LEARNING REWARD SYSTEMS IN A MULTI-AGENT CROWD SIMULATION ENVIRONMENT

BARAK OSHRI

1. Introduction

Crowd simulation is the process of simulating the movement and state of a group of entities, whom when given an intelligence collectively become known as a swarm intelligence model. Crowd simulations are useful tools to model real-life behavior, such as in idealizing the swarm characteristics of organisms (ants, birds, and humans in various situations) and predicting the growth of a system of closed agents (like a virus). It has even been used in film and video game AI (e.g., the major battles in *The Lord of the Rings*). However, the behavior of crowds is also of major interest to sociologists and public safety officials, who must consider how groups of people will respond to environments where their natural goals conflict, coincide, or depend.

Many swarm intelligences that describe the behavior and self-organization of a group based on external or internal objectives have already been studied. What many of these models require though, and assume to have, is an explicit understanding of what is rewarding for the agent. For example, the oft-studied model for an ant network is parameterized by, among others, the global and local desire to optimize food allocation and energy expenditure. While many such models have specifiable reward schemes, little research has been done on the reverse: on how, only by retrieving data about the agent's utility with respect to its environment, the agent can learn to define the features that lead to pleasurable states.

2. Formal Model

This project will investigate a multi-agent crowd system in a 2d simultaneous game. Agents at each time step can move by a maximum of a given radius from their position (or choose not to move at all), in the hope of maximizing a utility function that defines their score. The utility varies with the position of any (or all) other agents in the game, and the agents must study the organization of agents across the game to learn the utility function. The reinforcement learning task then is to characterize and predict what the function is based only on data about the agent's utility and the positions of other agents, so as to maximize the expected value of their score.

The formal model is defined by:

(1) A finite set $A = \{A_1, A_2, ..., A_m\}$ of agents.
(2) A position for the agents $\mathbf{x}(A_i) \in \mathbb{R}^2$
(3) A utility function $U^{(i)}(\mathbf{x}(A))$ which represents agent A_i's utility as a function of the position of all other agents.
(4) A radius function $r^{(i)}(A_i)$. We restrict the movement of agent A_i so that $\|\mathbf{x}_{t+1}(A_i) - \mathbf{x}_t(A_i)\| \leq r_t^{(i)}(\mathbf{x}_t)$, where $\mathbf{x}_t$ denotes the position at turn t.
(5) A score $S^i(t)$ at a given turn, which represents the discounted sum of utilities for agent A_i (history of past rewards)

3. Approach

The model of a crowd system given above provides a suitable simulation framework. We can view the utility function as a representation of an "ideal organization" among the agents, such as keeping distance from others or rewarding clusters of a moderate size. The agents on the board will begin to self-organize based on the defined function by moving to positions with better-serving organizations. Since the utility function is the same for all agents, the agents are supporting each other without knowing it because they all implicitly aim to achieve the same optimal position.

The learning model is complex for several reasons. The agents must use epsilon-greedy strategies to explore the utility at different positions in the world. However, despite not knowing the utility of other

agents, they can assume that, to some level, other agents are learning too, and so they can use other agents' behavior as a guide to what actions to explore as the game progresses.

The main learning method is feature-based Q-learning, as it captures the idea that features about the organization of the world can be used to represent higher rewarding positions. Choosing the right features will be a key goal in this project. Neural network approaches may also be considered to model non-linear utility functions.

We view each action as deterministic, so SARSA or Q-learning methods need not take randomness into account. However, while the position of the agents is deterministic, the utility is dependent on the movement of others outside the agent's control, which can be interpreted as a pseudo-random process. The agents will predict the movement of others by assigning a continuous distribution around their positions, weighing stronger points with higher expected scores based on what the agent has learned about the utility function.

However, we need to be cognizant of the fact that the possible action set of the agents is infinite: they may move to any position within their radius. Later in the game when agents are settling into their strategies, the must solve the optimizations at the radius around their position, or define a heuristic that summarizes the organizational theme of the utility function (like move towards clusters of size greater than six, or keep a distance of 10 units from particles). Heuristics will be useful and necessary, as the agents must balance learning with pursuing optimal policies given that a discount factor is applied on rewards.

