Online Learning for Multi-Agent Local Navigation

Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini

Department of Computer Science and Engineering, University of Minnesota
godoy@cs.umn.edu,ioannis@cs.umn.edu,sjguy@cs.umn.edu,gini@cs.umn.edu

Abstract. In this paper, we present an adaptive framework for improving the local navigation in multi-agent simulations. Our framework uses techniques from Reinforcement Learning allowing the agents to adapt their behavior *online* while the simulation is still running. It is general and can be easily combined with existing schemes for local navigation. As an example, we demonstrate its application using the *Optimal Reciprocal Collision Avoidance* method introduced in robotics. Preliminary results show that, using our adaptive approach, the agents exhibit a more polite and respectful behavior toward each other, as compared to approaches that do not use learning. This alleviates congestion phenomena observed in non-adaptive local navigation methods and helps the agents reach their goal destinations faster.

Keywords: Crowd simulation, multi-agent navigation, online learning, reinforcement learning

1 Introduction

Over the past two decades many new methods have been proposed for modeling the dynamics of agents moving together in virtual environments. Such techniques have important applications, such as guiding the motion of characters in video games, simulating human crowds for planning and analysis, and coordinating teams of robots. Our work seeks to extend these types of models by incorporating principles from *Reinforcement Learning* (RL) into local navigation methods for agents in virtual environments.

In this work, we assume that each entity in the virtual environment can be simulated with a *Belief-Desire-Intention* (BDI) Agent. That is, an agent is viewed as a dynamic entity that makes independent decisions. These agents have distinct characteristics and goals and can plan their own movements individually by continuously perceiving their surroundings and (re)acting accordingly. We also assume that the desired global motion of each agent is predetermined and thus focus on improving the local behavior of how agents interact with each other.

The local interaction and navigation problem between agents moving in virtual and physical environments is a well studied problem. Recent advancements have significantly improved the ability of the agents to interact with each other

and the environment in a realistic fashion. For example, current state-of-the-art techniques can guarantee collision-free motion between agents as they reach their goals [1]. However, for truly intelligent behavior, the agents need to move beyond simply avoiding collisions and display the ability to adapt intelligently to their changing environments.

Consider, for example, a number of agents trying to pass through a narrow passage (e.g., Fig. 2b in Section 6). As new agents enter the scene, congestion conditions evolve in front of the bottleneck, due to the fact that the agents are continually competing for the same free space. In these type of scenarios, humans have the ability to adapt their behavior based on the situation at hand. For example, many people will let the congestion dissipate before trying to enter the passageway, saving energy and increasing the overall flow rates. It is this type of intelligent reaction to one's environment which we seek to reproduce in simulated agents.

In this paper, we present a framework for improving the local navigation of agents in crowd simulations by incorporating techniques from Reinforcement Learning. Our main contribution is that, in our method, agents adapt their behavior *online* while the simulation is running. As a result, the agents are implicitly rewarded for cooperation which leads them to exhibit a more "polite" behavior toward each other, in comparison to non-adaptive agents. This alleviates congestion phenomena observed in non-adaptive local navigation methods and lets the agents reach their goal destinations faster. Our technique can be easily combined with many existing schemes for local collision avoidance. As an example, we demonstrate its application using the *ORCA* local navigation technique proposed in [1].

The rest of the paper is organized as follows. Section 2 provides an overview of prior work related to our research. In Section 3, we provide a more formal definition of the local navigation problem and highlight the potential role of learning approaches. A full explanation of our proposed framework is presented in Section 4, and details of our learning approach are given in Section 5. We review our experimental results in Section 6. Finally, some conclusions and plans for further research are discussed in Section 7.

2 Related Work

In this section, we highlight some of the most relevant work in crowd simulation, local navigation and reinforcement learning. We refer the interested readers to the excellent survey of Pelechano et al. [2] and Buşoniu et al. [3] for a more comprehensive discussion.

2.1 Crowd Simulation and Local Navigation

Numerous models have been proposed to simulate individuals, groups and crowds of interacting agents. The seminal work of Reynolds on *boids* has been influential in this field [4]. Reynolds used simple local rules to create visually compelling

flocks of birds and schools of fishes. Reynolds later extended this model to include autonomous reactive behavior [5]. Since his original work, many interesting crowd simulation models have been introduced that account for cognitive and behavioral rules [6, 7], sociological or psychological factors [8–10], and biomechanical principles [11].

