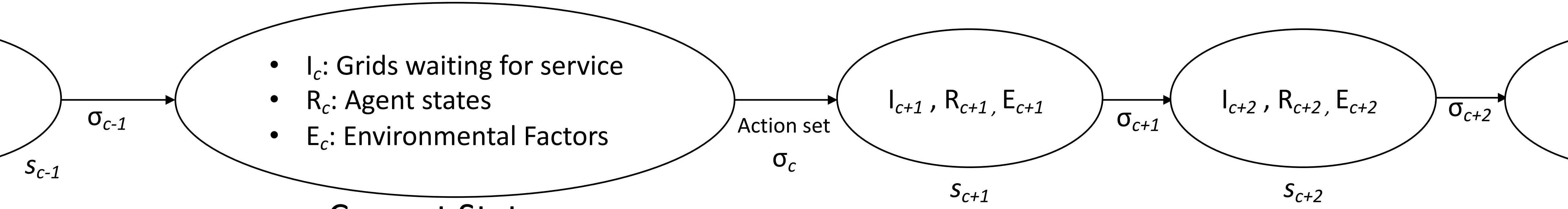
Historical data (incidents, traffic, weather, etc.)

Incident arrival model

https://www.nashville.gov

“Depots” – subset of cells where agents can wait

System Model – Multi Agent SMDP



Current State s_c

*SMDP diagram simplified for demonstration

States

- Continuous state space
- Discrete states of interest:
  - Incident occurrence
  - Responder availability
  - Rebalancing triggered

Actions

- Directing agents to valid cells:
  - Response: pending incident locations
  - Rebalancing: depots

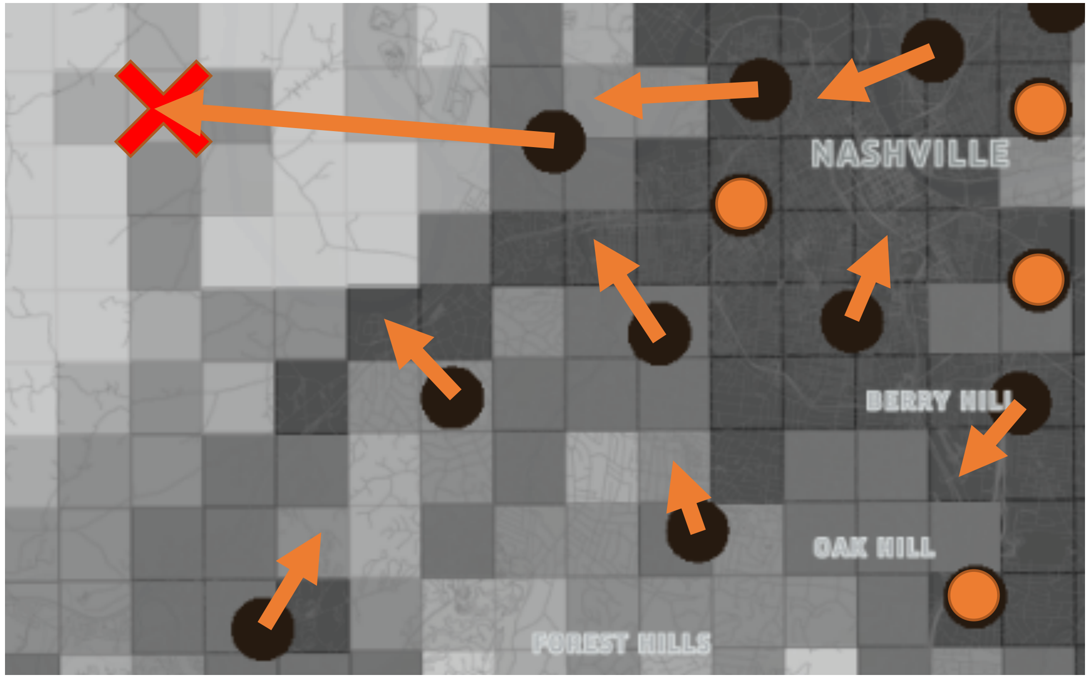
Transitions

- Time between incidents
- Incident Service time
- Computation time
- Travel time

Rewards

- Balance-
  - minimizing response times
  - minimizing distance traveled

Problem Definition



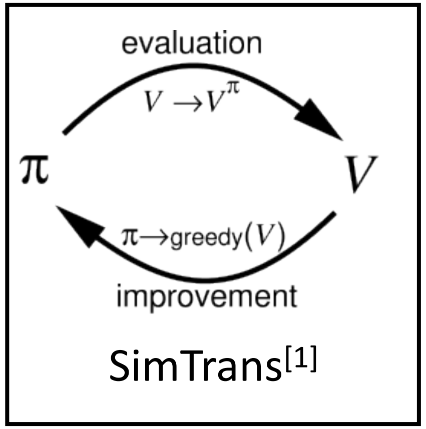
Given: System state, predicted spatial-temporal incident distribution
Return: Action recommendation set that maximizes expected reward

Reward can be fine-tuned

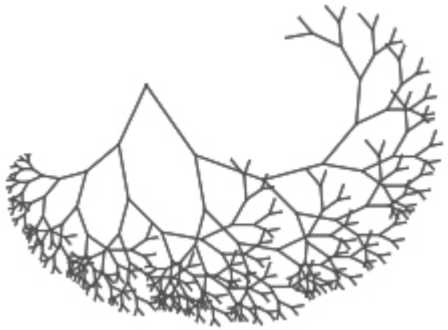
Approaches to Solving SMDP

Policy iteration (Dispatch)[1]

- Will converge to best dispatch policy eventually
- Slow – must estimate state transition probabilities



MCTS (Dispatch)

