Adaptive Learning for Multi-Agent Navigation

Julio Godoy, Ioannis Karamouzas, Stephen J. Guy and Maria Gini
Department of Computer Science and Engineering
University of Minnesota
Keller Hall 4-192, 200 Union St SE, Minneapolis, Minnesota 55455
{godoy, ioannis, sjguy, gini}@cs.umn.edu

ABSTRACT

When agents in a multi-robot system move, they need to adapt their paths to account for potential collisions with other agents and with static obstacles. Existing distributed navigation methods compute motions that are optimal locally but do not account for the aggregate motions of all the agents. When there are many agents that move in a crowded space, the result is an inefficient global behavior. To address this issue, we propose a new approach which leverages techniques from machine learning and game theory. Agents using our approach dynamically adapt their motion depending on local conditions in their current environment. We validate our approach experimentally in a variety of scenarios and with different numbers of agents. When compared to other machine learning techniques, our approach produces motions that are more efficient and make better use of the space, allowing agents to reach their destinations faster.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*

Keywords

multi-agent navigation; crowd simulation; on-line learning

1. INTRODUCTION

Real-time navigation of multiple agents is important in many domains such as swarm robotics, pedestrian navigation, and traffic engineering. What makes multi-agent navigation challenging is the fact that in many cases a centralized planning system cannot be used and hence the agents have to make local decisions to produce globally efficient motions. Centralized system do not scale to large numbers of agents, are unusable when communication is limited or non existing, and are not robust to motion errors and failure.

If agents have to plan their motions in a decentralized fashion, they need to deal with the conflicting constraints that are induced by the other moving agents and the static obstacles in the environment. To avoid collisions, an agent needs to make decisions in a few milliseconds, which limits its ability to do sophisticated planning.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

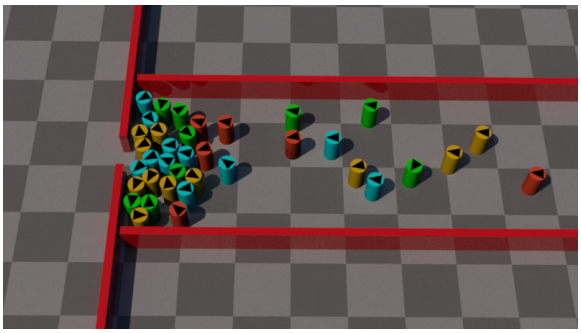

Figure 1: The *1-Exit* scenario. Multiple agents have to exit a room through a narrow doorway, leading to congestion.

Over the past twenty years, many decentralized techniques for multi-agent navigation have been proposed. These range from force-based approaches to more robust velocity-based solutions, which provide formal guarantees on the collision-freeness of the agents' motion. Although these robust approaches generate locally efficient motions for each agent, the overall behavior of the agents can be far from efficient; actions that are locally optimal for one agent are not necessarily optimal for the entire system of agents. Consider, for example, the agents in Fig. 1 that try to exit a room through a narrow doorway. When multiple agents compete for the exit, congestion arises around the doorway that causes long delays for the agents at the back and results in a time-consuming and inefficient crowd motion.

One way to improve the global efficiency of the agents' motion is to let each agent learn, through interaction with the environment, what motion is best given its local conditions. Unfortunately, many machine learning algorithms require an offline training phase to achieve learning, or take a long time before converging to an optimal policy. In addition, the computational complexity of learning methods becomes prohibitively high as the number of agents increases.

As such, our current work focuses on online learning methods that can be completely distributed and require no communication among the agents. We incorporate online adaptation into a novel framework, the ALAN framework (Adaptive Learning for Agent Navigation), which formulates the multi-agent navigation problem as a multi-armed bandit problem. The main challenge in these problems is to balance the exploitation of the current best action with the exploration of other actions that may later produce higher rewards [2]. We propose a *context-aware* action selection technique to

achieve the desired balance in real-time multi-agent navigation tasks. Our approach is inspired by the principles of two widely used techniques, ϵ-greedy [25] and Upper Confidence Bounds (UCB) [3] within the context of the '*win-stay, lose-shift*' strategy from game theory [18].

