Food Trucks AI

Bryce McLaughlin
Stanford Graduate School of Business, Stanford, CA 94305
brycem@stanford.edu

Wichinpong Park Sinchaisri
Haas School of Business, University of California, Berkeley, CA 94720
parksinchaisri@haas.berkeley.edu

March 22, 2025

We love food and we want more food trucks.

Key words:

---

1. Introduction

We design a location-based crowding game via a Dec-POMDP that allows us to explore human subjects interaction with recommendation algorithms that have superior information and influence subject both directly (through their recommendations) and indirectly (in the observations the new state-action evolution produces). Each location acquires demand according to an iid arrival process, while supply is determined by the actions of the decentralized agents. Supply reduces the demand at each location and rewards are earned based on utilization. The platform's 'goal' is to minimize average wait time using their recommendations.

2. Recommendations as Controllable Observations in Dec-POMDP

I'm hoping this is the final model that we iterate on

Dec-POMDP (Decentralized Partially Observable Markov Decision Processes) study a multiagent perspective on POMDP. In Dec-POMDP, many agents use their observations to take actions that maximize a long term reward without access to centralized knowledge or understanding of actions. Much of this literature has been advanced by robotic control researchers, studying how to design decentralized agent controllers that enable stable and near optimal solutions to the Dec-POMDP problem. We approach a different angle by trying to control a set of exogenous agents using a subset of the observation space that we control. Below we list the basics of Dec-POMDP, propose how to augment the structure for our model, and then develop a simple example. Much of this is based on/extends *A Concise Introduction to Decentralized POMDPs* by Oliehoek and Amato (copy in the drive).

2.1. Dec-POMDP

A Dec-POMDP is a tuple, $\mathcal{M} = \{\mathbb{I}, \mathbb{S}, \mathbb{A}, \rho_S, \mathbb{O}, \rho_O, R, s_0\}$, where,

- $\mathbb{I} = \{1, \ldots, n\}$ is the set of n agents affecting the dynamics of the process.
- $\mathbb{S}$ is the state space. It remains unknown to the agents at all times. At time t the state is $s_t \in \mathbb{S}$.
- $\mathbb{A}$ is the action space. It can be decomposed as $\mathbb{A} = \times_{i \in \mathbb{I}} \mathbb{A}_i$ where $\mathbb{A}_i$ is the action space of agent i. At time t each agent takes $a_{it} \in \mathbb{A}_i$ forming the joint action a_t.
- $\rho_S : \mathbb{S}^2 \times \mathbb{A} \to [0, 1]$ is the state transition function. Given a current state action pair of (s_t, a_t) the probability that $s_{t+1} = s'$ is $\rho_S(s'|s_t, a_t)$.
- $\mathbb{O}$ is the observation space. It can be decomposed as $\mathbb{O} = \times_{i \in \mathbb{I}} \mathbb{O}_i$ where $\mathbb{O}_i$ is agent i's observation set. At time t each agent observes $o_{it} \in \mathbb{O}_i$ which then informs how they take actions in future periods to maximize rewards (to be discussed shortly). Often times the process by which observations are utilized are described by finite state controllers. We can adapt this method by providing the controllers exogenously instead of making them part of our policy.
- $\rho_O : \mathbb{O} \times \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the observation transition function. Given a current state action pair of (s_t, a_t), the probability that $o_{t+1} = o'$ is $\rho_O(o'|s_t, a_t)$
- $R : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the (unobserved) reward function. Note that this reward function is global, and thus in Dec-POMDPs all agents collaborate toward the same goal. We will use this reward function as the principal (platforms) reward. We can use reduced forms to condense the preferences of agents into 'controllers' (explained below). Agents know the structure of R, and thus attempt to find state action paths which maximize it in the long run; however they do not observe R_t as it is informative of s_t.
- $S_0 \in \Delta(\mathbb{S})$ is the initial state distribution. Grounds the problem with a starting point which agents know and thus seeds their prior).

Methods for solving finite horizon Dec-POMDPs involve iterating to find optimal policy trees. Versions of these methods have been invented for the infinite case but instead of policy trees that evolve based on observations, agents are defined by finite state controllers (FSCs). FSCs keep track of an internal state and make actions based on the movement of that state. The state evolves deterministically according to (i) itself and (ii) observations in the current time period.

Using controllers to represent agents in a Dec-POMDP allows us to ignore game-like aspects as we assume some exogenous evolution of behavior from the agents. We can assume the controller structure of agents is unknown in its exact parameters, but that its general structure is known (ie they each come from some finite set or prior distribution), enabling us to imperfect interact with the controllers by manipulating the observations they control.

