Levels of Realism for Cooperative Multi-agent Reinforcement Learning

Bryan Cunningham and Yong Cao

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24060, USA
{bcunn06,yongcao}@vt.edu

Abstract. Training agents in a virtual crowd to achieve a task can be accomplished by allowing the agents to learn by trial-and-error and by sharing information with other agents. Since sharing enables agents to potentially reach optimal behavior more quickly, what type of sharing is best to use to achieve the quickest learning times? This paper categorizes sharing into three categories: realistic, unrealistic, and no sharing. Realistic sharing is defined as sharing that takes place amongst agents within close proximity and unrealistic sharing allows agents to share regardless of physical location. This paper demonstrates that all sharing methods converge to similar policies and that the differences between the methods are determined by analyzing the learning rates, communication frequencies, and total run times. Results show that the unrealistic-centralized sharing method – where agents update a common learning module – is the most effective of the sharing methods tested.

Keywords: cooperative learning, multi-agent reinforcement learning, crowd simulation, 2D virtual world, inter-agent communication.

1 Introduction

Single-agent reinforcement learning (RL) has been widely studied over the past few decades [4]. Its extension to multiple agents that share a common environment is called multi-agent reinforcement learning (MARL) [1]. In the recent years MARL has been studied and adapted to work in the crowd simulation domain [8,2]. The overarching goal of crowd simulation is to realistically emulate the outward behaviors of its constituents, or agents, for the purposes of replicating physical actions and their resultant effects in an environment. Its applications include architectural and urban planning, evacuation planning, and video game and movie domains. The reason for adapting MARL to work within the crowd simulation domain is simple: since people learn and consequently adapt to situations using a form of reinforcement learning, why not apply this technique to train simulated computer agents to learn how to behave in a crowd? Naturally, because the agents are surrounded by other agents, it is logical to assume that they will come into contact with one another during simulation. The agents

Y. Tan, Y. Shi, and Z. Ji (Eds.): ICSI 2012, Part I, LNCS 7331, pp. 573–582, 2012.
© Springer-Verlag Berlin Heidelberg 2012

could benefit from communicating what they have learned to others. This paper focuses on the concept of inter-agent sharing within the crowd simulation domain for an evacuation scenario. It seeks to understand the impact that various methods of sharing have on the effectiveness of the agents' learning while exploiting the benefits of using a layered MARL architecture. Effectiveness is defined by resultant navigational behavior and total training time. Ultimately, this research serves as a case study to explore the most efficient ways to scale up to larger crowds in larger environments. Before delving into the specifics of the research, this paper will briefly discuss further reasoning for the necessity of this study followed by the essential background material relevant to understanding MARL, inter-agent sharing, and layered MARL. This paper attempts to explore how the intersection of these three domains and their application to the crowd simulation domain results in a previously unstudied research void.

2 Related Work and Motivation

RL is good at capturing individuality, or diversity, in an agent because each agent learns based on its own experiences within an environment. These experiences shape the decisions made by an agent, causing them to appear as if they have their own unique personality as they navigate through the environment. In the real world, people not only learn from trial-and-error exploration but also from each other through observational and/or verbal communication.

For purposes of this paper, realistic sharing is defined as taking place amongst agents within close proximity to one another. Other literature on sharing methods within the RL domain follow a trend in which inter-agent sharing is done via sharing across all agents, independent of agent location [7,3,9]. These papers use unrealistic methods of sharing and claim to be able to train the agents using fewer numbers of episodes, which is indicative of faster learning rates. It is important to note that faster learning rates do not necessarily imply faster total run times for agent training. For example, one learning rate could be faster than another but have a much higher communication overhead associated with it. During training this communication overhead could prove to dominate the total simulation training time. This type of method would therefore prove to be much more ineffective, computationally-wise, than a method that has a slower learning rate but a smaller communication overhead. This raises questions about the differences in the resultant navigational behavior between learning with realistic, unrealistic, and independent (no sharing) methods. This paper also seeks to identify viable, potentially faster, methods for training agents in the crowd simulation domain.