This work makes four main contributions. First, we propose the ALAN framework, where the problem of time-efficient multi-agent navigation is formulated as a multi-armed bandit problem. Second, we show how action selection techniques can be adapted in an online fashion within our framework. Third, we propose a new hybrid action selection technique, which uses tools from machine learning and game theory and which is tailored to the highly dynamic conditions agents encounter in online navigation. Fourth, we experimentally validate our approach via simulations in a variety of scenarios and show that it leads to more time-efficient motions compared to pure greedy approaches, ϵ-greedy or a variant of UCB.

2. RELATED WORK

2.1 Multi-Agent Navigation

Numerous models have been proposed to simulate individuals and groups of interacting agents. After the seminal work of Reynolds on *boids*, many interesting crowd simulation models have been introduced that account for groups [4], cognitive and behavioral rules [23], and other factors [21, 8]. An extensive literature also exists on computing a collision-free motion for an agent among static and/or dynamic obstacles, including approaches that plan in a *state-time space* [10], artificial potential fields [12] and layered architecture techniques [5].

Closely related are social force methods that simulate agent interactions using a mixture of psychophysical forces [9]. However, all these techniques are reactive and do not account for the velocities of the agents, leading to collisions especially in environments where agents move at high speeds. To address this issue, *geometrically-based* algorithms have been proposed [6], which compute collision-free velocities for the agents using either sampling [11, 20] or optimization techniques [28, 13]. In our work, we use the navigation method proposed in [28] that is robust to different types of environments by accounting for agents' velocities.

2.2 Learning in Multi-Agent Settings

A popular framework for multi-agent learning problems is Reinforcement Learning, where agents learn by interacting with the environment to achieve a desired goal, while the decision-making problem is modeled as an MDP [24]. However, solving these MDPs exactly implies learning in a large state space, which requires a significant degree of exploration to create an accurate model for each agent. To reduce complexity, approaches have been proposed that focus on the interactions of each agent with its nearby neighbors [29, 30]. To completely eliminate the state space complexity, the learning problem can be formulated as a multi-armed bandit problem, where the agents use the reward of each action to make future decisions [24]. In multi-armed bandit problems, the relation between exploiting the current best action and exploring potentially better actions is critical [1, 15].

Extensive work has also been done on learning and adapting motion behavior of agents in crowded environments. Examples of desired learned behaviors include collision avoidance, shortest path to destination, and specific group formations [27, 16, 17]. However, these are offline approaches where agents are expected to learn the MDP model through repeated simulations. In online approaches, like ours, agents have only partial knowledge of their environment and the MDP model.

3. PROBLEM FORMULATION

In our problem setting, we assume we have a set $\mathcal{A}$ of n independent agents i ($1 \leq i \leq n$), each with a unique start position specified in $\mathbb{R}^2$. Each agent has to reach a specific goal position, also specified in $\mathbb{R}^2$, as quickly as possible. The environment for each agent i is defined as a 2D virtual or physical space that includes all the remaining $n-1$ agents, along with the set of static obstacles O.

Agents should move without colliding with each other and with static objects in the environment. The motions have to be planned in a decentralized fashion and in real-time.

Notation. We model each agent i as a disc with a fixed radius r_i, an initial position $\mathbf{p}_i^0$ and a goal position $\mathbf{g}_i$. At time instant t, agent i has position $\mathbf{p}_i^t$ and moves with velocity $\mathbf{v}_i^t$ which is limited by maximum speed $v_i^{\max}$. Furthermore, agent i has a preferred velocity $\mathbf{v}_i^{\text{pref}}$ (commonly directed toward the agent's goal $\mathbf{g}_i$ with a magnitude equal to $v_i^{\max}$). We assume that an agent can sense the radii, positions and velocities of neighboring agents within a limited fixed sensing range. For simplicity, we also assume that each static obstacle O_j ($1 \leq j \leq |O|$) present in the environment is modeled as a line segment.