An extensive literature also exists on modeling the local dynamics of the agents and computing a collision-free motion among static and/or dynamic obstacles. Over the past twenty years many different agent-based techniques for local collision avoidance have been proposed in control theory, traffic simulation, robotics and animation. These include *force-based* approaches which treat agents as particles and model their interactions using physical forces [5, 12, 13], *vision* techniques which combine visual stimuli with motor response laws to adapt the agents' velocities and resolve collisions [14], and *geometrically-based* algorithms which compute collision-free velocities for the agents using either sampling [15–17] or optimization techniques [1, 18]. Our framework also uses a well-known geomatrically-based algorithm for local navigation, and enhances it with an online learning component, that allows agents to improve their behavior while the simulation is running.

2.2 Multi-Agent Reinforcement Learning

Reinforcement Learning (RL) is a popular machine learning technique which addresses how autonomous agents can learn to act in an environment in order to achieve a desired goal [19]. A RL agent performs actions that affect its state and environment, and receives a reward value indicating the quality of the performed action and state transition. This reward is used as feedback for the agent to make better future decisions. Algorithms for agent Reinforcement Learning have been widely used for applications as diverse as robotics [20], game theory [21], scheduling [22], path-planning [23], and many others.

A well known family of RL algorithms is based on the Temporal Difference (TD) prediction approach introduced in [19], which is a combination of Monte Carlo sampling methods and dynamic programming for learning in environments that are formulated as Markov Decision Processes. The most well known of these algorithms is Q-learning [24], in which agents learn an action-value function that estimates the expected reward of choosing a specific action in a given state and following a fixed policy afterwards.

Different approaches have also been proposed to incorporate learning when multiple agents are present in a simulation scenario (see [3, 25] for an extensive overview). These multi-agent learning approaches originate both from extensions of RL algorithms and applications of game theory. However, most of them are focused on small problems and/or a very limited number of agents. RL algorithms need to collect information on how all agents behave in order to find a policy that maximizes their reward. This is expensive when the state space is too large and requires a significant degree of exploration to create an accurate model for each agent. Game theory algorithms are focused on proving properties of convergence and equilibrium requiring the agent to have previous knowledge

of the structure of the problem. An example of the application of both reinforcement learning and game theoretic principles for multi agent coordination can be found in [26].

In direct policy search methods, agents perform stochastic optimization based on the reward associated with each action [3]. These methods do not require each agent to maintain a complete state information on other agents, and as such, are more applicable to our work where we are targeting real-time environments with limited computational resources. A successful variant of Direct Policy search methods is the family of algorithms that use the WoLF principle ("Win or Learn Fast") [21], which varies an agent's learning rate based on recent rewards.

Finally, work has been done in learning and adapting motion behavior for agents in crowded environments [27]. This approach uses inverse reinforcement learning to enable agents to learn paths from recorded training data (example traces of virtual characters moving in a simulated crowd). Similarly, the approach in [28] applies Q-learning to plan paths for agents in crowds, placed on a small grid. In this approach, agents learn in a series of episodes the best path to their destination. A SARSA-based [29] learning algorithm has also been used in [30] for offline learning of behaviors in crowd simulations.

3 Problem Description

The main objective of this work is to introduce online learning techniques in multi-agent navigation so that the agents can exhibit a more intelligent collision avoidance behavior. In a typical multi-agent navigation problem, we are given a virtual environment containing static obstacles and n heterogeneous agents A_i $(1 \leq i \leq n)$ with specified start and goal positions. The task is then to steer each of these agents to its goal without colliding with the other agents and the obstacles present in the environment. We also require that the agents navigate *independently* without explicitly communicating with each other.

For simplicity, we assume that each agent A_i moves on a plane and is modeled as a disc with radius r_i. At a fixed time instant t, the agent A_i is at position $\mathbf{p}_i$, defined by the (x, y) coordinates of the center of the disc, and moves with velocity $\mathbf{v}_i$ that is limited by a maximum speed $v_i^{\max}$. Furthermore, at every simulation step, A_i has a preferred velocity $\mathbf{v}_i^{\text{pref}}$ directed toward the agent's goal with a magnitude equal to the speed v_i^{pref} at which the agent prefers to move. We assume that the radii, positions and velocities of other nearby agents can be obtained by local sensing. Given this formulation, the goal of a local navigation method is to independently compute for each agent a new velocity that is close to its preferred one and avoids collisions with obstacles or other agents. The agent adopts the new velocity for the next simulation cycle and updates its position accordingly.