Tan [7] includes discussions on the communicational overhead associated with unrealistic and independent sharing methods. However, these discussions are limited in that the author analyzed communication overhead from a theoretical perspective only and discussed learning rate separately from communication overhead. As the realistic sharing methods presented in this paper allow agents to share on an inconsistent basis, we cannot perform static, theoretical analysis.

Instead performance tests measure learning rate, actual communication overhead and total running time for each sharing method. Learning rate and communication overhead need to be analyzed in conjunction with one another in order to classify the effectiveness of a method more accurately. Total method run time sufficiently captures the effectiveness of a method.

3 Background Concepts

3.1 Multi-agent Reinforcement Learning

RL is a bottom-up programming methodology that imbues agents with the ability to generalize learned information and extract salient environmental cues online. RL relies on the concept of Markov decision processes (MDP) to model how an agent moves around in the environment. An MDP is a 4-tuple taking the form $(S, A, P^a_{ss'}, R^a_{ss'})$ where S is the state space, A is the action set, P is the transition function where $P^a_{ss'}$ represents the probability of transitioning from state s to state s' via action a, and R is the reward function where $R^a_{ss'}$ represents the expected value of the reward achieved when an agent moves from state s to state s' via action a. As an agent explores its environment, it updates its policy function π that maps each state $s \in S$ and action $a \in A(s)$ to $\pi(s, a)$ which represents the probability of taking action a in state s. Agents define an action-value function for policy π by $Q^\pi(s, a)$ which indicates the expected return given that the agent takes action a in state s and then applies policy π. The agent attempts to maximize the expected total sum of rewards gained over time to converge to an optimal policy π^* (of which there can be multiple). For this paper, we use a simple and popular form of RL called Q-learning: a type of temporal-difference (TD) learning. TD learning is a learning technique in which an agent will update its previously estimated state values using the differences between its current and former values. This effectively propagates more accurate estimates of the state values as learning continues. Q-learning represents an off-policy form of TD control which, for the one-step case used in this paper, takes the following form:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad . \tag{1}$$

where t is the time step parameter, α is the learning rate parameter, and γ is the discount rate parameter. In this paper, the single-agent form of RL is applied to each agent in a multi-agent environment. Essentially, this means that agent learning takes place simultaneously and agents treat each other as independent, dynamic forces acting within the same environment.

3.2 Layered MARL Architecture

By default, RL and its extension to MARL, use one learning module to capture the learned policy of an agent. In a layered MARL architecture, an agent will use multiple learning modules to capture the learned policy of an agent [5].

Takahashi et al. [6] state that decomposing the control structure into smaller chunks, or learning modules, allows for the module policies to be transportable and applicable to new situations. To understand this, think about how an agent views a state in their state space. States are composed of multiple state parameters – any environmental information that the programmer wants the agent to consider when learning needs to be encoded in a state variable. For instance, a state variable could represent a position on the grid described by Cartesian coordinates or a facing direction described by a cardinal direction. Learning modules can be designed in such a way as to split up and group states' parameters for the purposes of decomposing the problem into separate logical units. These logical units can then be used to train separate learning modules within an agent. The learning modules work together to decide an appropriate action for the agent to take in a given state. This is advantageous because by decomposing the state parameters into logical components, learned behaviors can become more generalized and less dependent on other state parameters that may have no correlation. This helps reduce the total state space necessary to navigate, increases learning efficiency, and allows for sharing at the module level where data is less coupled by design.

3.3 Inter-agent Sharing

People can learn based on consciously or subconsciously observing and/or communicating various types of information based on the situation. Tan and Ribiero et al. [7,3] show that inter-agent sharing of sensation, policies and/or episodes decreases the steps necessary to reach optimal or good convergence points when compared to agents that did not share. This paper explores inter-agent sharing methods and classifies each as either realistic, unrealistic, or independent. With regard to this paper, sharing between two agents is classified as realistic only when the agents are sharing within communication distance of one another. This communication distance represents the range at which an agent is able to physically see or talk to another agent in the environment. Within this range, sharing would imitate realistically how people learn based on observing others or by verbally communicating information they might have learned.