Let S_i be a trajectory for agent i, where a trajectory denotes a sequence of positions and velocities ($\mathbf{p}_i$, $\mathbf{v}_i$) of agent i from its start to its goal position. We say that S is a *feasible* set of trajectories if it contains exactly one trajectory per agent and the trajectories are collision-free and respect the agents' kinematic constraints. More formally:

$$\begin{aligned} S \;=\; & \{S_i, i = 1, \ldots, n \mid \\ & \forall j \in [1,n], j \neq i : \|\mathbf{p}_i^t - \mathbf{p}_j^t\| > r_i + r_j \\ & \wedge\ \forall k \in [1,|O|] : dist(\mathbf{p}_i^t, O_k) > r_i \\ & \wedge\ \|\mathbf{v}_i\| \leq v_i^{\max}\} \end{aligned} \tag{1}$$

where $dist(\mathbf{p}_i^t, O_k)$ denotes the Euclidean distance between agent i and obstacle k.

Our objective then is to find the set of feasible trajectories for the agents that minimizes the arrival time of the last agent. Let t_{opt} denote this minimum time, defined formally as:

$$t_{opt} = \min_{S \in \mathcal{S}}(maxTravelTime(S)), \tag{2}$$

where $maxTravelTime(S)$ is the travel time of the last agent that reaches its goal and $\mathcal{S}$ is the collection of all feasible sets of trajectories. Since the quality of a set of trajectories S cannot be evaluated in an online manner while the simulation is running, we assess it when all agents have reached their goals, by computing the regret R_S of S with respect to the minimum travel time, t_{opt}:

$$R_S = maxTravelTime(S) - t_{opt} \tag{3}$$

Unfortunately, the problem of computing t_{opt} in environments with multiple agents and static obstacles, even with a

centralized planner and with complete information, is $\mathcal{NP}$-hard. Therefore, R_S cannot be computed directly, except in trivial cases. However, we can derive a lower bound of t_{opt} by considering the time that it takes for the farthest agent to traverse its shortest path to its goal.

Definition 1. *We define $R_S^{'}$ to be the difference between $maxTravelTime(S)$ and the lower bound of the optimal travel time:*

$$R_S^{'} = \quad maxTravelTime(S) - \max_{i \in \mathcal{A}} \left(\frac{shortestPath(\mathbf{p}_i^0, \mathbf{g}_i)}{v_i^{\max}} \right) \tag{4}$$

We note that the lower bound of $R_S^{'}$ is zero. It also follows that $R_S^{'}$ is a strict upper bound of R_S and, as consequence, finding a set S that minimizes $R_S^{'}$ also minimizes R_S.

4. THE ALAN FRAMEWORK

In our problem formulation, as the agents navigate *independently* and without explicit communication with each other, Eq. 4 has to be minimized in a decentralized manner. In this setting, each agent has to compute a feasible trajectory $S_i \in \mathcal{S}$. However, the agents do not know in advance which trajectories are feasible as they have only partial observation of the environment (only their local neighborhood), and cannot predict the future motion of other agents. Therefore, instead of optimizing directly on the space of unknown feasible trajectories, we find, at each time instant, a velocity that respects the agent's geometric and kinematic constraints while ensuring its progress towards its goal in an efficient manner. To achieve this, we follow a two-step process. First, we plan the agents' motion over known but potentially colliding velocities $\mathbf{v}^{\text{pref}}$, and then project these velocities to collision-free ones, $\mathbf{v}^{\text{new}}$, for execution.