In our current framework, we use the RVO2-Library for the local navigation of the agents. This library uses the *Optimal Reciprocal Collision Avoidance* (ORCA) formulation which allows each agent to select a reciprocally collision-avoiding velocity by solving a low dimensional linear program [1]. ORCA can

guarantee collision-free motions for the agents as they move towards their goal. However, like in any local navigation method, the parameters that capture an agent's internal state (i.e., the preferred speed v^{pref}, separation distance ρ, etc.) remain fixed over the entire simulation leading to unrealistic motions. Consider, for example, the preferred speed v^{pref} of an agent. It is only assigned once, at the beginning of the simulation, and does not change even when an agent encounters challenging and time-consuming situations. Ideally, though, an agent should be able to adapt its preferred speed based on the situation that it faces, the behavior of the other agents, the complexity of the environment and so on. This adaptation could be partially encoded in rules, however, this approach lacks flexibility and scalability. In contrast we aim for a Reinforcement Learning (RL) based approach as a natural choice for improving the behavior of the agents by allowing them to adapt to the scenario at hand.

However, there are certain challenges in the application of RL algorithms to multi-agent navigation. First, in our problem setting, we are interested in agents that can learn *online*, while the simulation is running. Furthermore, even if the agents are able to learn an optimal strategy in a given environment, this strategy may not be equally successful in different scenarios and thus, the agents should be trained again. Also, agents in our problem are staged in a 2D continuous state space, which prevents simple state-action algorithms from being directly applied. Although there are approaches that deal with this type of state space by discretizing it, they involve tradeoffs that do not align with our objective in this work. Finally, an agent's strategy for adapting to the environment may depend on several factors including the full state of all the other agents. However, this can be computationally expensive and memory intensive. Existing algorithms to model and track other agents' actions and states are mainly applicable to small problems with a very small number of agents [3].

Consequently, given these issues in applying traditional RL and rule-based approaches, we intend to use an alternative adaptive learning method.In our approach, agents do not learn a policy for the complete state-action space, and do not store the values for each instance of this space. Instead, our proposal is to update an agent's beliefs across the available actions in the current state using the recent history of action-reward pairs and feedback from the simulation. This way, agents learn to adapt ther policies to dynamic environmental conditions on the fly. We adapt a method inspired by the popular WoLF ("Win or Learn Fast") policy hill climbing algorithm (WoLF-PHC) [21] which involves a variable learning rate for the agent given the feedback it receives after its previous action. Hence, each agent keeps an updated probability distribution on the actions, and stochastically picks one based on these probabilities.

In Section 5, we introduce a streamlined adaptive learning method which works on the reduced state space consisting of an agent's preferred speed and the resulting velocity from ORCA. If more computational resources are available per agent, a more complex learning approach may be preferable. Additionally, while an agent's true state space is continuous, we discretize its potential actions

into a finite number of choices. It may be possible to apply approaches such as Tile Coding [29] to overcome this limitation.

4 Online Adaptive Framework

In our adaptive framework, at every time step of the simulation, each agent in the environment stochastically selects and performs an action that changes its state and provides an appropriate reward. More formally, we define:

Environment A 2D space consisting of agents and static obstacles.

State A specific set of parameter values for the agent. An example state would consist of the agent's position, current velocity and preferred velocity.

Actions A discrete set of actions the agent can choose from. In the example state described above, the agent can either speed up, slow down (by specified values δ_{inc} and δ_{dec} respectively), or keep the same preferred speed. See Sect. 5 for more details.

Goal The goal position of the agent.

Reward A numerical value that represents the progress the agent has made towards its goal.

We refer the reader to Fig. 1 for a schematic overview of our framework. As can be observed, the agents use feedback received from the environment to make decisions about actions that influence their behavior (for example, increase or decrease their preferred speed), with the belief that the chosen action will improve their collision avoidance strategy and help them achieve their goals as fast as possible. Hence, the traditional sensing-acting cycle for an agent is defined as follows:

1. Agent decides on a certain action.
2. Agent performs the action, which changes its state.
3. Agent receives feedback or reward for performing the chosen action.
4. Agent uses the feedback to update its beliefs, which in turn affect its future decisions.
5. Back to Step 1.

Finally, we should point out that, in our framework, we are not interested in agents that reach a convergence in their behaviors, as our goal is to make them adaptable to environmental changes, even if this implies that sometimes they will choose suboptimal actions. In contrast, we are mainly interested in how the decisions of the agents at each simulation step affect both their individual behavior as well as the aggregate behavior of the entire crowd. Do agents account for congestion when they are more reactive to external conditions? Do they actually reach their destinations faster?