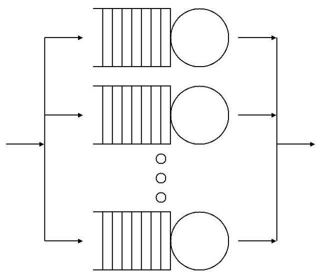


- Anytime algorithm
- Not scalable to dynamic balancing

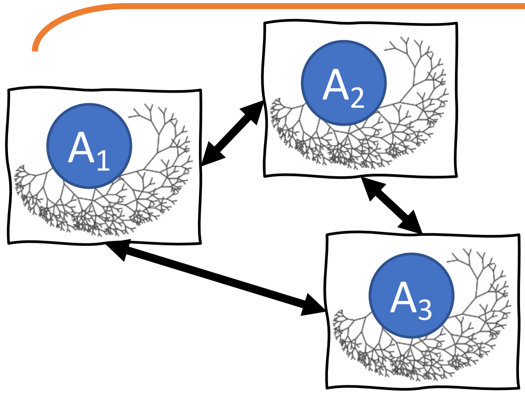
New Work

Greedy Heuristic Search

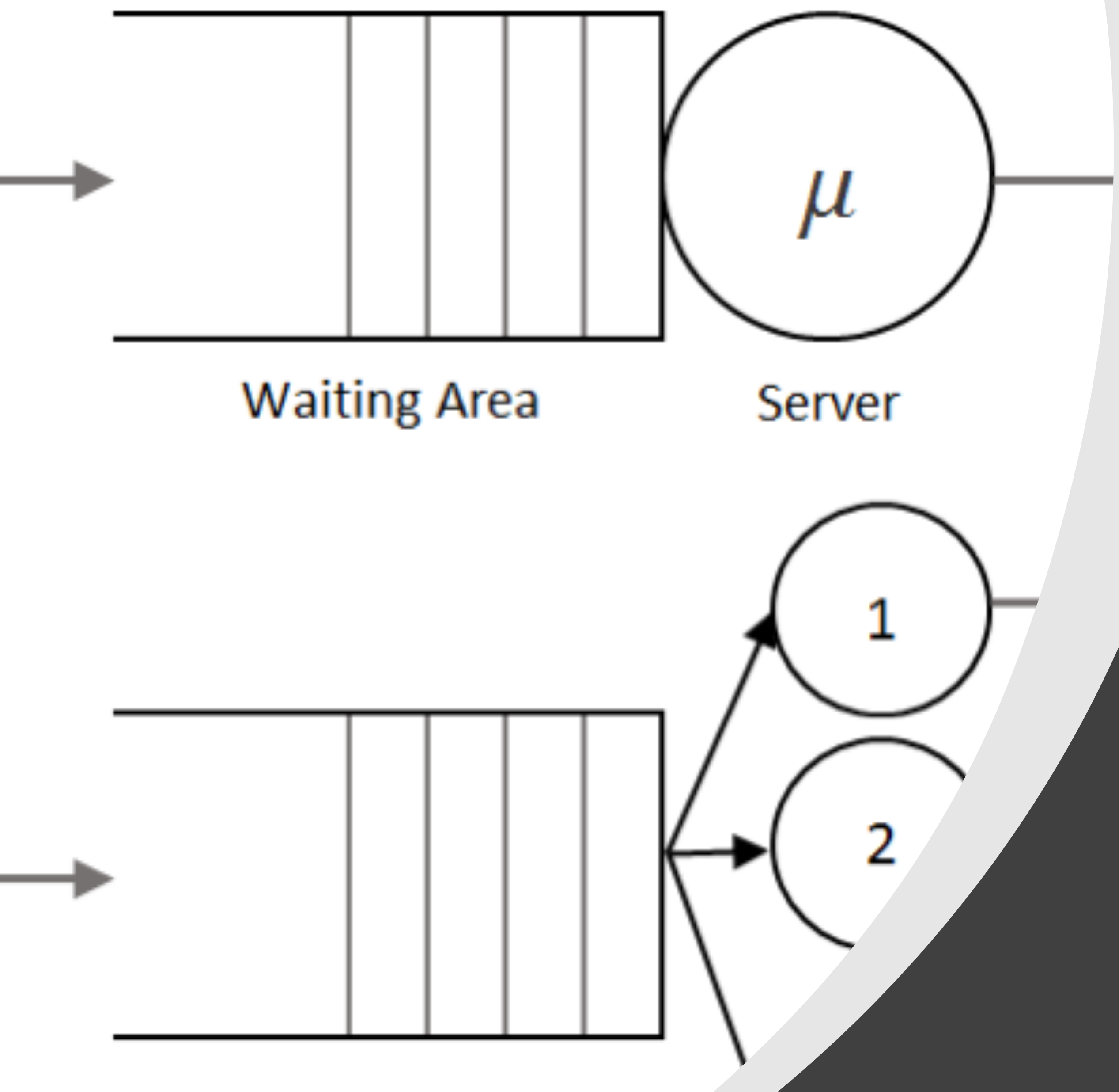
Queueing Theory



Multi- Agent Monte Carlo Tree Search

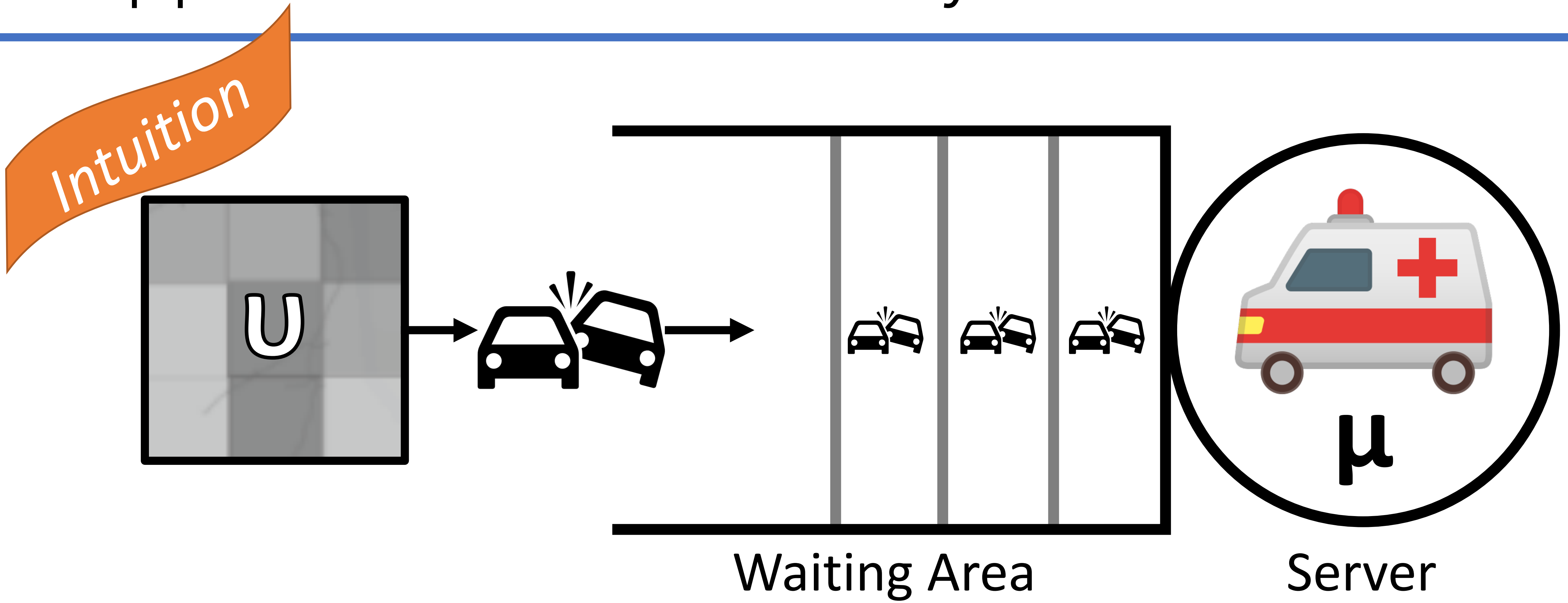


[1] Ayan Mukhopadhyay, Zilin Wang, and Yevgeniy Vorobeychik. 2018. A Decision Theoretic Framework for Emergency Responder Dispatch. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 588–596.



Approach 1: Greedy Search with Queue-Heuristic

Approach 1: Queue Theory Heuristic Search

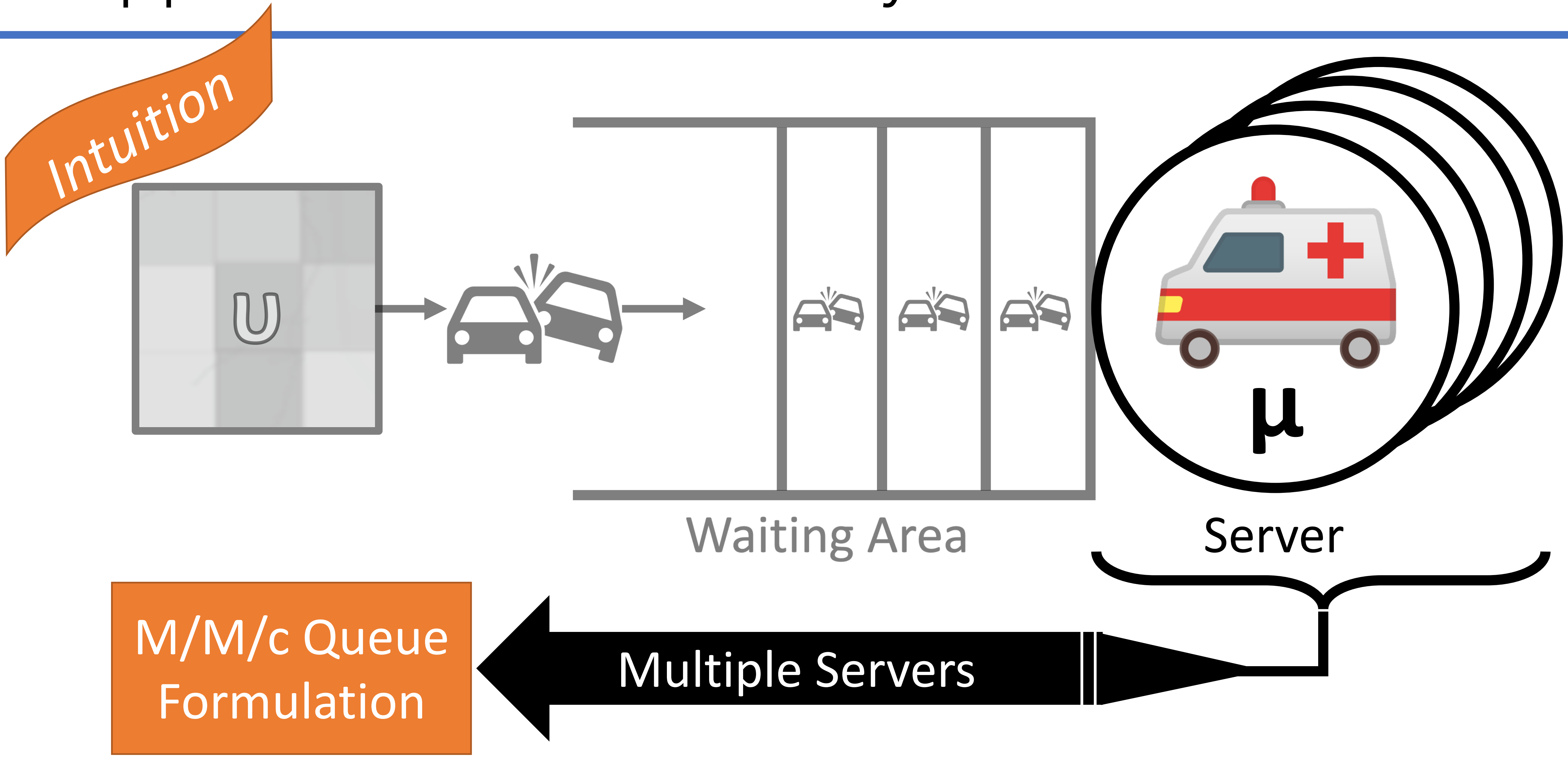


Grid Cell incident rate := **υ**

Incidents Placed in Waiting Queue

Ambulances 'serve' incidents with mean rate := **μ**

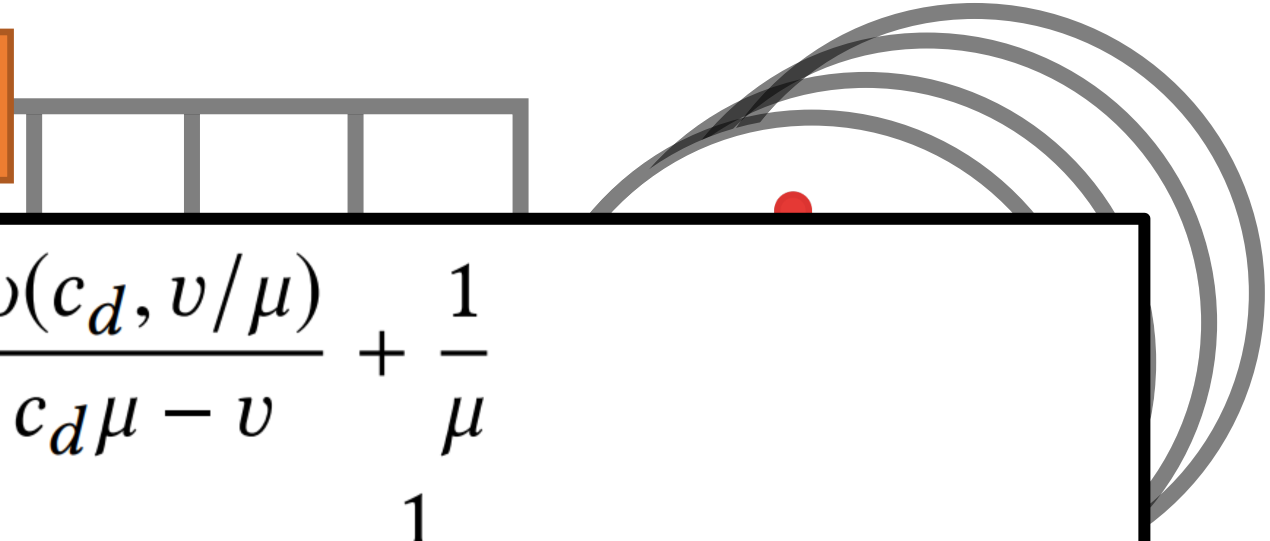
Approach 1: Queue Theory Heuristic Search



Approach 1: Queue Theory Heuristic Search

Intuition

Queue Response time:
Avg time in system



$$\text{responseTime}(c_d, v, \mu) = \frac{\omega(c_d, v/\mu)}{c_d\mu - v} + \frac{1}{\mu}$$