For the mapping of $\mathbf{v}^{\text{pref}}$ to $\mathbf{v}^{\text{new}}$ we use the Optimal Reciprocal Collision Avoidance (ORCA) navigation framework. ORCA allows each agent to select an optimal collision-free velocity by solving a low dimensional linear program [28]. It takes as input a preferred velocity $\mathbf{v}^{\text{pref}}$ for a given agent and returns a new velocity $\mathbf{v}^{\text{new}}$ that is collision-free and as close as possible to $\mathbf{v}^{\text{pref}}$. This way, we formulate the global regret minimization problem (cf. Eq. 3) as a decentralized optimization problem, and use reinforcement learning techniques to solve it.

Learning Framework. We propose a learning framework that allows agents to plan over the space of preferred velocities and dynamically adapt their motion to their local conditions (Alg. 1). Allowing the agent to select from a set of preferred velocities requires making a choice at every simulation step in a continuous 2D space of preferred velocities. However, in an online learning setting, each agent is limited in the number of samples to learn from; increasing the number of choices tends to reduce performance by either reducing the amount of samples per choice, leading to a larger learning error, or by significantly increasing the time needed for learning [26]. Therefore, in our learning framework agents plan their motions over a discretized space of a small number of preferred velocities. We also adapt a stateless representation for our framework; since agents are unlikely to encounter themselves again in a similar local situation while advancing towards their goals, a state-based learning is not ideal. Online learning problems with a discretized set of actions and stateless representation can be well formulated as a multi-armed bandit problem.

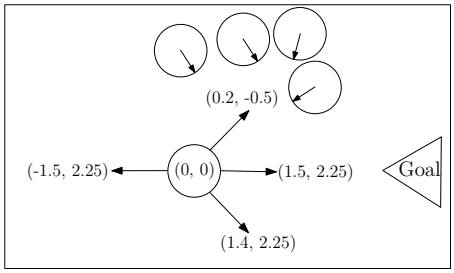

Figure 2: Example of reward values for different actions under clear and congested local conditions. The reward $\mathcal{R}_a$ of each action a is described by a goal-oriented and a politeness component ($\mathcal{R}^{goal}$, $\mathcal{R}^{polite}$).

In a multi-armed bandit problem, an agent makes sequential decisions on a set of actions to maximize its expected reward. In our domain, each action corresponds to a specific $\mathbf{v}^{\text{pref}}$. This formulation is well-suited for stationary problems, as existing algorithms guarantee a logarithmic bound on the regret. Although our problem is non-stationary in a global sense, as the joint local conditions of all agents are highly dynamic, individual agents can often undergo long periods of stationary reward distributions. Therefore, by learning the optimal motion in each of these stationary periods, we allow agents to adapt to different local conditions. Ideally, in each one of these periods, each agent should select a $\mathbf{v}^{\text{pref}}$ that minimizes Eq. 3. However, we have argued that this is not possible. Instead, we can aim at minimizing an agent's individual travel time by driving the agent towards its goal, while taking into account its interaction with other agents. We expect that considering these two factors in the local motion of each agent will implicitly improve the global motions of the agents.

Reward Function. We propose a reward function that evaluates the quality of a selected $\mathbf{v}^{\text{pref}}$ based on an agent's goal-oriented behavior and its effect on neighbor agents. Specifically, the reward $\mathcal{R}_a$ for an agent performing action a is a convex combination of a *goal-oriented* component and a *politeness* component:

$$\mathcal{R}_a = (1-\gamma) * \mathcal{R}_a^{goal} + \gamma * \mathcal{R}_a^{polite} \tag{5}$$

where the parameter γ controls the influence of each component in the total reward ($0 \leq \gamma \leq 1$).