4. Preliminaries

4.1. **Baseline and Oracle.** The oracle, given the learning assignment is to estimate the utility function, is informing the agents of the utility function and running the simulation with that knowledge. This would lead to the optimal state in a non-cooperative model, and is the best we can expect in the game we defined.

The baseline is equipping the agents with a minimal intelligence: including basic features such as the average euclidean distance to all other points, indicator functions of a minimum or maximum distance away from other agents in the board, and discretizing the radius around the agent's position to simplify the task of choosing an action across a continuous range.

The gap between the oracle and baseline is sufficiently large because the baseline knows how to achieve an average score but cannot perform meaningful optimizations, while the oracle performs simple but strict optimization calculations at any given step.

4.2. **Evaluation.** The success of the learning model will be evaluated against the oracle using the score (summed utilities over the time steps considered). We can also evaluate the stability of the trial, where stability is measured as the average movement of agents in the late game. A stable configuration (less movement) indicates that the agents are confident about their policies and are happy about their position.

5. Previous work and possible directions

Multiple multi-agent simulation systems current exist for other sociological and entrepreneurial purposes. For example, the buyer-vendor relationship is extensively modeled in [Ter98], the study of the behavior of tourists is examined in [AN01], and the behavior of agents in dense crowds in [BNCM14]. Our model eschews specific domain knowledge in favor of generality. It is suspected that our system will not perform as accurately as those created to accomplish a specific purpose.

5.1. **Case Study: The Household.** An early example in this investigation is a utility function based on a household crowd system, which attempts to model how humans congregate into households.

Definition 1. *The household crowd system with m agents is defined by the following data:*

- *Let $N_i(k)$ denote the set of the k nearest neighbors to agent A_i. Define an approximate bump function centered at b via*
$$\Psi_b(x) = \frac{1}{1+(x-b)^2}.$$
Define a value function via
$$U^{(i)}(\mathbf{x}) = \sum_{j \in N_i(2)} \Psi_1(d(\mathbf{x}(A_i), \mathbf{x}(A_j))) - \sum_{j \notin N_i(2)} \mathbb{1}\{d(\mathbf{x}(A_i), \mathbf{x}(A_j)) < 10\}.$$
- $r^{(i)}(\mathbf{x}) = 1$ *for all i.*

The household crowd system encourages agents to congregate in "households" of 3 and discourages any two households from being too close to each other. The average scores have a 90% correspondence with the oracle.

Provided the board is sufficiently large (which is assumed), each agent can assume its optimal value 2.

References

[AN01] Klaus Auer and Tim Norris. "ArrierosAlife" a multi-agent approach simulating the evolution of a social system: Modeling the emergence of social networks with "Ascape". *J. Artificial Societies and Social Simulation*, 4(1), 2001. http://jasss.soc.surrey.ac.uk/4/1/6.html.

[BNCM14] Andrew Best, Sahil Narang, Sean Curtis, and Dinesh Manocha. DenseSense: Interactive crowd simulation using density-dependent filters. *The Eurographics / ACM SIGGRAPH Symposium on Computer Animation, SCA '14, Copenhagen, Denmark, 2014.*, pages 97–102, 2014. http://dx.doi.org/10.2312/sca.20141127.

[MMM13] Karam Mustapha, Hamid Mcheick, and Sehl Mellouli. Modeling and simulation agent-based of natural disaster complex systems. *Procedia Computer Science*, 21(0):148–155, 2013. http://www.sciencedirect.com/science/article/pii/S1877050913008144.

[Tan93] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

[Ter98] Pietro Terna. Simulation tools for social scientists: Building agent based models with SWARM. *Journal of Artificial Societies and Social Simulation*, 1(2), 1998. http://EconPapers.repec.org/RePEc:jas:jasssj:1998-4-1.

[Wik14] Wikipedia. Crowd simulation, 2014. http://en.wikipedia.org/wiki/Crowd_simulation.

E-mail address: boshri@stanford.edu