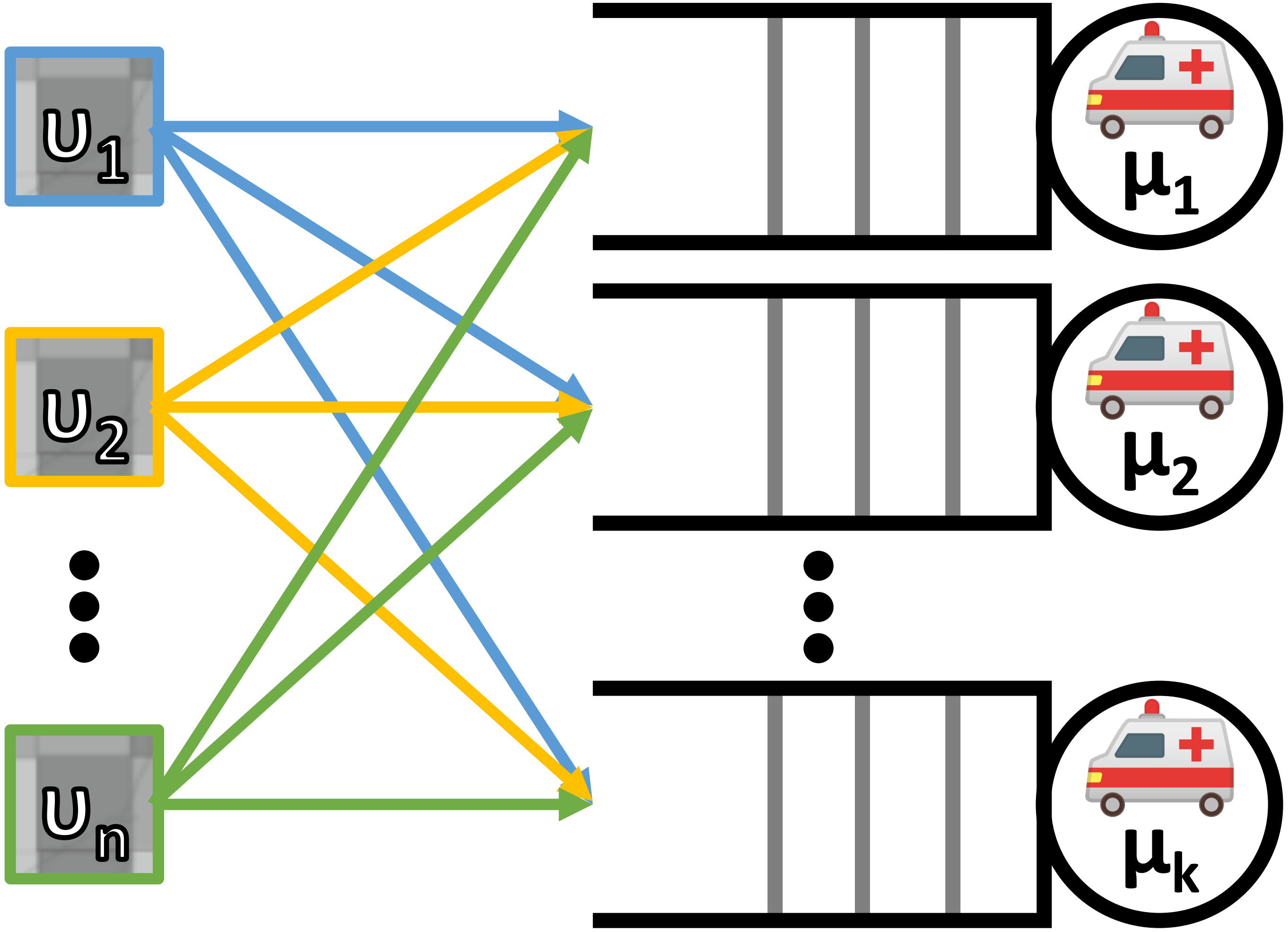
$$\text{where } \omega(c_d, v/\mu) = \frac{1}{1 + (1 - \frac{v}{c_d\mu})(\frac{c_d!}{(c_d q)^c}) \sum_{k=0}^{c_d-1} \frac{(c_d q)^k}{k!}}$$

M/M/c Queue Formulation

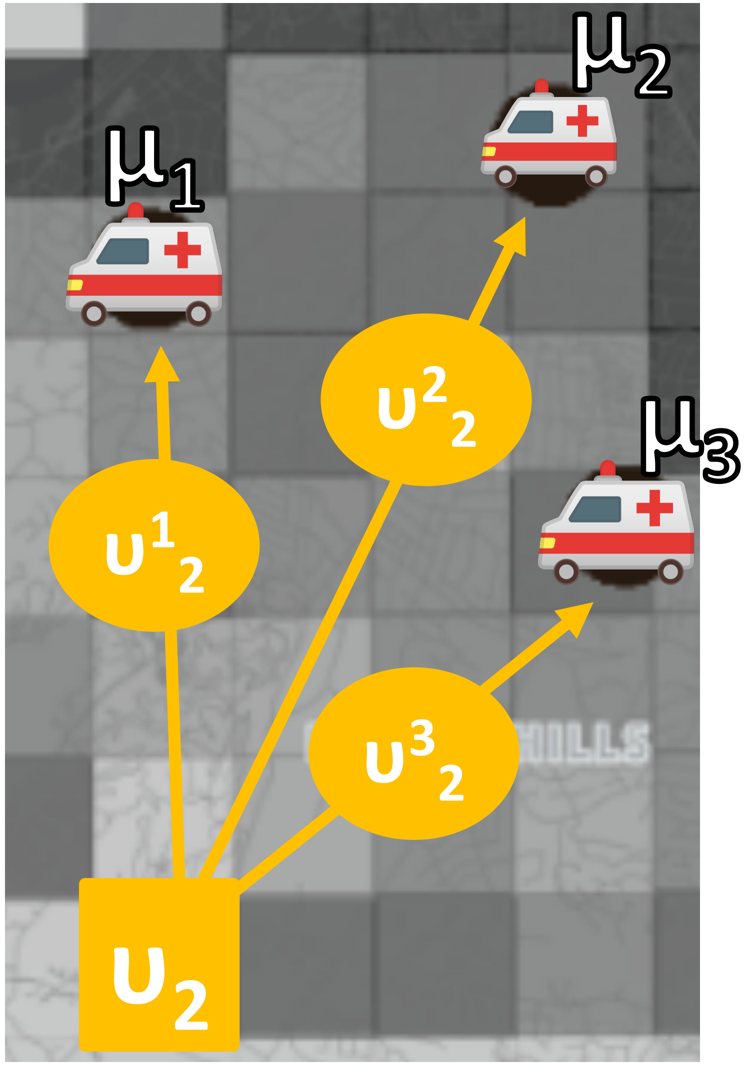
Multiple Servers

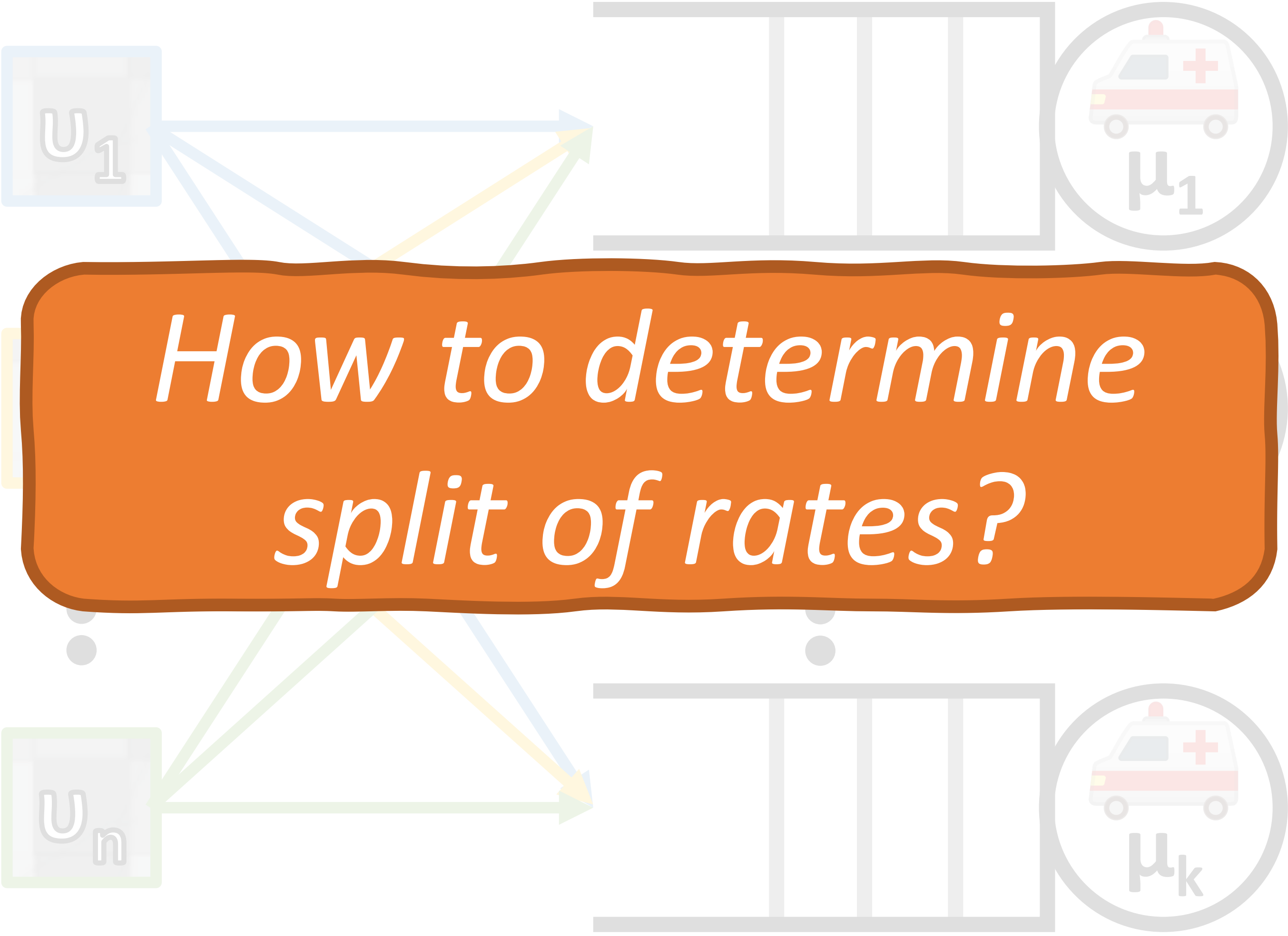
Approach 1: Queue Theory Heuristic Search

"Multi Class, Multi Server Queue Formulation"



Approach 1: Queue Theory Heuristic Search





How to determine split of rates?

Approach 1: Queue Theory Heuristic Search

$$\sum_{d \in D} \upsilon_g^d = \upsilon_g \qquad (2a)$$

$$\text{dist}(\tilde{d}, g)\upsilon_g^{\tilde{d}} = \text{dist}(d_i, g)\upsilon_g^{d_i} \quad \forall d_i \in D \backslash \tilde{d} \qquad (2b)$$

- For each cell, distribute rate among depots *inversely proportional to the distance from the cell to the depot*
  - Closer depots => higher portion of rate
- Solve System of Linear equations above for each cell
  - υ_g^d is the fraction of arrival rate for cell *g* that is shared by depot *d*

Approach 1: Queue Theory Heuristic Search

- To score a particular allocation of agents:
  - Must consider travel times => not memoryless, so model explicitly
  - Υ represents collection of split rates
  - Score π_Υ => sum across all cells and depots
    - Estimated (queue) response times (waiting + service time)
    - Travel time from depot to cell

$$\boxed{\pi_\Upsilon = \sum_{d \in D} \sum_{g \in G} \mathbb{1}(d, \Lambda)\{responseTime(c_d, v_g^d, \mu) + travelTime(d, g)\}}$$

Approach 1: Queue Theory Heuristic Search

- Depot selection: Greedy Search
  - One by one select depot that minimizes π_Υ
    - Add to chosen set
  - Re-split rates and calculate new scores with each new depot placed
  - Continue until the number of depots chosen is the same as number of agents
- Assign agents to chosen depots by minimizing distance traveled (Linear Program)

Algorithm 1: Iterative Greedy Action Selection

INPUT: number of agents $|\Lambda|$, depots D, depot capacities C, grid rates $v_g \forall g \in G$;
final_depot_occupancy := Hash $\{d : 0\} \forall d \in D$;
do
  candidate_depots := Set $\emptyset$;
  candidate_scores := Hash $\emptyset$;
  for $d \in D$ **do**
    if *final_depot_occupancy*$[d] < C(d)$ **then**
      temp_occ := final_depot_occupancy;
      temp_occ$[d]$ + = 1;
      find Υ_d by solving system of linear equations {2a, 2b} given temp_occ;
      $\pi_{\Upsilon_d} := \sum_{d \in D} \sum_{g \in G} \mathbb{1}(d, \Lambda)\{responseTime(\text{temp\_occ}[d], v_g^d, \mu) + travelTime(d, g)\}$;
      candidate_depots := candidate_depots $\cup d$;
      candidate_scores := candidate_scores $\cup \{d : \pi_{\Upsilon_d}\}$
  best_depot := argmin π_{Υ_d} $\forall d \in$ candidate_depots;
  final_depot_occupancy[best_depot] + = 1;
while $sum(final\_depot\_occupancy) < |\Lambda|$;
return chosenDepots;