The *goal-oriented* component $\mathcal{R}_a^{goal}$ computes the scalar product of the collision-free velocity $\mathbf{v}^{\text{new}}$ of the agent with the normalized vector which points from the position $\mathbf{p}$ of the agent to its goal $\mathbf{g}$. This component promotes preferred velocities that lead the agent as quickly as possible to its goal. More formally:

$$\mathcal{R}_a^{goal} = \mathbf{v}^{\text{new}} \cdot \frac{\mathbf{g} - \mathbf{p}}{\|\mathbf{g} - \mathbf{p}\|} \tag{6}$$

The *politeness* component $\mathcal{R}_a^{polite}$ compares the last chosen preferred velocity with the resulting collision-free velocity. If the chosen preferred velocity leads to potential collisions, a different velocity is returned by ORCA. If, instead, the preferred velocity does not conflict with other agents' motions, a similar collision-free velocity is computed. Hence,

Algorithm 1: The ALAN framework for an agent

1: initialize simulation
2: initialize $\mathcal{R}_a$ for all the actions to their maximum values
3: **while** not at the goal **do**
4: **if** $UpdateAction(t)$ **then**
5: $\mathbf{v}^{\text{pref}} \leftarrow ActionSelection$
6: **end if**
7: $\mathbf{v}^{\text{new}} \leftarrow CollisionAvoidance(\mathbf{v}^{\text{pref}})$
8: $p_t \leftarrow p_{t-1} + \mathbf{v}^{\text{new}} \cdot \Delta t$
9: $\mathcal{R}_a^{goal} \leftarrow GoalReward(a)$ (cf. Eq. 6)
10: $\mathcal{R}_a^{polite} \leftarrow PoliteReward(a)$ (cf. Eq. 7)
11: $\mathcal{R}_a \leftarrow (1-\gamma) * \mathcal{R}_a^{goal} + \gamma * \mathcal{R}_a^{polite}$
12: **end while**

the similarity between $\mathbf{v}^{\text{new}}$ and $\mathbf{v}^{\text{pref}}$ indicates how polite is the corresponding action, with respect to the motion of the other agents. Polite actions reduce the constraints on other agents' motions, allowing them to move and therefore advancing the global simulation state. Formally:

$$\mathcal{R}_a^{polite} = \mathbf{v}^{\text{new}} \cdot \mathbf{v}^{\text{pref}} \tag{7}$$

If an agent only aims at maximizing $\mathcal{R}_a^{goal}$, its behavior would be myopic and it would not consider the effects of its actions on the other agents. On the other hand, if the agent only tries to maximize $\mathcal{R}_a^{polite}$, it has no incentive to move towards its goal, which means it might never reach it. Therefore, an agent should aim at maximizing a combination of both components. Different behaviors may be obtained with different values of γ. In Section 7, we analyze the performance of our approach while varying γ. Overall, we found that $\gamma = 0.5$ provides an appropriate balance between these two extremes.

Figure 2 shows an example of different local conditions an agent may encounter. The reward values shown correspond to ($\mathcal{R}^{goal}$, $\mathcal{R}^{polite}$) tuples of an example subset of discretized $\mathbf{v}^{\text{pref}}$. Here, four of the five actions are not constrained and consequently their rewards are higher. However, congestion has formed on one side of the agent, which results in low reward values for the left angled motion. In this case, the agent will choose the straight goal-oriented action, as it maximizes $\mathcal{R}_a$ ($\gamma = 0.5$). We could determine this optimal action a^* by directly computing the reward for each $\mathbf{v}^{\text{pref}}$ and evaluating its resulting $\mathbf{v}^{\text{new}}$ (but without executing it). However, this would require an agent to simulate, for each action, the positions and velocities of its entire neighborhood and its own until the next decision-making step, which is too computationally expensive for a real-time approach. Instead, by formulating the problem of finding a^* as the problem of balancing exploration with exploitation in a multi-armed bandit problem, we can exploit action selection techniques to address it.