When sharing using MARL, agents share learned information – for instance, in the form of Q-Values – from their policies and incorporate new information being shared with them into their existing policies. Unrealistic sharing indicates the broad range of sharing methods that are not based on reality as we have defined it above. In this case, sharing may not necessarily take place when agents are within one another's communication fields. The sharing method described earlier in which agents share regardless of their location on the map is an example of this. Another example of unrealistic sharing that will be encountered in this paper is centralized sharing. Centralized sharing enables agents to share a single Q-Value table and collectively contribute to and use its learning knowledge to make policy decisions. Note that realistic sharing methods are always localized sharing methods, but localized sharing methods are not exclusively realistic sharing methods.

4 Problem Statement

This paper will address the following questions: How do the sharing methods affect an agent's resultant navigational behavior? How do the sharing methods affect the learning rates of convergence, the communication overhead, and the overall training time for an agent? Can the layered MARL architecture be used to enable agents to successfully navigate both static and dynamic obstacles in the environment, in addition to finding and arriving at a goal location?

5 Approach

5.1 Agents, Environment, and Task

To explore the problems presented, a small-scale environment in which a simple evacuation simulation will take place was created. Agents attempt to evacuate to a common location using the most optimal path while encountering obstacles along the way. Agents traverse a discrete environment consisting of 7 x 7 cells. At each agent's turn, or step, agents are able to move Up, Right, Down, Left, or Stay in the current spot. There are two types of obstacles in the environment: walls (static) and other agents (dynamic). No two agents may be in the same cell as another, so if an agent attempts to move into an occupied cell, it will stay where it is. Similarly, if an agent attempts to leave the map or move into a space blocked by a wall, it 'bounces' off the map edge or wall and stays in place. Agents attempt to navigate to the same goal location and therefore are homogeneously oriented.

Testing occurs on 3 maps that are designed not only to explore the effectiveness of the sharing methods but also to investigate whether layered MARL is a feasible architecture within the crowd simulation domain. To do this, the three maps test dynamic obstacle avoidance, static obstacle avoidance, and a combination of the two, respectively. Map 1 tests dynamic obstacle avoidance because it contains no walls and focuses on agents learning to reach the goal in cell (6,3) while avoiding the other agents. Map 2 tests static obstacle avoidance because it contains a randomized placement of walls and disables agent-agent collisions. Map 3 tests both static and dynamic obstacle avoidance by reusing the same randomized map from Map 2 but turns agent-agent collisions back on. For all three maps, agents start in an assigned location. Agents 1, 2, and 3 start in positions (0,1), (0,3), and (0,5), respectively.

5.2 Sharing Methods

The following descriptions provide detail regarding the four variations of sharing methods that will be used in testing: (1) *Independent (No Sharing)*: Agents do not share any information with one another. (2) *Realistic-Localized Sharing*: For realistic-localized sharing, agents share the Q-values in their policies with other agents when they are within one another's communication fields. An agent's communication field is determined by the communication field size, where the

size corresponds to the depth of cells directly surrounding the agent. A field size of 1, for instance, would indicate that all cells directly around the agent's cell are part of the communication field. A cell C in a communication field must have an unobstructed line-of-sight, free of walls, to the agent the field emanates from; otherwise the agent will not be able to communicate with another agent that may be in C at the time. For both the realistic-localized and unrealistic-localized sharing methods, sharing is performed by using the frequency of state-action visitation to determine which agent has the most experience with that particular state-action pair. When this is determined, the Q-value for that particular state-action pair is synchronized to this *best* Q-value across all agents who are participating in the share. This continues for each possible state-action pair for both of the learning modules every time a share event occurs. Variables that are adjusted for testing this method are communication field size (CFS) and sharing step size (SS), where SS represents the minimum frequency, in terms of steps, with which an agent is allowed to share with another agent. (3) *Unrealistic-Localized Sharing*: Similar to realistic-localized sharing, except that sharing takes place uniformly at defined step sizes by all agents at once. Only the SS variable will be adjusted for testing this method as CFS is not applicable. (4) *Unrealistic-Centralized Sharing*: Agents update a shared, central Q-Value table and no explicit sharing occurs.