Approach 1: Overview

Choose depots via greedy search

- Repeat until # chosen depots == # agents:
  - Split incident rates across depots

$$\sum_{d \in D} v_g^d = v_g \tag{2a}$$

$$\mathrm{dist}(\tilde{d}, g) v_g^{\tilde{d}} = \mathrm{dist}(d_i, g) v_g^{d_i} \quad \forall d_i \in D \backslash \tilde{d} \tag{2b}$$

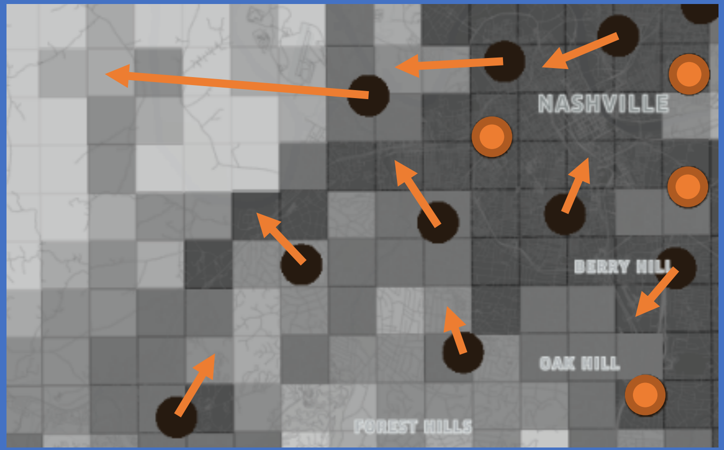
  - Score allocations

$$\pi_\Upsilon = \sum_{d \in D} \sum_{g \in G} \mathbb{1}(d, \Lambda)\{responseTime(c_d, v_g^d, \mu) + travelTime(d, g)\}$$

  - Add depot that minimizes score

Assign agents to chosen depots

- Minimize distance traveled
- LP, Greedy Search, etc.



Approach 1: Overview

Choose d … sen depots

- Repeat u … ed agents:
  - Split in

$\sum_{d \in D} v_g^d$ …

$\mathrm{dist}(\tilde{d}, g) v_g^{\tilde{d}} = \mathrm{dist}(d_i, g) v_g^{d_i} \quad \forall d_i \in D \backslash \tilde{d}$ (2b)

  - Score a

$\pi_\Upsilon = \sum_{d \in D} \sum_{g \in G}$ …

  - Add de

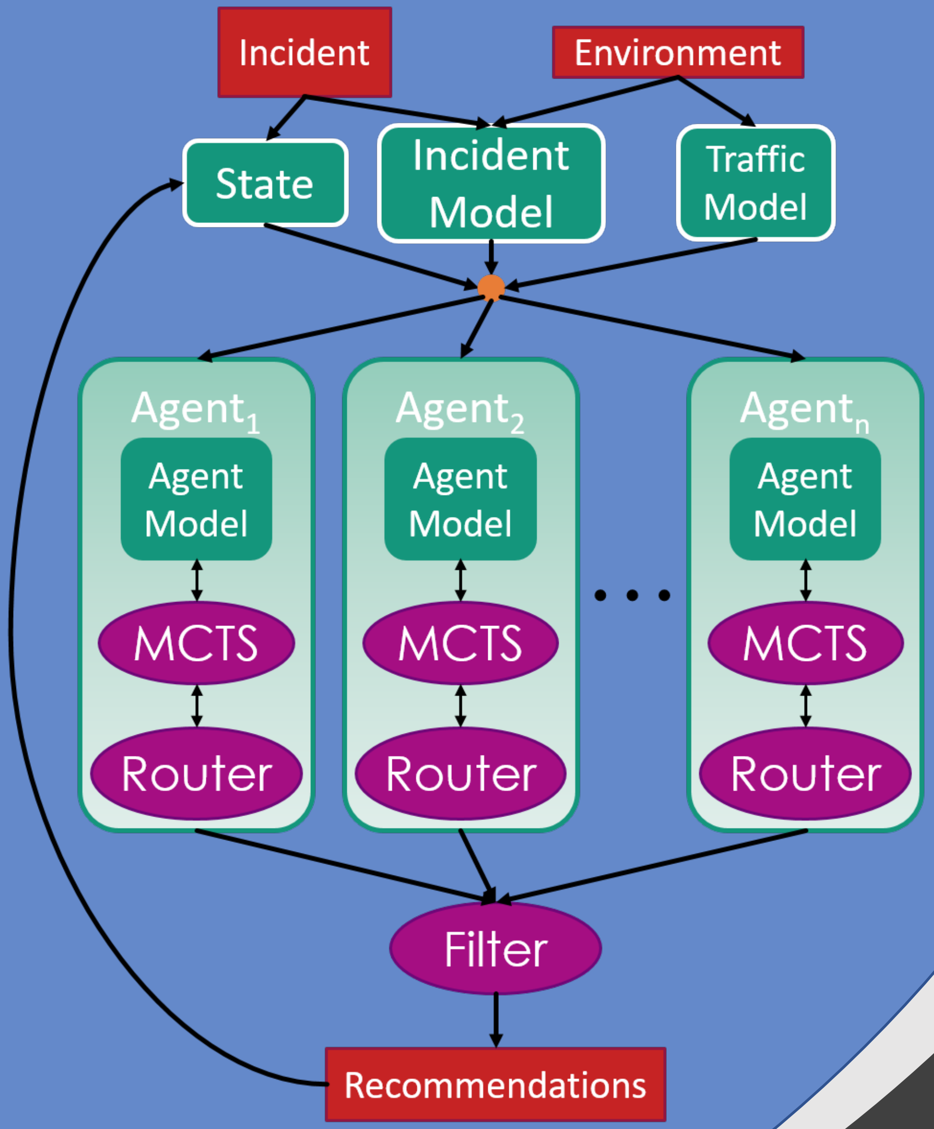
Advantage:

- Computationally efficient

Disadvantages:

- Doesn't take internal system state into account
- Ignores dynamic incident rate distribution

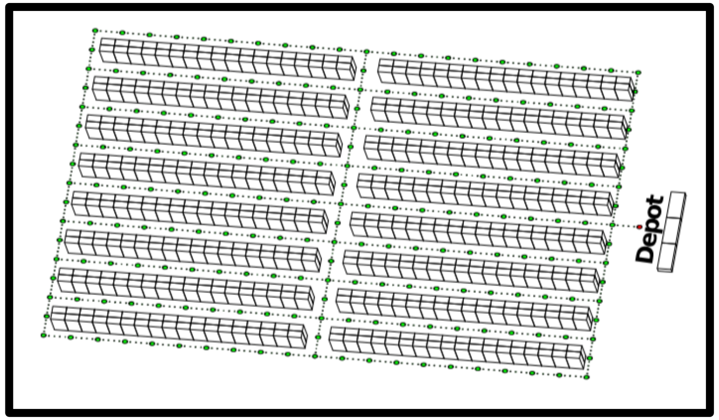
Partially Decentralized Decision Process



Approach 2: Multi-Agent Monte Carlo Tree Search (MMCTS)

Approach 2: MMCTS

Inspiration



Typical Warehouse Model

Decentralised Online Planning for Multi-Robot Warehouse Commissioning

Daniel Claes
smARTLab, Department of Computer Science
University of Liverpool, UK
daniel.claes@liverpool.ac.uk

Frans Oliehoek
smARTLab, University of Liverpool
AMLab, University of Amsterdam
fao@liverpool.ac.uk

Hendrik Baier*
Digital Creativity Labs, Department of Computer Science
University of York, UK
hendrik.baier@york.ac.uk

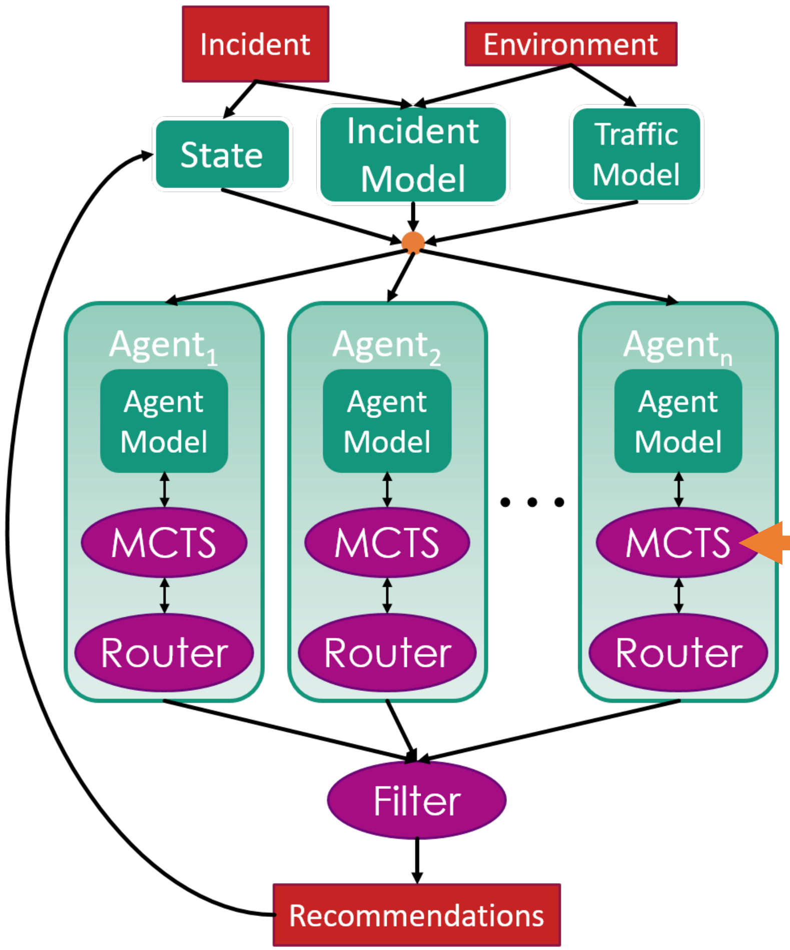
Karl Tuyls
smARTLab, Department of Computer Science
University of Liverpool, UK
k.tuyls@liverpool.ac.uk

Claes, Daniel, et al. "Decentralised online planning for multi-robot warehouse commissioning." *AAMAS'17: PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS*. 2017.