Learning Episode. Algorithm 1 shows an overview of our ALAN framework that adopts the traditional sensing-acting cycle in the context of a navigation task. In each cycle, each agent acquires information about the positions and velocities of other nearby agents and obstacles, and then decides how to move. In particular, an action selection method outputs a $\mathbf{v}^{\text{pref}}$ (line 5) which is given as input to the collision-avoidance framework. After determining potential collisions, this component outputs a new collision-free velocity $\mathbf{v}^{\text{new}}$ (line 7) which is used to update the position of the agent for the next simulation timestep (line 8). When a new $\mathbf{v}^{\text{pref}}$ needs to be chosen, the agent determines the quality of the previous action taken by computing its reward value (lines 9-11). This value is then given as input to the action selection mechanism, which selects a new $\mathbf{v}^{\text{pref}}$ and the cycle repeats until the agent reaches its goal.

In our implementation, we update the positions of the agents every $\Delta t = 0.025$ s. To avoid synchronization artifacts, agents are given a small random delay in how frequently they can update their $\mathbf{v}^{\text{pref}}$ (with new $\mathbf{v}^{\text{pref}}$ decisions computed every 0.1 s on average, line 4). This delay also gives ORCA a few timesteps to incorporate sudden velocity changes before the actions are evaluated.

5. ACTION SELECTION

The dilemma of exploitation versus exploration, that is, choosing the highest-valued action or trying other possibly better actions, is common in sequential decision making under uncertainty. It is particularly important in online learning, where excessive exploration might produce suboptimal motions, while too little exploration might prevent quick adaptation to local changes.

In our problem setting, agents need to explore with enough frequency to detect the changes in the reward distributions, so that they can update their reward estimates with the new values in every period. This means that the amount of information that an agent should remember, related to the previous rewards, should be carefully limited. Therefore, we use two widely used action selection techniques to estimate the optimal action a^* through exploration and exploitation in non-stationary domains: a memory-less ϵ-greedy, and a version of Upper Confidence Bounds with short-term memory (similar to [7]).

The ϵ-greedy approach [25] selects the highest valued action with probability (1-ϵ), and a random action with probability ϵ, $\epsilon \in [0, 1]$. We choose a memory-less implementation where every new sampled value of an action replaces its previous value. Consequently, by not averaging the new sampled values with older ones, one sample of the optimal action is sufficient to identify it as optimal and exploit it.

Well known results of the optimality of ϵ-greedy control carry naturally to our domain. More importantly:

Theorem 1. *ϵ-greedy is expected to find and exploit a new optimal action a^* if, on average, the reward distribution changes, at least, every $\frac{|Actions|}{\epsilon}$ timesteps, where Actions denote the set of discretized $\mathbf{v}^{\text{pref}}$.*

Proof: Assume that the optimal action $a^* \neq max(\mathcal{R}_a)$, $\forall a \in Actions$. ϵ-greedy is expected to choose a random action, on average, every ϵ^{-1} timesteps. Since an agent has to choose between $|Actions|$ actions, each one is expected to be selected every $\frac{|Actions|}{\epsilon}$ timesteps. After this time, a^* will be exploited, on average, every $(1-\epsilon)^{-1}$ timesteps. ■

Therefore, as long as the reward distribution remains stationary for a period longer than $\frac{|Actions|}{\epsilon}$ timesteps, ϵ-greedy is expected to learn the new reward distribution and optimize accordingly. However, the performance of ϵ-greedy depends heavily on the value of ϵ used and must be manually

tuned for different navigation tasks. Furthermore, the random exploration of ϵ-greedy can only take advantage of the reward distribution of the optimal action.

Another widely used action selection technique, UCB, samples the actions proportionally to the upper-bound of the estimated value of their rewards [3]. However, UCB assumes that the distributions of the rewards do not change over time, an assumption that does not apply to our domain. To address this issue and allow UCB to adapt to changing local conditions, we consider a moving time window (of T timesteps) instead of the entire history of rewards when computing the UCB estimates. We call our modified implementation wUCB:

$$w\text{UCB}(a) = \overline{\mathcal{R}_a} + \sqrt{\frac{2\ln(\nu)}{\nu_a}}, \quad (8)$$

where $\overline{\mathcal{R}_a}$ corresponds to the average reward of action a, ν_a denotes the number of times action a has been chosen and ν the total number of actions' decisions made so far by the agent, all with respect to the moving window. At each timestep, wUCB selects the action a with the maximum value of wUCB(a) across all actions. With the time window, wUCB(a) is able to adapt to changes in the distribution of the rewards and exploit the optimal action faster.