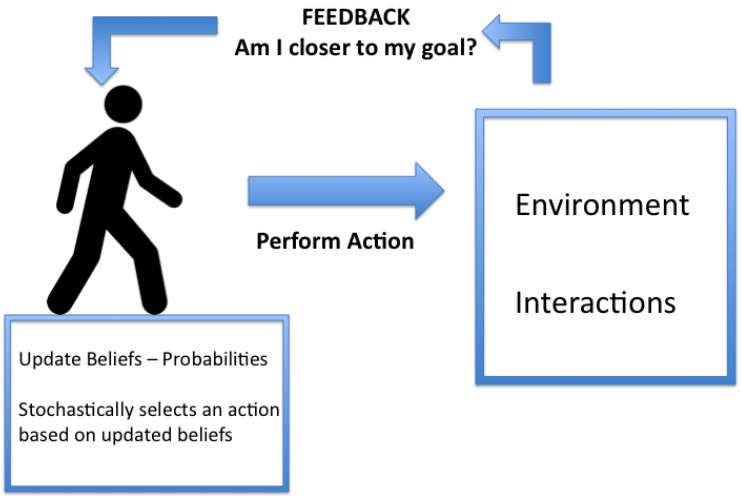


Fig. 1: A schematic overview of our framework. A continuous cycle of sensing and acting is run, allowing the agent to adapt its behavior based on the feedback received from the environment after performing a certain action.

5 Learning Approach

In this section, we describe a simple learning approach that allows an agent to adjust the value of its *preferred speed* in order to reach its destination as quickly as possible, while adapting to dynamic environmental conditions. Our learning method works on a reduced state space consisting of the agent's preferred speed and the new velocity that ORCA computes for the agent at each time step of the simulation.

Given this reduced state space, we consider the following actions that the agent can perform:

Speed Up Increase the preferred speed by δ_{inc}.
Slow Down Decrease the preferred speed by δ_{dec}.
Keep Speed Maintain current preferred speed.
Stop Stay in the current position for the next step.
Go Back Move in the opposite direction of the goal.

Stochastic action selection

The agent keeps a probability distribution on the available actions and chooses one based on stochastic policy hill climbing. Initially, because the agent has not

explored the environment and does not know the consequences of its actions, every action a has the same probability of being picked, that is:

$\forall i \in I : P_{a_i} = \frac{1}{|I|}$, where I denotes the set of all available actions for the agent.

Feedback and reward

After performing the action, the agent receives as feedback from the ORCA algorithm a new velocity that needs to take in order to avoid collisions with other agents and static obstacles present in the environment. The agent's reward is computed based on this feedback by projecting the ORCA velocity to the agent's normalized preferred speed. This allows us to determine whether and how much the agent has progressed toward its goal after performing its selected action.

Variable learning rate

To measure how 'profitable' was the undertaken action, the agent compares its reward with the reward that it received at the previous time step. It then updates its action probability distributions based on the concept of *Variable Learning Rate* and the *"Win or Learn Fast"* ($WoLF$) principle introduced in [21] for learning in competitive environments.

In [21], the authors use two *learning rates* to update an agent's beliefs based on the outcome of the last interaction so that the agent can "learn quickly while loosing and slowly while winning". Specifically, a WoLF-based agent uses a larger learning rate to adapt quickly in low reward situations, and a smaller learning rate to cautiously adapt in high reward situations. Instead, our adaptation employs a different learning strategy. We aim at using a variable learning rate to maximize the probability of choosing a profitable action and slowly decrease the probability of an action that is less beneficial to the agent.

The rationale behind this is that when an agent is slowed down to resolve collisions, no action will improve its speed and, therefore, its reward in the short term. We still want, though, the agent to keep a reasonable probability on actions that have proven to be profitable in the recent past. Hence, we do not want agents to minimize their chance of speeding up too quickly. On the other hand, if the environment is highly congested, the gradual decrease on probabilities will eventually force the agent to stop only when it is very close to the congestion ahead.