**Improved on state of the art, particularly in cases with large state space*

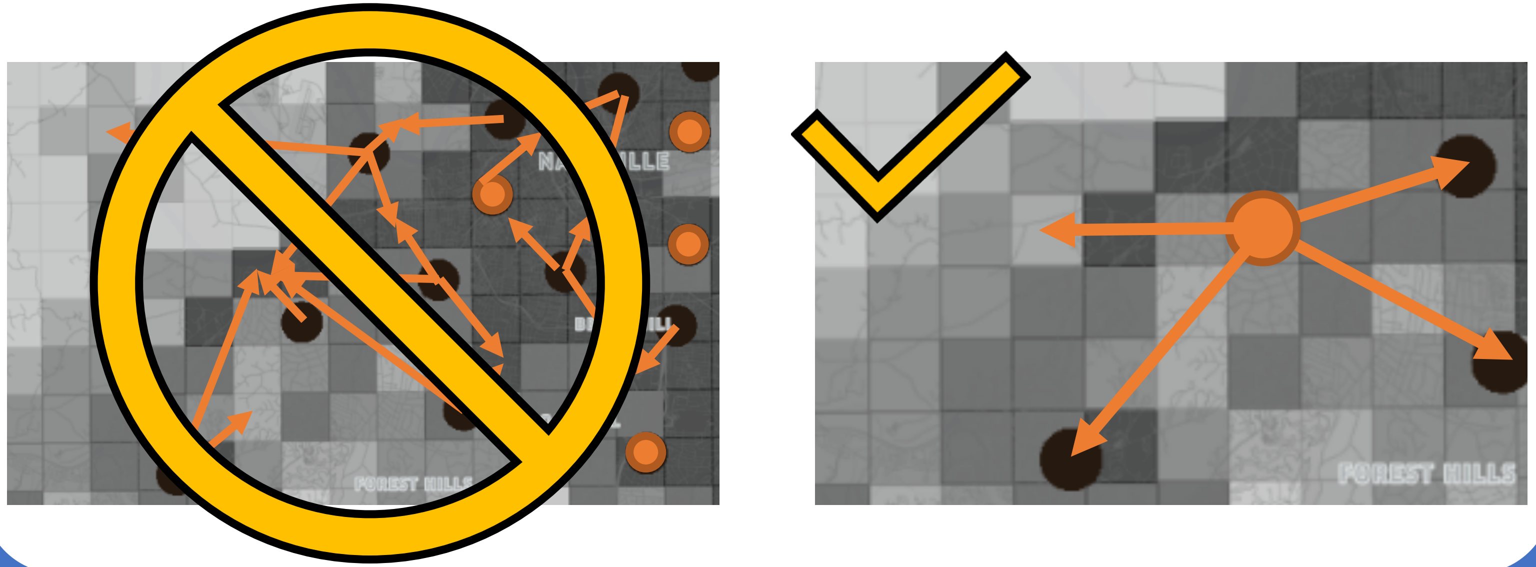
Approach 2: MMCTS

Partially Decentralized Decision Process



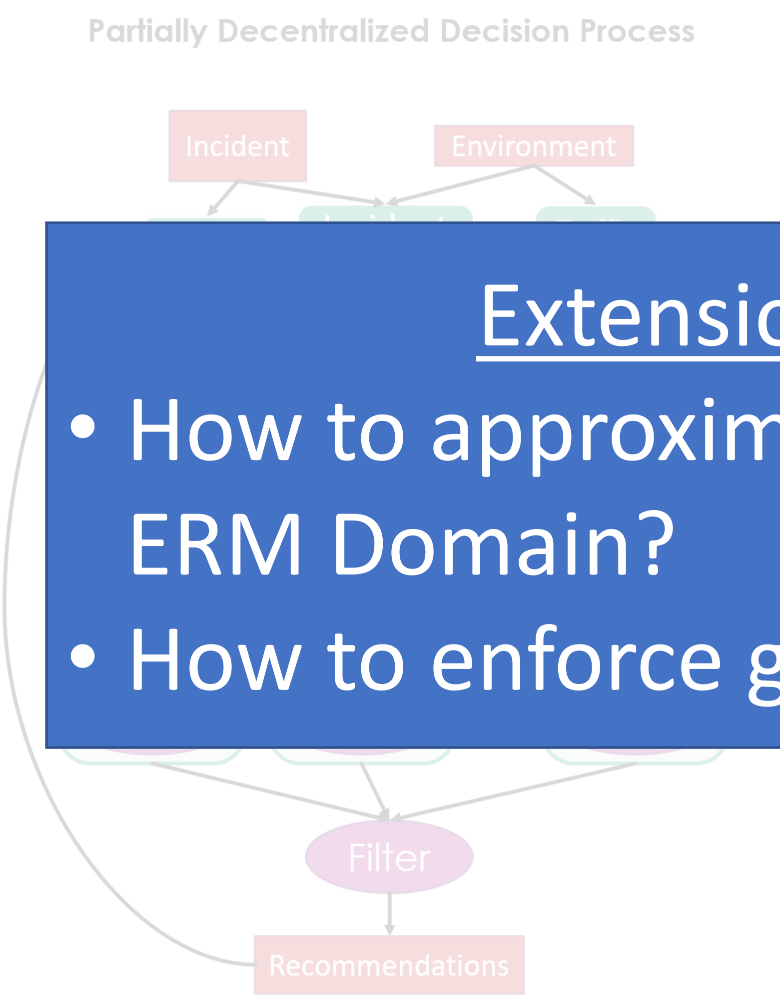
Monolithic State Space

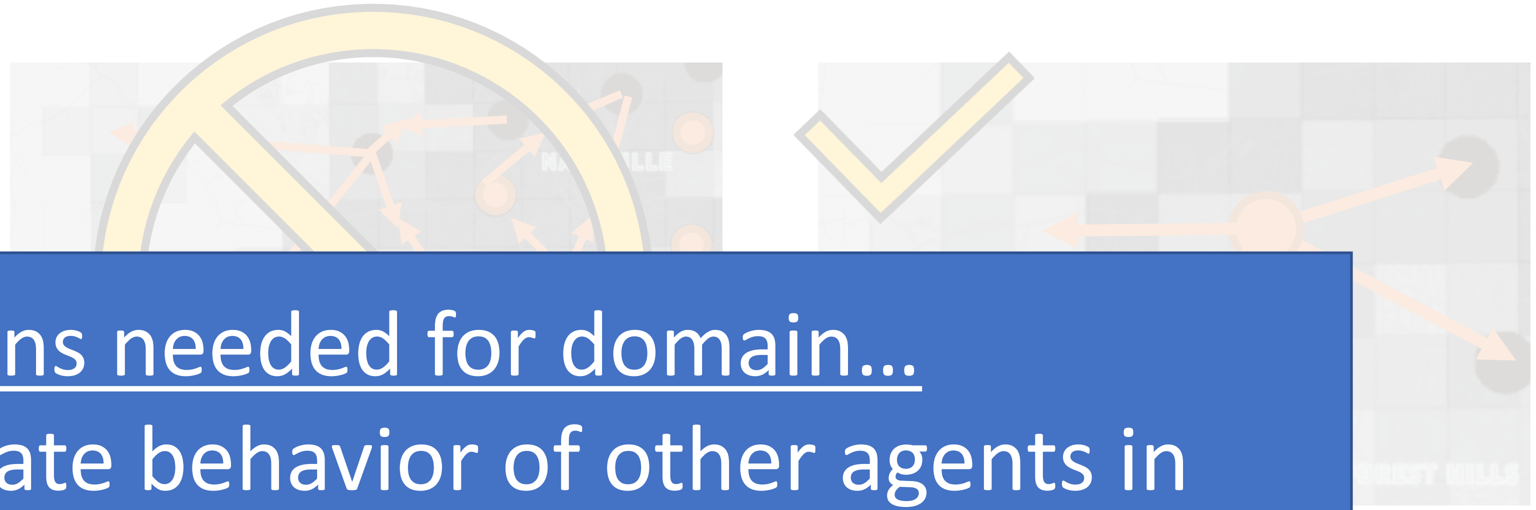
State space split for each agent



Standard MCTS;
Action space limited to relevant actions for the Agent

Approach 2: MMCTS





<u>Extensions needed for domain…</u>

- How to approximate behavior of other agents in ERM Domain?
- How to enforce global constraints?

relevant actions for the Agent

Approach 2: MMCTS

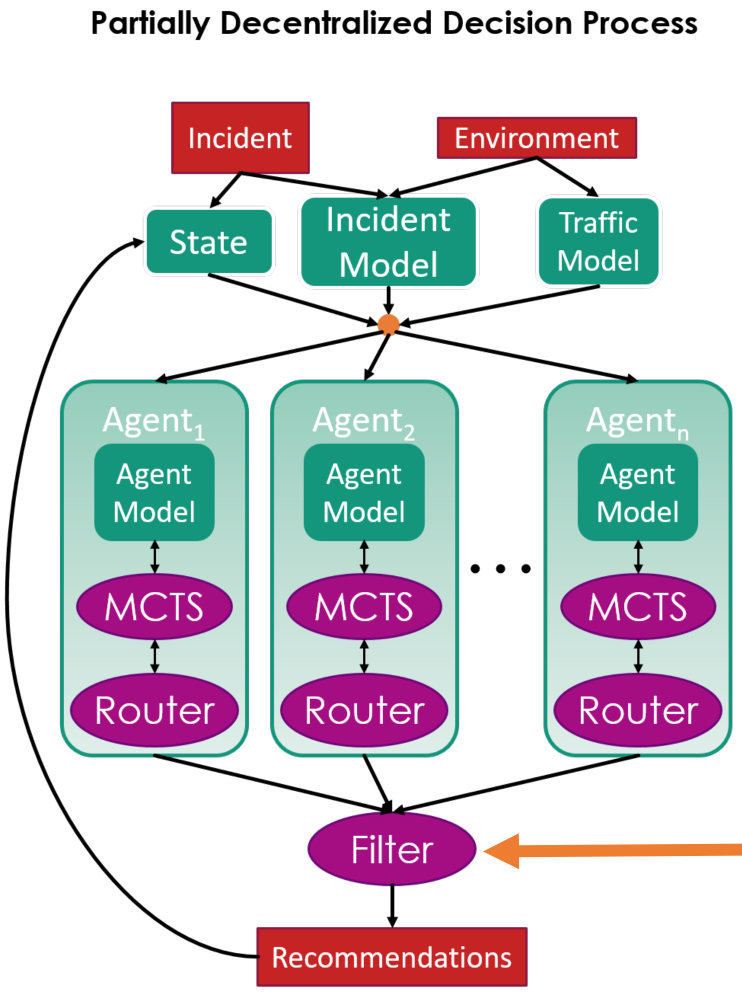
Approximate Agent behavior

- 1) *Naïve policy:* other agents do not rebalance
- 2) *Informed policy:* use queue-based heuristic formulated in approach 1!