Theorem 2. *wUCB is guaranteed to identify and exploit the optimal action a^* of any reward distribution in at most T timesteps, where T is the size of the time window and $T > |Actions|$.*

Proof: Assume a worst case scenario where the agent sampled action a at time t and obtained the lowest reward value of all actions denoted as $\mathcal{R}_{min}$. Assume further that at time $t+1$ the distribution of rewards changes so that action a becomes a^*, that is, the optimal action. a^* will be sampled again when $wUCB(a^*) = max(wUCB(a))\,\forall a \in Actions$. As the value of $\overline{\mathcal{R}_a}$ (c.f. Eq. 8) does not change while action a is not selected, this will only occur when at two different times, t and $t^{'}$, the following equation is satisfied:

$$\sqrt{\frac{2\ln(\nu)}{\nu_{a^*}(t')}} - \sqrt{\frac{2\ln(\nu)}{\nu_{a^*}(t)}} \geq max(wUCB(a)) - wUCB(a^*) \quad (9)$$

As the rewards of the actions are bounded by Eq. 5, the right side of the inequality in Eq. 9 will always have a finite value. However, ν is bounded by T, the size of the time window. If $T < (t^{'} - t)$, then $wUCB(a^*)$ will become ∞ when $\nu_{a^*} = 0$, and hence it will be selected, as long as $T > |\text{Actions}|$. In this case, $wUCB(a^*) = \mathcal{R}_{a^*}$, and hence a^* will be regarded as the optimal action and exploited accordingly. ∎

We experimentally found out that using wUCB or ϵ-greedy as the action selection method in our learning framework (Alg. 1) to estimate the optimal action a^* improves the overall time-efficiency of the agents' motions (see Fig. 4 for details). Although this formulation helps agents to keep learning when the environment changes, the constant exploration mechanism (fixed ϵ or fixed T) represents the assumption that all stationary periods have at least a fixed length (T for wUCB or $\frac{|Actions|}{\epsilon}$ for ϵ-greedy). However, there is no guarantee for such assumption to hold in all navigation tasks. An agent can go through longer periods of time where no exploration is required (e.g., an unconstrained path to goal), followed by shorter periods of highly constrained motion, where exploration must be done to find an efficient path. Although choosing large values of ϵ or small values of T may help identify more changes in the reward distributions, it comes with the price of excessive trashing due to unnecessary exploration. If, instead, the value of ϵ were to be adjusted online based on local conditions, a more refined exploration-exploitation process could produce more efficient motions. Therefore, in the following section, we propose to dynamically adapt the exploration mechanism to the local conditions of the agent by using a learning strategy used in the game theoretic domain.

6. CONTEXT-AWARE LEARNING

We propose a new *context-aware* action selection technique that is suited to the highly non-stationary conditions that agents encounter in our problem domain. Our approach takes advantage of the two action selection techniques, ϵ-greedy and wUCB, introduced in Section 5, and of the '*win-stay, lose-shift*' learning strategy from game theory, enabling each agent to adapt the amount of exploration it has to perform.

Win-stay-lose-shift. The '*win-stay, lose-shift*' strategy was originally proposed as a strategy for improving the action selection performance in multi-armed bandit problems [22]. It has been shown to promote cooperation in multi-agent systems and other domains [18, 19]. It can be simply described as the strategy of maintaining one's course of action if such an action is 'winning', and modifying it otherwise. The interpretation of a 'winning' situation varies depending on the application domain, though it refers in general to the interaction between agents. We propose to use this strategy in the multi-agent navigation domain by assuming that an agent is 'winning' if it is allowed to move freely towards its goal without any constraint from its environment. Our course of action in this case is to keep the goal-oriented velocity $\mathbf{v}^{\text{goal}}$, without performing unnecessary exploration. Consequently, the agent is in a 'losing' situation if its goal-oriented motion is constrained, i.e., the agent has to slow down or deviate from its direct path to the goal. In this case, the agent should explore for potentially better actions.