2.2. Recommendation Controlled Dec-POMDP (RC-Dec-POMDP)

For each i, let the observation set be a product $\mathbb{O}_i = \mathbb{O}_i^{sys} \times \mathbb{O}_i^{rec}$. The system observations $\mathbb{O}_i^{sys}$ are produced as before (according to the combination of state and action). The recommendation observations are controlled by the company via an algorithm $\pi : \mathbb{S} \to \mathbb{O}^{rec}$, where $\mathbb{O}^{rec} = \times_{i \in \mathbb{I}} \mathbb{O}_i^{rec}$

Instead of being built, assume each agent i is controlled by a finite state controller f_i from an exogenously specified potential controller set $\mathcal{F}$. We can either robustly optimize against controllers in $\mathcal{F}$, or we can assume $f_i \overset{\text{iid}}{\sim} \Psi$ where $\Psi \in \Delta(\mathcal{F})$. The transition conditions of each controller may depend on π.

One issue with this model is that the 'learning' of π by agents is exogenously specified in the controllers. They do not have an objective and do not learn to use the observations generated by π to improve their reward as part of this setup. Instead the controllers transition functions depend on π, representing some form of steady state learning. We could still test human agents in the experiment for how they learn but will need to generate recommendations using a model without learning if we go in this direction. Adding learning is particularly difficult as the agents understanding of the current π will influence how they use their observation to update their understanding of the current state s_t. Thus we also would need to model the agents prior and how it updates and impacts the FSC.

2.3. Simple Example

Below we describe an example problem using RC-Dec-POMDP which mirrors our experiment.

- $\mathbb{I} = \{1, \ldots, I\}$ are the agents, each has an associated controller f_i and home location h_i, which form the agent set.
- $\mathbb{A} = \{\emptyset, 1, \ldots, L\}^{\mathbb{I}}$ is the action set (thus $\mathbb{A}_i = \{\emptyset, 1, \ldots, L\}$). Each $a \in \mathbb{A}_i$ represents a location at which agent i could provide supply except for $\emptyset$ which represents the agent choosing to go inactive.
- $\mathbb{S} = \mathbb{I}^{\mathbb{A}} \times \mathbb{R}_+^L$ is the state. We shorthand $S_\ell \subseteq \{1, \ldots, I\}$ and $d_\ell \in R_+$ for $\ell \in \{1, \ldots, L\}$ as the supply and demand, respectively, at each location. We refer to the inactive agents as $S_\emptyset$.
- $\rho_S = \times_{\ell \in \{1,\ldots,L\}} \rho_{S(\ell)}$ describe the demand dynamics at each location in terms of that locations prior state and the action taken $a \in \mathbb{A}$. Note that a gives the new supply $S'_\ell = \{i \in \mathbb{I} | a_i = \ell\}$, which applies impact this period, but exists as information via the state into the next period. For each $\ell \in \{1, \ldots, L\}$,

$$\rho_{S(\ell)}(d'_\ell | d_\ell, a) = \begin{cases} 0 & \text{if } k_\ell < 0 \\ \frac{\lambda_\ell^{k_\ell} e^{-\lambda_\ell}}{k_\ell!} & \text{otherwise} \end{cases}$$

  where $k_\ell = d'_\ell - \min(0, d_\ell - |S'_\ell|)$ is the number of (incoming) arrivals, and λ_ℓ is a known location specific arrival rate. This turns each location into an M/D/n queue where n is determined

each round via the actions of the agents. WLOG assume $\lambda_L > \lambda_{L-1} > \dots \lambda_1$. This will give the agents reason to perfer one location over another if they have no information about either location aside from these rates.

- $R(s,a) = \sum_\ell \min(0, |S'_\ell| - d_\ell)$ keeps track of the cumulative waiting time of the demand across all locations. This can be computed as $|S'_\ell|$ is determined by a.
- $\mathbb{O}_i^{sys} = \mathbb{R} \times [0, I]^{2L+1}$. Observations in the system are given based on the agents action $a_i = \ell$. All observations are given at a highly detailed level, though naive agents may not use all observations (as determined by f_i their state controller). The first observation is $\hat{d}_\ell$ and is the demand at location ℓ in the period. The remaining observations describe the flow of servers at location $\ell \in \{1, \dots, L\}$. Specifically for any action $a \in \{\emptyset, 1, \dots, L\}$ the agents observe the flow in $\lambda_{a\ell}$ and the flow out $\lambda_{\ell a}$. Inactive agents observe this information at their home location h_i (ie $d_{h_i}, \lambda_{ah_i}, \lambda_{h_i a} \ \forall \ a \in \{\emptyset, 1, \dots, L\}$),
- ρ_O is here presented deterministically, but we can add noise if necessary,

$$\hat{d}_\ell = d_\ell \ \forall \ \ell \in \{1, \dots, L\}$$
$$\lambda_{\ell_1 \ell_2} = \sum_{i \in S_{\ell_1}} \mathbf{1}\left[a_i = \ell_2\right] \ \forall \ \ell_1, \ell_2 \in \{\emptyset, 1, \dots, L\}$$