Enforcing Global Constraints

- Centralized filter
- Ensure that
  - incidents are responded to
  - Depots aren't filled over capacity
- Uses greedy action assignment based on returned rewards

Approach 2: MMCTS

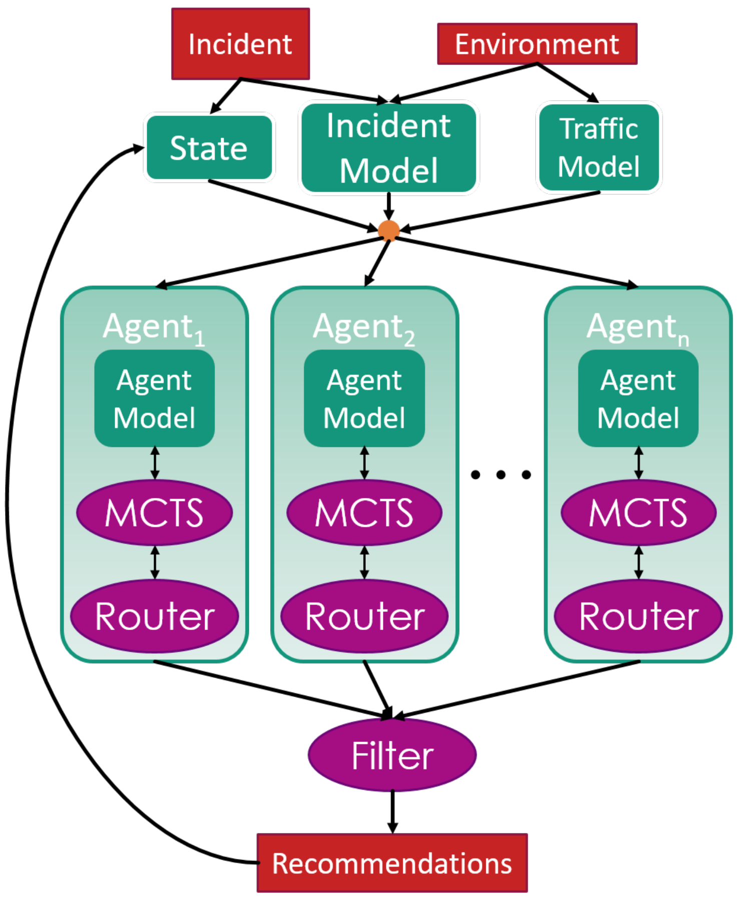


Enforcing Global Constraints

- Centralized filter
- Ensure that
  - incidents are responded to
  - Depots aren't filled over capacity
- Uses greedy action assignment based on returned rewards

Approach 2: MMCTS

Partially Decentralized Decision Process



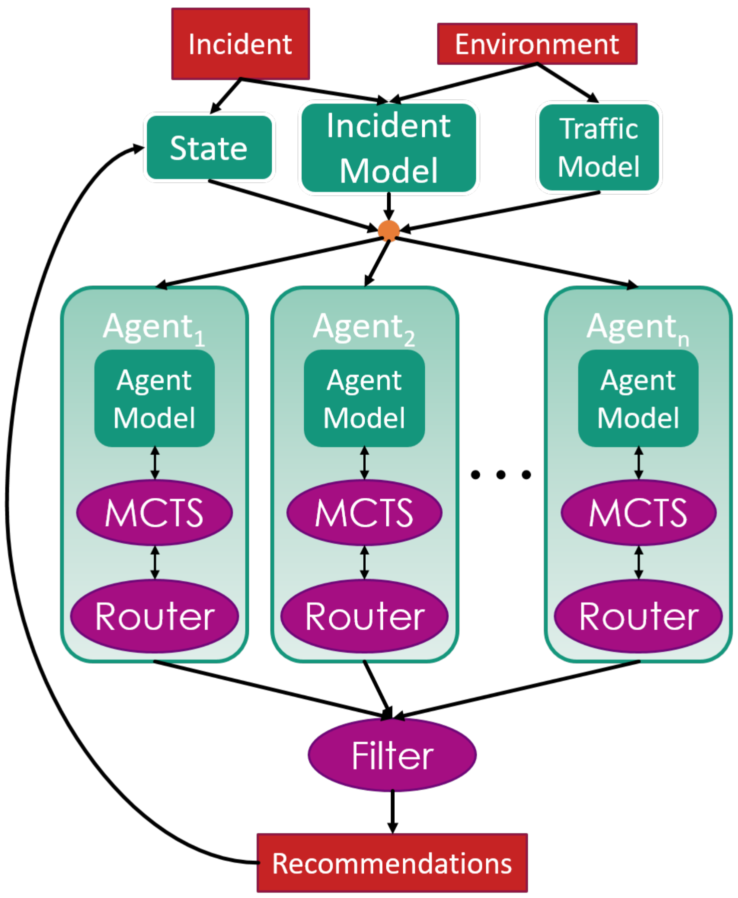
Reward Structure

- Accounts for -
  - Incident dispatch -> *response time*
  - Balancing -> *distance traveled*

$$\rho(s,a) = \begin{cases} \rho_{s-1} - \alpha^{th}(t_r(s,a)), & \text{if responding to an incident} \\ \rho_{s-1} - \alpha^{th}\psi \frac{\sum_{\lambda_k \in \Lambda}(\phi_k(s,a))}{|\Lambda|}, & \text{if balancing at } s \end{cases} \tag{4a}$$

Approach 2: MMCTS

Partially Decentralized Decision Process



Reward Structure

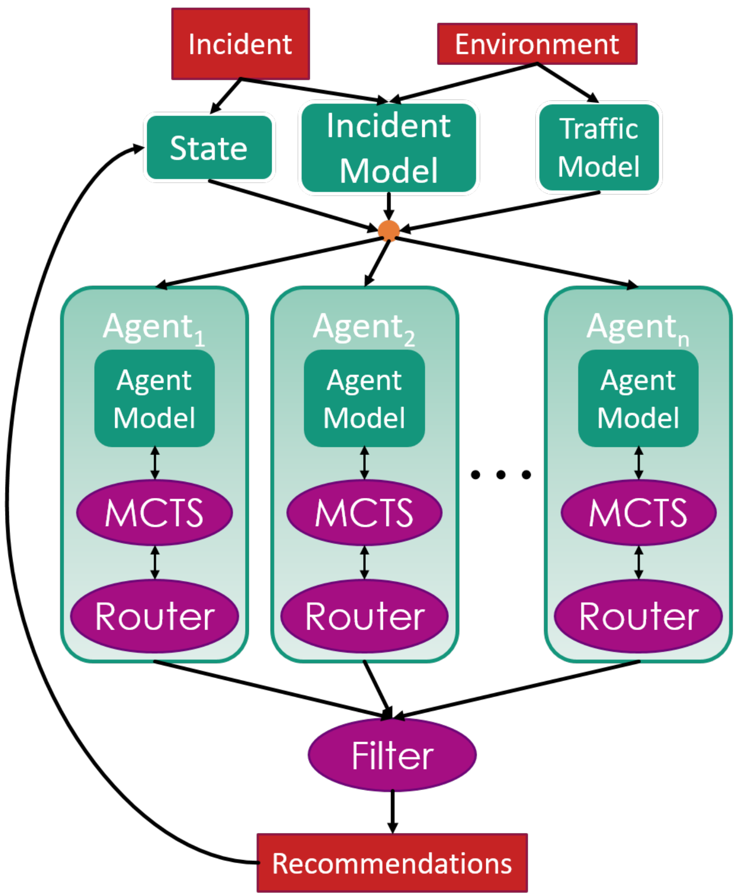
- Accounts for…
  - Incident dispatch -> *response time*
  - Balancing -> *distance traveled*

Discounted Travel Time

$$\rho(s,a) = \begin{cases} \rho_{s-1} - \boxed{\alpha^{t_h}(t_r(s,a)),} & \text{if responding to an incident} \\ \rho_{s-1} - \alpha^{t_h}\psi\,\dfrac{\sum_{\lambda_k \in \Lambda}(\phi_k(s,a))}{|\Lambda|}, & \text{if balancing at } s \end{cases} \tag{4a}$$

Approach 2: MMCTS

Partially Decentralized Decision Process



Reward Structure

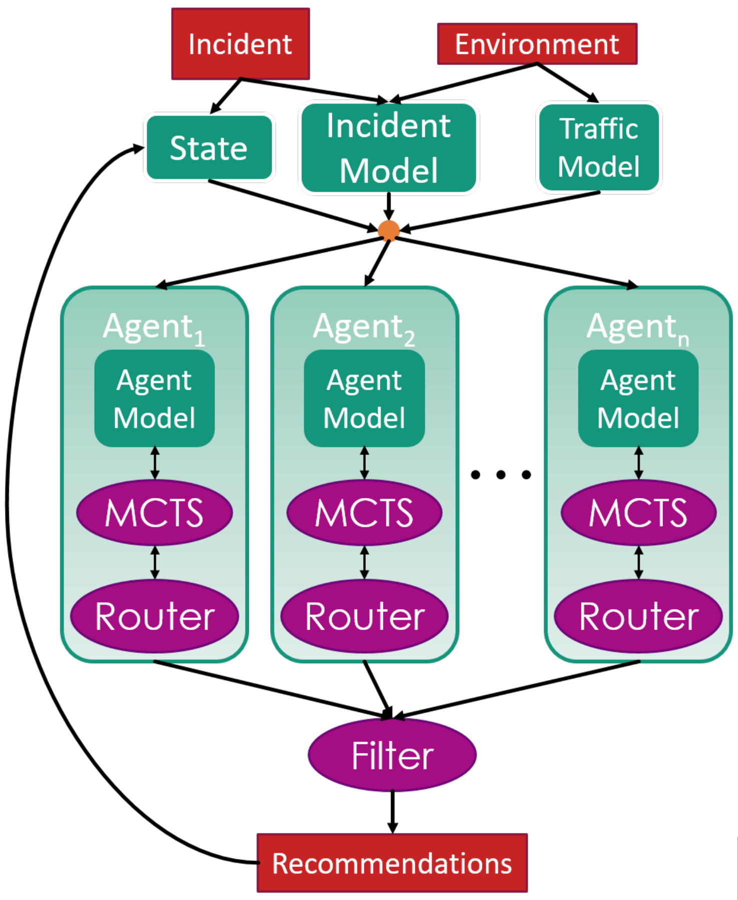
- Accounts for…
  - Incident dispatch -> *response time*
  - Balancing -> *distance traveled*

$$\rho(s,a) = \begin{cases} \rho_{s-1} - \alpha^{th}(t_r(s,a)), & \text{if responding to an incident} \\ \rho_{s-1} - \alpha^{th}\psi \boxed{\frac{\sum_{\lambda_k \in \Lambda}(\phi_k(s,a))}{|\Lambda|}}, & \text{if balancing at } s \end{cases} \quad (4a)$$

Average Distance Traveled For Re-balancing

Approach 2: MMCTS

Partially Decentralized Decision Process



Reward Structure

- Accounts for…
  - Incident dispatch -> *response time*
  - Balancing -> *distance traveled*

$$\rho(s,a) = \begin{cases} \rho_{s-1} - \alpha^{t_h}(t_r(s,a)), & \text{if responding to an incident} \\ \rho_{s-1} - \alpha^{t_h} \boxed{\psi} \frac{\sum_{\lambda_k \in \Lambda}(\phi_k(s,a))}{|\Lambda|}, & \text{if balancing at } s \end{cases} \tag{4a}$$