Consequently, our approach defines two learning rates: $\gamma_{\min}$ and $\gamma_{\max}$ which are used when the agent is decreasing and increasing its rewards, respectively. Then, if the agent took an action a_i at timestep $t-1$ which increased its reward compared to the previous timestep, at t it updates the probability of taking that action from $P_{a_i}^{t-1}$ to $P_{a_i}^{t}$ as follows:

$$P_{a_i}^{t} = (P_{a_i}^{t-1} + \gamma_{\max}), \text{ where } (P_{a_i}^{t-1} + \gamma_{\max}) < 1 \quad (1)$$

$$P_{a_j}^t = \frac{1 - P_{a_i}^t}{|I-1|}, \forall j \in I, j \neq i \tag{2}$$

If instead, the agent's reward was decreased, $P_{a_i}^t$ is defined as:

$$P_{a_i}^t = P_{a_i}^{t-1} - \gamma_{\min}, \text{ where } P_{a_i}^{t-1} - \gamma_{\min} > 0 \tag{3}$$

$$P_{a_j}^t = P_{a_j}^{t-1} + \frac{\gamma_{\min}}{I-1} \tag{4}$$

In other words, if an agents receives a higher reward by choosing an action, then in the consequent steps, the probability of chosen that action is very high. The agent still keeps a small probability for choosing the other actions, which may be necessary for continuously exploring the dynamic environment preventing the agent from getting stuck at local optima. On the other hand, if the action reduces the agent's reward, then its probability is reduced by γ_{min} in order to gradually give more weight to other potentially more rewarding actions.

5.1 Implementation Details

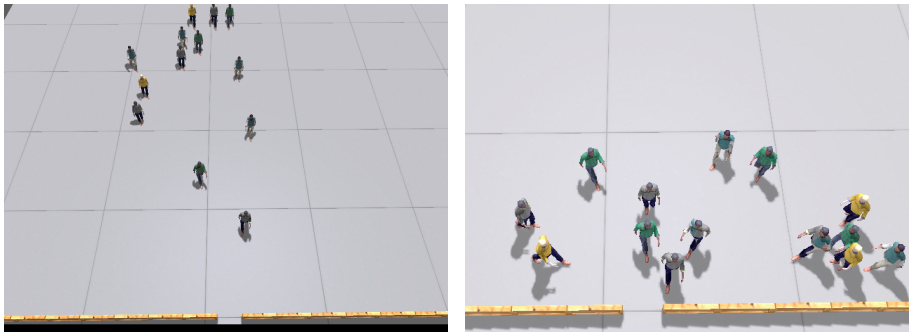
We implemented the adaptive learning approach described above using the parameter values given in Table 1. In our implementation, the agents initially have to choose between the *Speed up*, *Slow down* and *Keep Speed* actions. Only if an agent's preferred speed is reduced to almost zero, the other two actions (i.e., *Stop* and *Go Back*) become available. Note that in very congested scenarios, an agent chooses randomly between these two actions as a deterministic choice will result in all agents having the same policy (either stopping or moving back).

Consequently, in our current implementation, the simulated agents :

- learn to increase their preferred speed when it is profitable for them to do so,
- learn to keep their current preferred speed when increasing does not payoff,
- learn to take more conservative actions, whether by stopping or taking a step back, in more congested scenarios when their speed reduces significantly.

Table 1: Default parameter values for each agent.

Description	**Symbol**	**Value**
Agent's radius	r	1.1 m
Maximum speed	$v^{\max}$	2.5 m/s
Initial preferred speed	v^{pref}	1.25 or 1.8 m/s
Preferred speed increment	δ_{inc}	0.2 m/s
Preferred speed decrement	δ_{dec}	0.05 m/s
Minimum variable learning rate	$\gamma_{\min}$	0.1
Maximum variable learning rate	$\gamma_{\max}$	$0.9 - P_{a_i}^{t-1}$



(a) Hallway Scenario (b) Congested Exit Scenario

Fig. 2: **Initial Configurations** (a) In the hallway scenarios agents must walk down a hallway to an exit. (b) In the congested exit scenario agents are placed at an exit near each other and must navigate through a narrow opening.