5.3 Layered MARL Implementation Details

This paper uses a simple two-module layered architecture: the pathfinder (P) and collision-avoidance (CA) learning modules. The pathfinder learning module's purpose is to find a path from the initial starting position to the goal position. The pathfinder module's set of actions are A_{P}={Up, Right, Down, Left, Stay}. The module's set of states S_{P} represent each cell position (x,y) on the map and therefore $|S_{\mathrm{P}}| = n \times m$ where n is the number of cells in the vertical direction and m is the number of cells in the horizontal direction. The collision-avoidance learning module's purpose is to avoid colliding with obstacles in the environment. The collision-avoidance module's set of actions are A_{CA}={Up, Right, Down, Left, Stay}. The module's set of states S_{CA} are represented by each unique permutation of the 8 cells directly surrounding the agent where each cell can be either empty (= 0) or not empty (= 1) for a total of $2^8 = 256$ states. These two learning modules work together because the pathfinder module determines an action and passes it to the CA module. Based on both the surrounding obstacles and the action suggested by the pathfinder module, the CA module then determines an action to take and instructs the agent to take that action. The pathfinder's module defines rewards as follows: Q_{P}: $(S_{\mathrm{P}} \times A_{\mathrm{P}}) \rightarrow \mathbb{R}$ where all actions that lead to a non-goal state receive a reward of -0.005 and all actions that lead to a goal state receive a reward of 1.0. The CA's module defines rewards as follows: Q_{CA}: $(S_{\mathrm{CA}} \times A_{\mathrm{CA}}) \rightarrow \mathbb{R}$ where an action in agreement with the action A chosen by the pathfinder module receives a reward of 0.005. An action that, oriented with respect to A, points to the side (left or right) receives a reward of -0.005. An action that, oriented with respect to A, points backwards receives a reward of -0.1. An action that, when A was any action but stay, was stay

receives a reward of -0.005. Finally, an action that, when A was stay, was any of the other actions other than stay, receives a reward of -0.005.

5.4 Experimental Setup

In this paper, agents operate using a discrete MDP in an environment conducive to episodes. Each episode is defined with an initial starting state and spans until a terminal, or goal, state is reached for each agent. A series of episodes define a simulation run. 150 simulation runs were tested for each sharing method in order to generate dependable data averages. An agent's learned policy was carried over from one episode to the next within a run, so ideally this results in convergence to an optimal policy as the agent learns more about the state-action space. In order to equally test the sharing methods we used an ϵ-greedy exploitation-exploration method with $\epsilon = 0.05$. The simulation was run until each sharing method converged to the same number of steps per episode – meaning that a path convergence point had been reached. The CFS parameter for realistic-localized sharing will vary and take the values of 1, 2, 3, and 4 where 1 represents a very limited communication field and 4 represents a fairly wide communication field. The SS parameter for both realistic and unrealistic-localized sharing will vary and take the values of 1, 5, 10, and 15. A SS value of 1 corresponds to agents being able to share their policies after a minimum of every step and a SS value of 15 indicates that sharing will occur less often, after a minimum of every 15 steps. The values for the simulation are set as follows: $\alpha_{\mathrm{P}} = 0.9$, $\gamma_{\mathrm{P}} = 0.8$, $\alpha_{\mathrm{CA}} = 0.2$, $\gamma_{\mathrm{CA}} = 0.8$.