- $\mathbb{O}_i^{rec} = \{0, 1, \dots, L\}$ is possible actions the platform can recommend where 0 is no recommended action. $\pi(s)$ gives the recommendation vector for any state s, where $\pi_i(s)$ is sent to agent i.
- In addition to the home state of each agent, $\mathcal{F}$ gives finite state controllers driving each agent. Each controller has a common structure with three internal states,
  - Ignore, where the agent takes action independently of the recommendations given.
  - Comply, where the agent to recommendations if given, but otherwise takes an independent action.
  - Inactive, where the agent is currently none operational and takes $a_i = \emptyset$ until there is sufficient demand in the home state.

  Transitions between these states determine are determined via a combination of the observations the agent receives and some agent specific parameters.
  - α_i^{in}, the agents movement response to other agents into their location. Can be positive or negative.
  - α_i^{out}, the agents movement response to other agents away from their location. Can be positive or negative.
  - β_i, the agents compliance response to other agents. Can be positive or negative.
  - $\tau_i^{\emptyset}$, the reservation threshold
  - τ_i^{C}, the compliance threshold

If the agent is inactive, they can only transition to the ignore state and take the action of their home location, or remain inactive. They take the action of their home state h_i iff there is enough local demand relative to supply, ie

$$\frac{d_{h_i}}{\sum_{\ell'} \lambda_{\ell' h_i}} \geq \tau_i^{\emptyset}.$$

Agents not in the inactive state determine their (independent) action from ℓ via the following function.

$$a_i^{\perp} = \arg\max_{a \in \{1,\ldots,L\}} \alpha_i^{in} \lambda_{a\ell} + \alpha_i^{out} \lambda_{\ell a} + \mathbf{1}[a = \ell] \left(\frac{d_\ell}{\sum_{\ell'} \lambda_{\ell'\ell}} - \alpha_i^{in} \sum_{\ell' \neq \ell} \lambda_{\ell\ell'} - \alpha_i^{out} \sum_{\ell' \neq \ell} \lambda_{\ell'\ell} \right)$$

Ties are broken by going to the location with higher incoming demand. They follow the action $a_i^{\perp}$ unless they are in the comply state and given a recommendation. Transitions between comply and ignore are determined by the compliance threshold, τ_i^C. Specifically you transfer to Ignore from either state if

$$\frac{d_\ell}{\sum_{\ell'} \lambda_{\ell'\ell}} + \beta_i \left(\sum_{\ell'} [\lambda_{\ell\ell'} + \lambda_{\ell'\ell}] - \lambda_{\ell\ell} \right) \geq \tau_i^C$$

and to Comply if the inequality holds in the other direction. These transitions and actions are superseded by transition/action combos to the Inactive state which occurs if

$$\frac{d_{h_i}}{\sum_{\ell'} \lambda_{\ell'\ell}} < \tau_i^{\emptyset}.$$

3. Theory of Multiagent Human-AI Interactions

When you change the recommendation, we're impacting the decisions.

Compared with Multiagent Bayeasian Persuasion.

Recommendation we send How humans respond to the recommendation (over time this is more learning) Recommendations are now being sent to other people / other observations they get

4. Experimental Studies

The agents get information from the state of the world (we cannot control that) and from us (we can control). How our recommendation affects their behaviors in changing environments.

- 1A: How do humans form future predictions? This is the sequential learning evidence. Manipulate different history.
- 1B: How do my future predictions affect my actions.
- 1C: How do humans respond to different types of tips

In context of model Q1 asks what does the set $\mathcal{F}$ look like

- 2A: How humans trade off between some objective info vs observations of peers' decisions (see how biased they are), fixing the objective info, varying the fraction of others who choose one option.
- 2B: Varying the level of what fractions of people having access to the AI tool
- 2C: Whether or not they get the same vs possibly different tip

  High level structural implications about π^* based on controller structure
- 3: Large-scale study putting everything together

  Process of learning f_i and optimizing π simultaneously.

Binary space for decisions.

4.1. Phase I: Learning Agents' Learning Behavior

We manipulate the paths agents observe and within each manipulation we can vary the observations of other agents. Typify people based on their strategies.

4.2. Phase II: Learning Agents' Responses to Recommendations

We now provide the tip as well as for some provide whether other agents have access to the same tool and whether others receive the same recommendation.

Insights from the first two phases, we incorporate the insights back to our POMDP and generate better recommendations.

4.3. Phase III: Improving the Recommendations

Proof of concepts

5. Concluding Remarks

When one person sends recommendations, it affects everyone. Multiple streams you have to account for