6 Results and Analysis

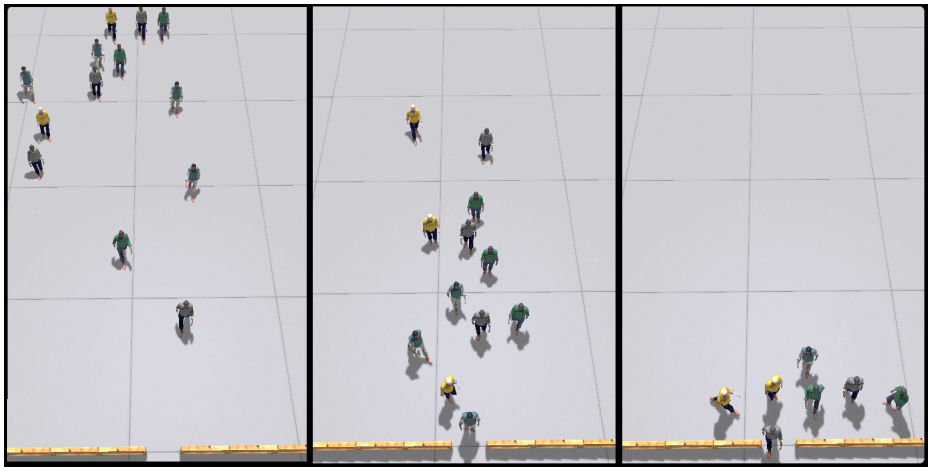
We tested our Adaptive learning approach across different scenarios, and compared the results to those obtained using ORCA (which uses constant preferred speeds). In all of our experiments, we set the initial preferred speeds of the Adaptive agents to 1.25 m/s. We also capped the agent's preferred speeds to a maximum value of 1.8 m/s in order to ensure that the virtual agents prefer to walk at speeds similar to those seen in normal human motion. In all of our scenarios, we compare the Adaptive agents to ORCA agents having a preferred speed of 1.25 m/s and ORCA agents having a preferred speed of 1.8 m/s.

Experimental Setup

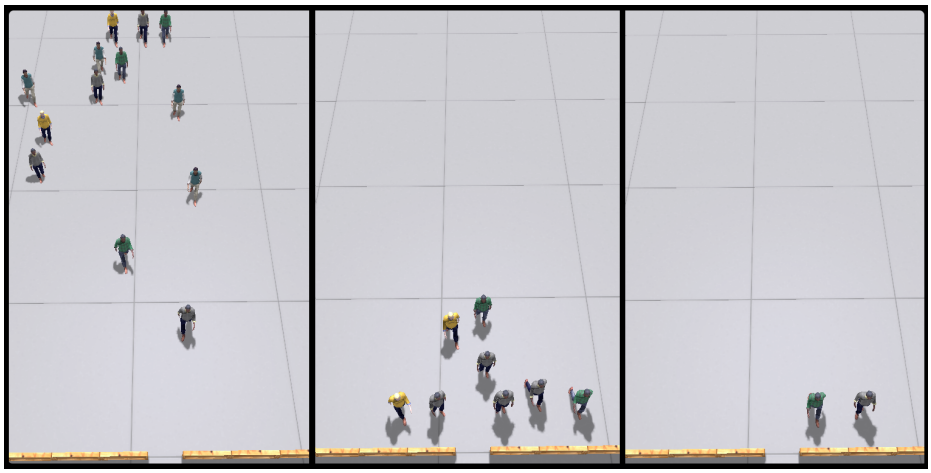
For our experiments, we use two different scenarios. The first was the *Hallway* scenario shown in Fig. 2a. In this scenario, 16 agents are placed at the end of a long hallway and must pass through a narrow exit located at the other end of the hallway to reach their goals. The second was the *Congested Exit* scenario depicted in Fig. 2b. In this scenario, 16 agents are placed very close to a narrow exit of a room and are given a goal outside of this exit. Here, unlike the Hallway scenario, agents must immediately face congestion. In each scenario, we ran 100 simulations with the Adaptive approach and another 100 with each configuration of ORCA.

6.1 Hallway Scenario

A series of time lapse images for the Hallway scenario comparing the motion of agents using our Adaptive method and those using ORCA is shown in Fig. 3. In this scenario, agents display two types of behavior as compared to ORCA's



(a) ORCA



(b) Adaptive

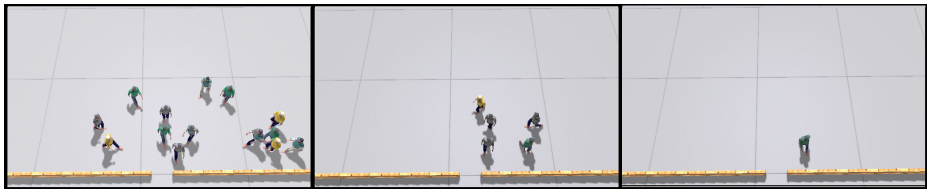
Fig. 3: **Simulation of Hallway Scenario (time lapse)** Agents are initialized walking down a hallway. (a) Using ORCA, agents do not attempt to speedup in the clear sections of the hallway and thus, they arrive later. (b) Using our adaptive method, agents respond to the situation by speeding up when the hallway is clear, and slowing down near the exit to help resolve congestion.

default behavior. First, when given a low initial velocity agents learn to speed up when there are no other agents in their way. This can been seen by comparing the first two panes of Fig. 3. At the same timestep, agents using our Adaptive approach have managed to get to the exit and leave. Secondly, the adaptive agents react as the congestion starts to form around the exit by slowing down,

stopping or taking small steps back. This increase their overall speed and rate of exit.