6 Results and Contributions

Results were gathered across the 3 maps for the 3 agents. The resulting trends associated with Maps 1 and 2 were the same as the trends associated with Map 3, therefore only the findings from Map 3 will be presented. Similarly, overall data trends amongst agents agreed and only the findings for agent 1 will be shown. Results from the path convergence testing measure learning rate (in average steps per run), communication frequency, and total run time (in seconds) for each sharing method. Communication frequency measures every time a Q-table is shared with another agent – for example, sharing both a pathfinder and a CA Q-table count as two communication units because two Q-tables are shared.

Table 1 provides a detailed table containing results from all variations of the sharing methods used. Recall that faster learning rates (indicated by smaller learning rate numbers) signify that an agent has reached convergence in a fewer number of episodes. As expected, the methods that share the most have the lowest learning rates. Figure 1 illustrates differences in learning rates across a run for the variations of the four unique sharing methods with the best learning rates. The figure shows that the unrealistic-centralized sharing method significantly outperforms the other methods. This is understandable as the frequency of sharing, uniformity of sharing, and quality of sharing all increase as we move

from independent sharing to unrealistic-centralized sharing. To determine the overall computational differences between the sharing methods, especially with the realistic-localized method, we had to consider both learning rate and communication frequency in conjunction, which most clearly translated to a total run time.

Table 1. Learning rate, communication frequency, and total run time for the sharing methods

Sharing Method	Learning Rate (avg. steps)	Communication Frequency	Total Run Time (s)
Independent	95.88	0.00	0.01
Realistic-Localized:			
CFS = 1, SS = 1	61.31	361.13	0.75
CFS = 1, SS = 5	62.68	135.93	0.30
CFS = 1, SS = 10	63.79	98.71	0.22
CFS = 1, SS = 15	65.36	83.31	0.18
CFS = 2, SS = 1	59.87	829.76	1.59
CFS = 2, SS = 5	60.80	259.55	0.54
CFS = 2, SS = 10	62.51	166.60	0.36
CFS = 2, SS = 15	63.95	134.47	0.29
CFS = 3, SS = 1	59.07	1060.96	1.86
CFS = 3, SS = 5	60.33	321.05	0.61
CFS = 3, SS = 10	61.58	202.08	0.38
CFS = 3, SS = 15	62.18	158.24	0.30
CFS = 4, SS = 1	57.03	1166.88	1.99
CFS = 4, SS = 5	58.32	364.44	0.69
CFS = 4, SS = 10	60.55	233.88	0.43
CFS = 4, SS = 15	61.40	177.40	0.33
Unrealistic-Localized:			
SS = 1	49.53	8641.49	7.06
SS = 5	50.96	1776.64	1.48
SS = 10	51.29	852.56	0.72
SS = 15	52.78	577.79	0.50
Unrealistic-Centralized	45.01	0.00	0.01

Performance timing for the total run times provides a general picture of how effective each method is with regard to one another. Figure 2 depicts the differences in time per episode across a run for the variations of the four sharing methods with the fastest total times. Both the independent and unrealistic-centralized sharing methods performed equally well as no sharing takes the same amount of time that implicit sharing does.

The realistic-localized and unrealistic-localized methods perform much more poorly, on the order of 19x and 51x more slowly, respectively. Overall, with respect to both learning rate and total run time, the unrealistic-centralized sharing method is the most effective method tested in this experiment. Realistic sharing