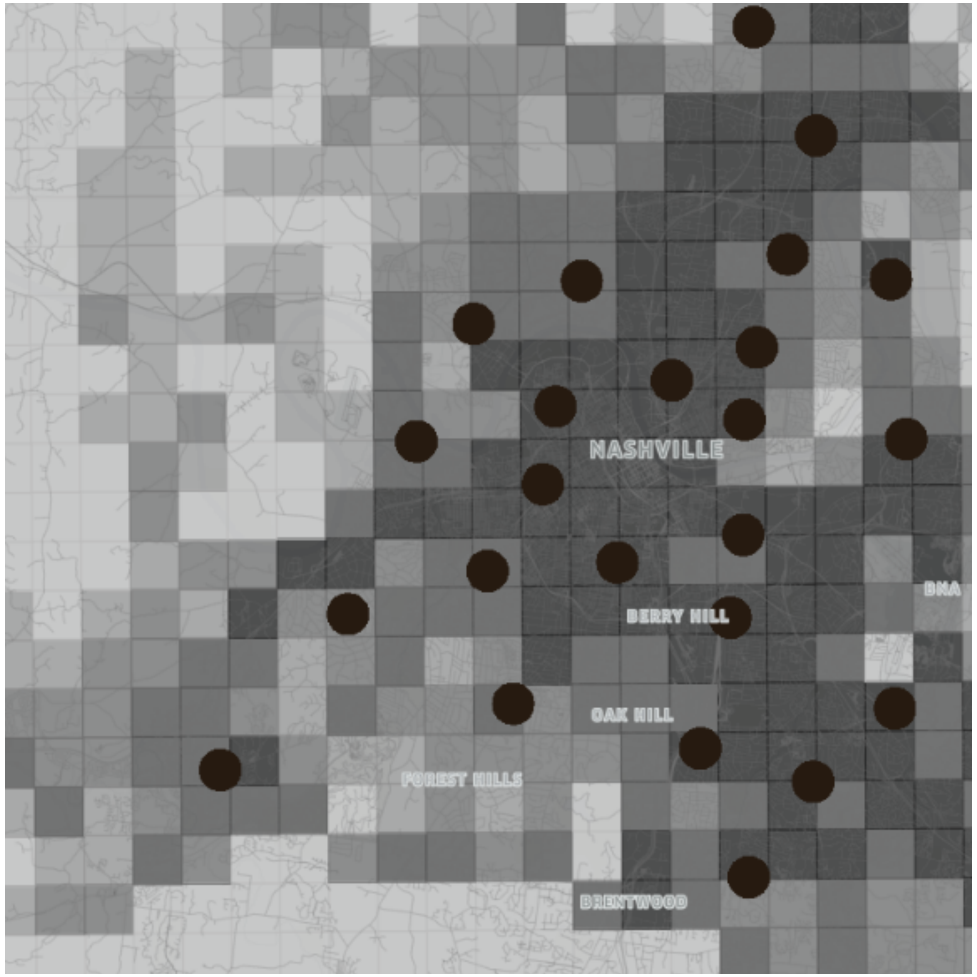
Average Distance Traveled For Re-balancing; [>0]

Experiments and Discussion

Experimental Configuration

Davidson County: Nashville Fire Department Administration Area

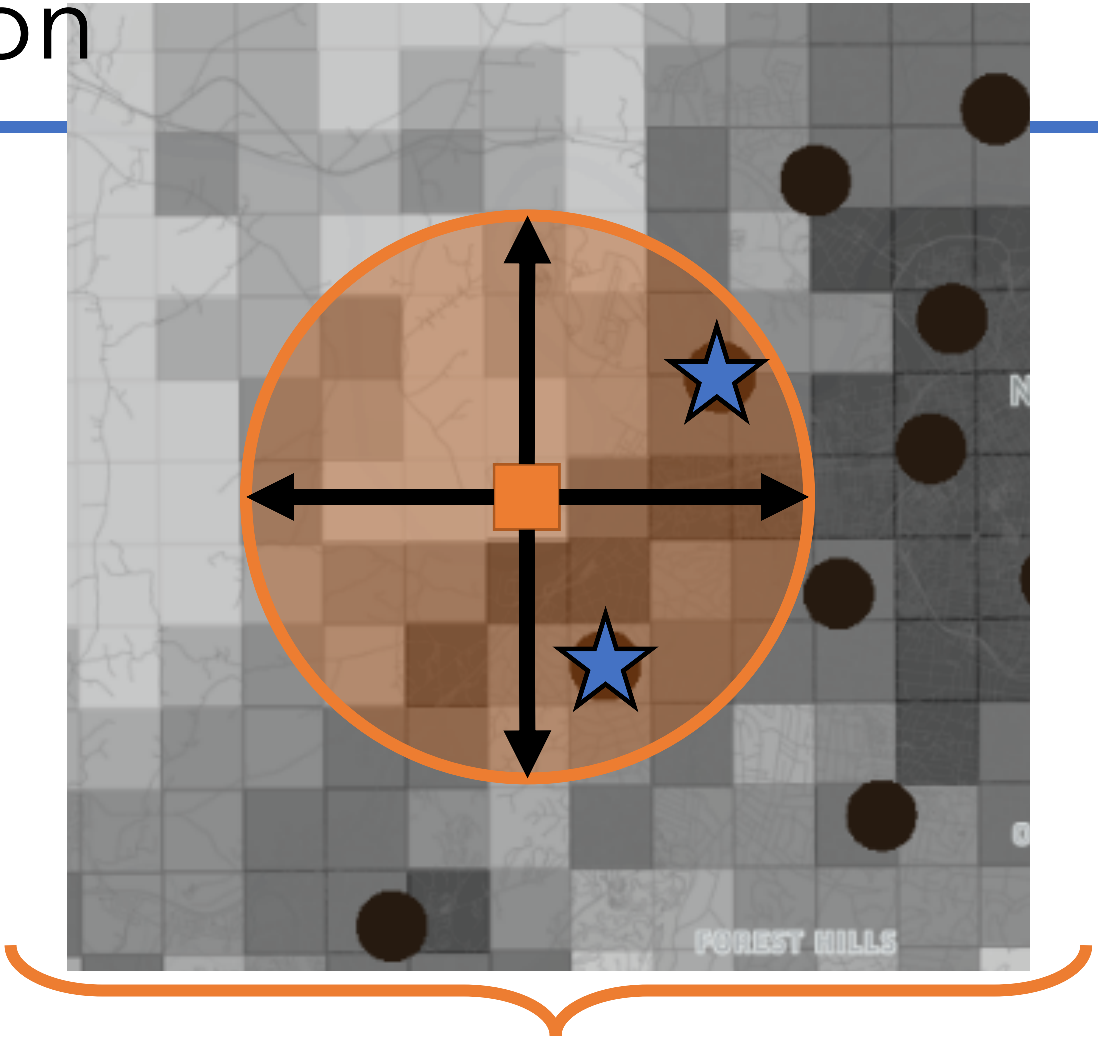
- 26 Responders (Agents)
- 36 Depots
- Incident model training set: 35858 traffic incidents occurring in 2018
- Entire system evaluated on 2728 incidents occurring in January, 2019



Experimental Configuration

Radius of Influence (RoI)

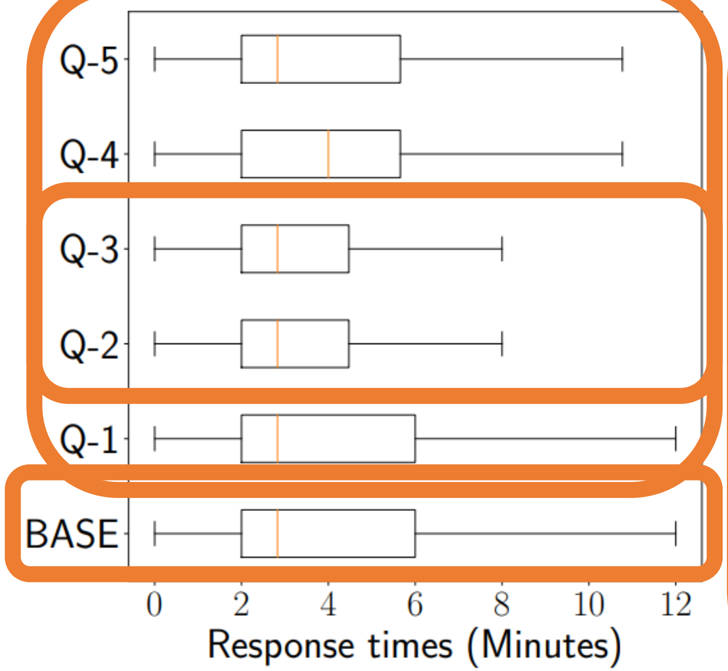
- Only depots within a cell's RoI are considered when splitting rates in heuristic score
- Encourages even agent distribution
- Reduces computation time

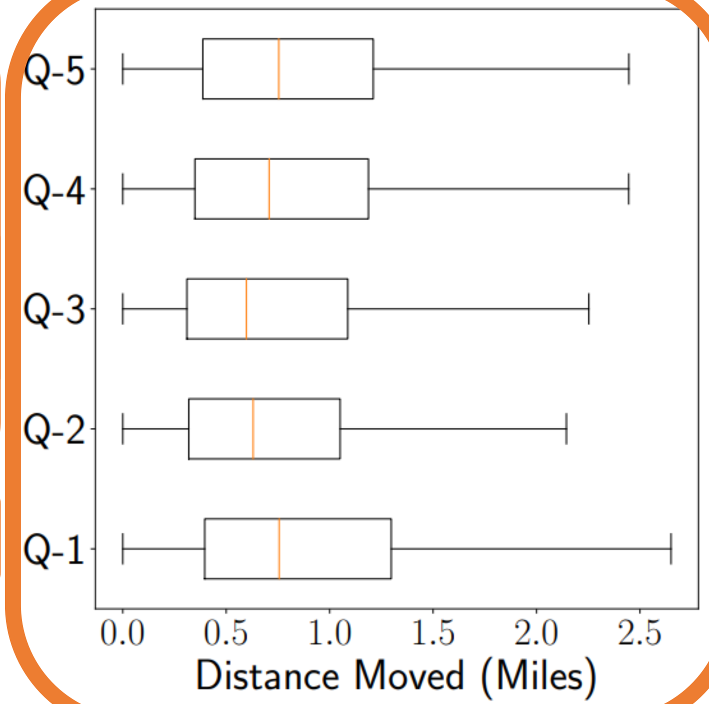


RoI := 3 cells

Results – Greedy Heuristic Search

BASE	Greedy Baseline Without Rebalancing	N/A
Q-1	Queue Based Rebalancing Policy with RoI of 1	RoI = 1
Q-2	Queue Based Rebalancing Policy with RoI of 2	RoI = 2
Q-3	Queue Based Rebalancing Policy with RoI of 3	RoI = 3
Q-4	Queue Based Rebalancing Policy with RoI of 4	RoI = 4
Q-5	Queue Based Rebalancing Policy with RoI of 5	RoI = 5