6.2 Congested Exit Scenario

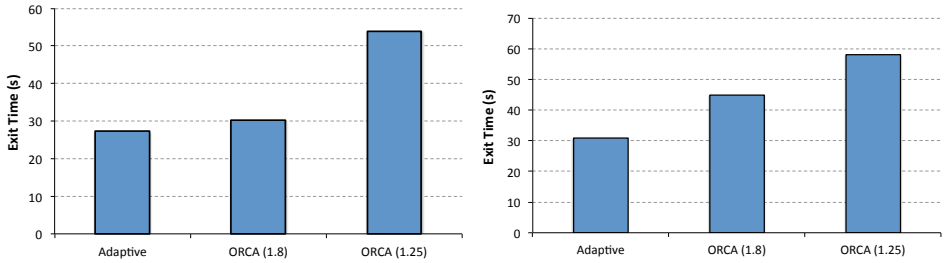
(a) ORCA



(b) Adaptive

Fig. 4: **Simulation of Congested Scenario (time lapse)** Agents are initialized in a congested formation around a narrow exit. (a) When using ORCA agents don't cooperate and have difficulty resolving congestion. (b) When using our adaptive method agents respond to the situation by slowing down and give way to each other increasing the overall flow.

Figure 4 shows a series of time lapse images in the Congested Exit scenario comparing the motion of agents using our Adaptive method to those using ORCA. In this scenario, the Adaptive agents again show behaviors not seen in ORCA. Because ORCA agents start already in congestion, they are not able to go faster by simply increasing their preferred speed (a clear example of the faster-is-slower phenomena [12]). As a result, they get caught in congestion regardless of their preferred speeds. In contrast, the Adaptive agents learn that increasing their preferred speeds does not improve their rewards. Instead, by varying their preferred speeds and being willing to stop or even step back, they are able to cooperate to reduce the congestion and increase their individual speeds and overall flow rates. This can be clearly seen in Fig. 4, as more agents exit faster and with less congestion when using our Adaptive approach than with ORCA. The ORCA agents are still struggling with congestion (Fig. 4a - last pane) while the last of the Adaptive agents is about to exit the room (Fig. 4b - last pane).



(a) Hallway Scenario (b) Congested Exit Scenario

Fig. 5: **Exit Times (smaller is better)** for (a) the Hallway Scenario and (b) the Congested Exit Scenario. The same scenario was simulated using our Adaptive approach (with a preferred speed capped to a maximum of 1.8 m/s), ORCA with a constant preferred speed of 1.25 m/s, and ORCA with a constant preferred speed of 1.8 m/s. Signicant pairwise differences between our approach and the other models are illustrated by $*(p < .05)$. Error bars represent standard error.

6.3 Quantitative Comparisons

Figure 5 shows the exit times for both ORCA and our Adaptive approach in both of our scenarios. In all cases, our Adaptive approach provided the fastest exit times. In the Hallway scenario, ORCA agents exited much quicker when they are given a higher preferred speed (1.8 vs 1.25 m/s). This is because there is not much contention early on when they are moving down the hallway, and so moving faster is the correct strategy. In the Congested Exit scenario, the numerical difference is even larger.

The results demonstrate the ability of agents using our Adaptive approach to adopt the correct parameter values across varying scenarios. We believe that these results show that such a learning approach has a good potential in making agents learn behaviors that increase their own rewards and the reward of the entire multi-agent system being simulated.

7 Conclusions and Future Work

In this work, we have combined ORCA, a popular method for multi-agent navigation, with an online learning algorithm. We have demonstrated that the resulting framework allows agents to learn to adapt in dynamic environments. We built a simple probabilistic framework where agents modify their behaviors based on feedback received from the environment, and stochastically select actions accordingly. By using principles from a known policy hill climbing approach (WoLF), agents are able to select more profitable actions with respect to their preferred walking speeds. Preliminary experiments show that, by learning to take more conservative actions in the presence of congestion, and more aggressive ones in

the absence of it, agents using our proposed approach reach their destinations faster than those agents simulated using non-adaptive navigation methods.

We believe there are many possibilities for future work. We plan to improve our learning approach by considering a more complete representation of an agent's state space. We would also like to explore different reward functions. In addition, we plan to conduct a user study to validate the behavior that the simulated agents exhibit by using our framwork. Finally, we would like to extend our adaptive navigation approach to more complex scenarios.