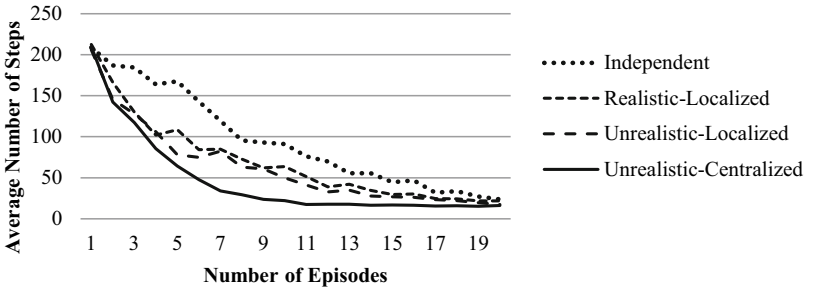

Fig. 1. Average number of steps vs. number of episodes for the sharing methods

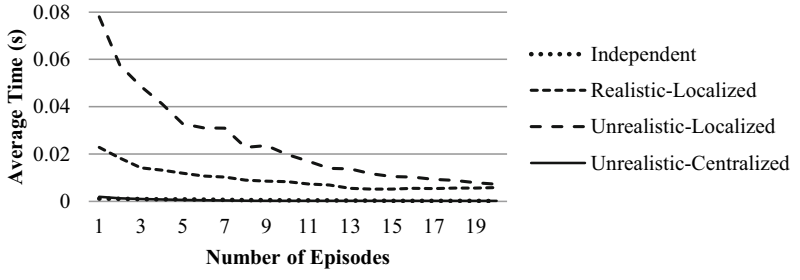

Fig. 2. Average time vs. number of episodes for the sharing methods

limits the rate at which learning can take place because sharing takes place less frequently and less uniformly. Localized sharing in general does not allow for policy information to be as readily absorbed as it is with centralized sharing – not to mention the explicit communication overhead present that overwhelms the training time for the agent. It is important to mention that while centralized sharing is the best choice for the purpose of efficiency, simulations that study communication flows in crowds, for example, must be done using realistic-localized methods. This type of method preserves communication fidelity and ensures accurate inter-agent communication patterns.

7 Conclusions and Future Work

This paper demonstrates that all sharing methods converge to the same policies. The differences between realistic, unrealistic, and independent sharing methods are determined by analyzing the learning rates, communication frequencies, and total run times. The unrealistic-centralized sharing method proved to be the most effective of the sharing methods tested. Finally, testing showed that agents successfully navigated both static and dynamic obstacles in the environment, in addition to finding and arriving at a goal location. This research opens up the possibility to study more detailed problems concerning applying a layered MARL architecture within the crowd simulation domain. For instance, how does

adding more learning modules to each agent affect the complexity of the state-action space? Additionally, how well do the sharing methods presented in this paper port to the GPU in order to support the simulation of more agents in a more finely discretized environment?

Acknowledgements. This work is partially funded by National Science Foundation, IIS 0940723, titled EAGER: Drummer Game: A Massive-Interactive Socially-Enabled Strategy Game".

References

1. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(2), 156–172 (2008)
2. Martinez-Gil, F., Lozano, M., Fernández, F.: Multi-agent Reinforcement Learning for Simulating Pedestrian Navigation. In: Vrancx, P., Knudson, M., Grześ, M. (eds.) ALA 2011. LNCS, vol. 7113, pp. 54–69. Springer, Heidelberg (2012)
3. Ribeiro, R., Borges, A., Enembreck, F.: Interaction models for multiagent reinforcement learning. In: 2008 International Conference on Computational Intelligence for Modelling Control Automation, pp. 464–469 (2008)
4. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
5. Takahashi, Y., Asada, M.: Multi-controller fusion in multi-layered reinforcement learning. In: International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 7–12 (2001)
6. Takahashi, Y., Asada, M.: Behavior acquisition by multi-layered reinforcement learning. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, pp. 716–721 (1999)
7. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the Tenth International Conference on Machine Learning, pp. 330–337. Morgan Kaufmann (1993)
8. Torrey, L.: Crowd simulation via multi-agent reinforcement learning. In: Youngblood, G.M., Bulitko, V. (eds.) Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. The AAAI Press (2010)
9. Zhang, P., Ma, X., Pan, Z., Li, X., Xie, K.: Multi-Agent Cooperative Reinforcement Learning in 3D Virtual World. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010, Part I. LNCS, vol. 6145, pp. 731–739. Springer, Heidelberg (2010)