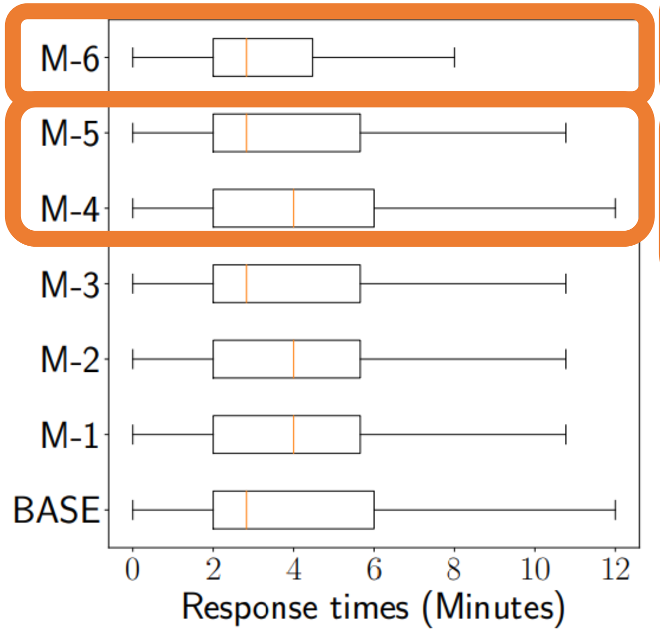


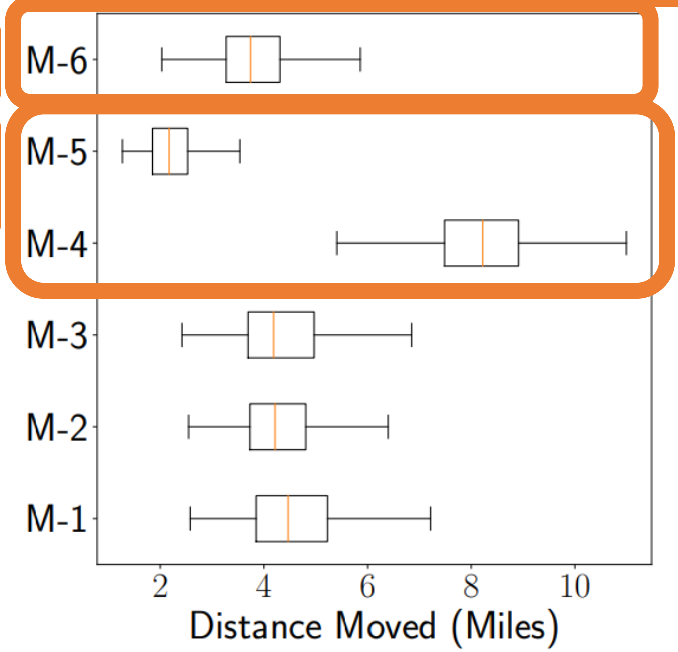
Observations

- Radius of Influence (RoI) has significant impact
- Best RoI => significant impact on tail of response time distribution
- <1 mile moved per rebalancing step on average

Results – MMCTS w/ Incident Model

M-1	MMCTS - Baseline The foundation for the parameter search. Each parameter varies independently while other parameters retain these values. (All M-* experiments use generated incident chains and a Static Agent Policy)	MCTS Iteration Limit = 250 Lookahead Horizon = 120 min Reward Distance Weight ψ = 10 Reward Discount Factor = 0.99995 Rebalance Period = 60 min
M-2	MMCTS - Iteration Limit of 100	MCTS Iteration Limit = 100*
M-3	MMCTS - Iteration Limit of 500	MCTS Iteration Limit = 500*
M-4	MMCTS - Reward Distance Weight ψ of 0	Reward Distance Weight ψ = 0*
M-5	MMCTS - Reward Distance Weight ψ of 100	Reward Distance Weight ψ = 100*
M-6	MMCTS - Rebalance Period of 30 minutes; Lookahead Horizon of 30 minutes	Lookahead Horizon = 30 min Rebalance Period = 30min*





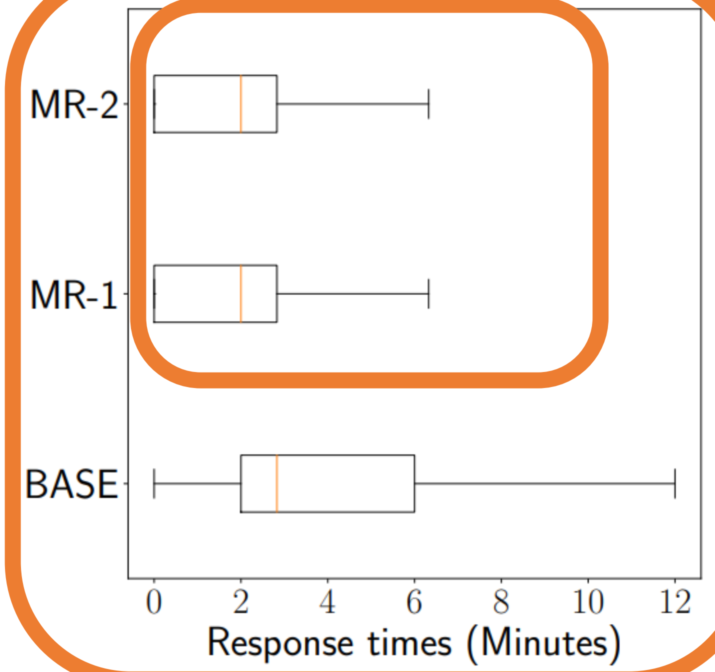
*Other hyperparameters same as M-1

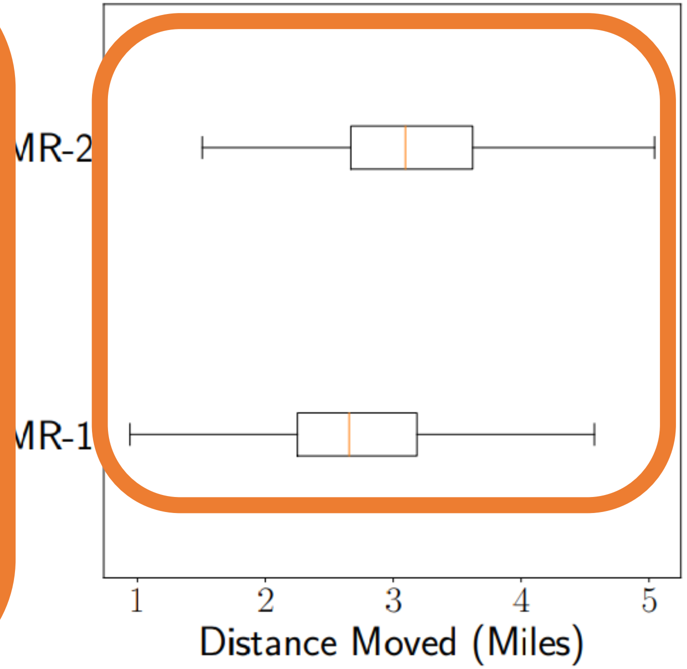
Observations

- Distance-reward weight -> large impact on amount traveled
- Lookahead Horizon and Rebalance Period -> impact on response time distribution

Results – MMCTS w/ Oracle

MR-1	MMCTS - using an oracle for future incidents and a Static Agent Policy	*Same as MMCTS Baseline M-1*
MR-2	MMCTS - using an oracle for future incidents and a Queue Rebalancing Policy	*Same as MMCTS Baseline M-1*

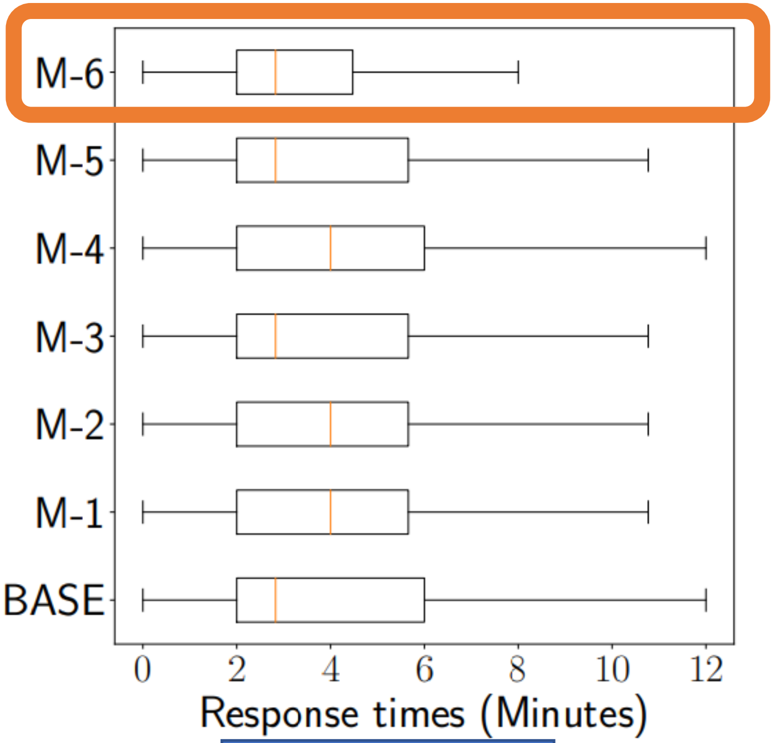




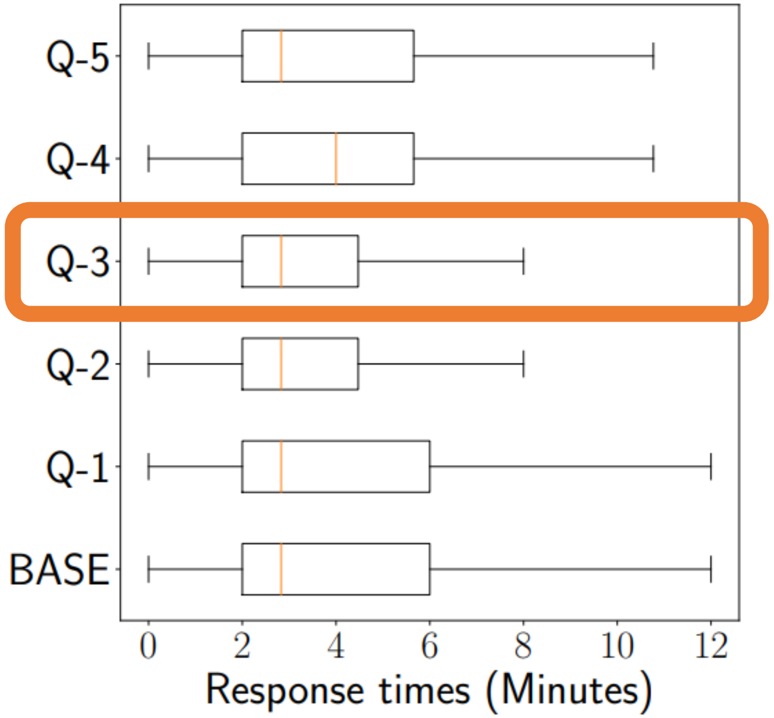
Observations

- Large *potential* improvement
- Despite increase in distance moved, Queue rebalancing shows little improvement over static
- More distance traveled than queue heuristic approach

Results – MMCTS vs Heuristic



MMCTS



Queue Heuristic

Observations

- Similar best-case performance for each approach

Key Takeaways

EMS Specific

Not feasible for real system

- Both approaches improve on baseline
- MMCTS w/ oracle demonstrates entanglement with efficacy of incident model
- MMCTS is more configurable than heuristic, but more sensitive to hyperparameter choices

General:

- Planning performance dependent on quality of underlying event prediction models
- Imperative to understand needs and constraints of target domain for it to be implemented
- Computational capacity of agents has evolved -> should use

Contact Info

- Presenter – Geoffrey Pettet:
  - Geoffrey.a.pettet@Vanderbilt.edu
- Collaborators:
  - Ayan Mukhopadhay
  - Mykel Kochenderfer,
  - Yevgeniy Voroybeychik,
  - Abhishek Dubey