References

1. van den Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body collision avoidance. In: International Symposium of Robotics Research. (2009) 3–19
2. Pelechano, N., Allbeck, J., Badler, N.: Virtual crowds: Methods, simulation, and control. Synthesis Lectures on Computer Graphics and Animation **3**(1) (2008) 1–176
3. Buşoniu, L., Babuška, R., De Schutter, B.: A comprehensive survey of multi-agent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **38**(2) (March 2008) 156–172
4. Reynolds, C.W.: Flocks, herds, and schools: A distributed behavioral model. Computer Graphics **21**(4) (1987) 24–34
5. Reynolds, C.: Steering behaviors for autonomous characters. In: Game Developers Conference. (1999) 763–782
6. Funge, J., Tu, X., Terzopoulos, D.: Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In: 26th annual conference on Computer graphics and interactive techniques. (1999) 29–38
7. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Graphical Models **69**(5-6) (2007) 246–274
8. Pelechano, N., Allbeck, J., Badler, N.: Controlling individual agents in high-density crowd simulation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2007) 99–108
9. Guy, S., Kim, S., Lin, M., Manocha, D.: Simulating heterogeneous crowd behaviors using personality trait theory. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2011) 43–52
10. Popelová, M., Bída, M., Brom, C., Gemrot, J., Tomek, J.: When a couple goes together: walk along steering. Motion in Games (2011) 278–289
11. Guy, S.J., Chhugani, J., Curtis, S., Pradeep, D., Lin, M., Manocha, D.: PLEdestrians: A least-effort approach to crowd simulation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2010) 119–128
12. Helbing, D., Farkas, I., Vicsek, T.: Simulating dynamical features of escape panic. Nature **407**(6803) (2000) 487–490
13. Karamouzas, I., Heil, P., van Beek, P., Overmars, M.: A predictive collision avoidance model for pedestrian simulation. In: Motion in Games. Volume 5884 of LNCS., Springer (2009) 41–52
14. Ondřej, J., Pettré, J., Olivier, A.H., Donikian, S.: A synthetic-vision based steering approach for crowd simulation. ACM Transactions on Graphics **29**(4) (2010) 1–9
15. van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: IEEE International Conference on Robotics and Automation. (2008) 1928–1935

16. Pettré, J., Ondrej, J., Olivier, A.H., Crétual, A., Donikian, S.: Experiment-based modeling, simulation and validation of interactions between virtual walkers. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2009) 189–198
17. Karamouzas, I., Overmars, M.: Simulating and evaluating the local behavior of small pedestrian groups. IEEE Transactions on Visualization and Computer Graphics **18**(3) (2012) 394–406
18. Guy, S., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P.: Clearpath: highly parallel collision avoidance for multi-agent simulation. In: ACM SIGGRAPH/Eurographics Symposium on Computer Animation. (2009) 177–187
19. Sutton, R.S.: Learning to predict by the methods of temporal differences. In: Machine Learning, Kluwer Academic Publishers (1988) 9–44
20. Mataric, M.J.: Reinforcement learning in the multi-robot domain. Autonomous Robots **4** (1997) 73–83
21. Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. Artificial Intelligence **136** (2002) 215–250
22. Zhang, W., Dietterich, T.G.: A reinforcement learning approach to job-shop scheduling. In: the Fourteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1995) 1114–1120
23. Singh, S.P., Barto, A.G., Grupen, R., Connolly, C.: Robust reinforcement learning in motion planning. In: Advances in Neural Information Processing Systems 6, Morgan Kaufmann (1994) 655–662
24. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3-4) (1992) 279–292
25. Uther, W., Veloso, M.: Adversarial reinforcement learning. Technical report, Carnegie Mellon University (1997)
26. Kaminka, G.A., Erusalimchik, D., Kraus, S.: Adaptive multi-robot coordination: A game-theoretic perspective. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, IEEE (2010) 328–334
27. Henry, P., Vollmer, C., Ferris, B., Fox, D.: Learning to navigate through crowded environments. In: IEEE International Conference on Robotics and Automation. (2010) 981–986
28. Torrey, L.: Crowd simulation via multi-agent reinforcement learning. In: Artificial Intelligence and Interactive Digital Entertainment. (2010)
29. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
30. Martinez-Gil, F., Lozano, M., Fernández, F.: Calibrating a motion model based on reinforcement learning for pedestrian simulation