$$winning(\mathbf{v}^{\text{new}}) = \begin{cases} 1 & \text{if } \mathbf{v}^{\text{new}} = \mathbf{v}^{\text{goal}} \\ 0 & \text{otherwise} \end{cases}$$

Context-aware Action Selection. Our context-aware action selection technique uses ideas from ϵ-greedy and wUCB to enhance the ability of the agents to adapt to their local conditions, while exploiting the '*win-stay, lose-shift*' rule to strategically adapt the amount of exploration that the agents perform. Algorithm 2 summarizes our technique as a part of our ALAN framework (Alg. 1, line 5).

Similar to ϵ-greedy, the parameter ϵ controls the probability of selecting the best current action or exploring a different one. However, unlike ϵ-greedy, the exploration is performed by choosing the wUCB-suggested action. In our implementation, the value of ϵ is controlled by the '*win-stay lose-shift*' strategy. The agent uses its previously chosen action to determine whether it is in a 'winning' situation or not. When the agent is 'winning', it sets ϵ to 0 (line 6) to fully exploit the goal-oriented action. However, when this

motion is constrained, the agent is encouraged to explore different actions by gradually incrementing ϵ using steps of size β (line 8). This increases the probability of selecting the wUCB-suggested actions that have high wUCB estimates. When the agent finds a new best action (other than the goal-oriented one), it will exploit it with (1-ϵ) probability, while keeping ϵ fixed. This prevents the algorithm from increasing the exploration rate when it is not necessary. Finally, the algorithm outputs a new action a, that is, a new $\mathbf{v}^{\text{pref}}$, which is passed to the collision-avoidance component of our ALAN framework (Alg. 1, line 7).

Algorithm 2: Context-aware action selection

```
1: Input: ε ∈ [0, 1]
2: Output: a
3: a ← action(t − 1)
4: if a = goal-oriented motion then
5:    if winning(v^new) then
6:       ε ← 0
7:    else
8:       ε ← ε + β
9:    end if
10: end if
11: randomly generate variable p ∈ [0, 1]
12: if p ≤ (1-ε) then
13:    a ← arg max_a(R_a)
14: else
15:    a ← arg max_a(wUCB(a))
16: end if
```

Similar to ϵ-greedy and wUCB, our context-aware approach is also bounded in the time it is guaranteed to exploit the optimal action.

Theorem 3. *The context-aware action selection is expected to find and exploit the optimal solution in at most* $T \times \epsilon^{-1}$ *timesteps.*

Proof: We can divide the analysis in the two scenarios of 'winning' or 'losing' that an agent can encounter. If the agent is 'winning' ($\epsilon = 0$) it fully exploits the goal-oriented motion. In this case, the goal-oriented motion is the optimal action a^*, hence there is no need to explore for potentially better ones. If instead, the agent is 'losing' ($\epsilon > 0$), it exploits the estimated optimal action on average every $(1-\epsilon)^{-1}$ timesteps, while exploring on average every ϵ^{-1} timesteps. From the wUCB convergence proof of Theorem 3, we derived an upper bound of T timesteps for exploiting the optimal action. However, actions suggested by wUCB are selected with probability ϵ. Therefore, our context-aware approach is expected to identify and exploit the optimal action a^* at most in $\text{T} \times \epsilon^{-1}$ timesteps. ■

Theorems 1, 2 and 3 showcase the importance that values of ϵ and T have on the ability of our approach to adapt to changing local conditions. As part of our experimental results, we analyzed the effect of different values for T in the travel time of the agents, while the value of ϵ is dynamically adjusted by our context-aware technique.

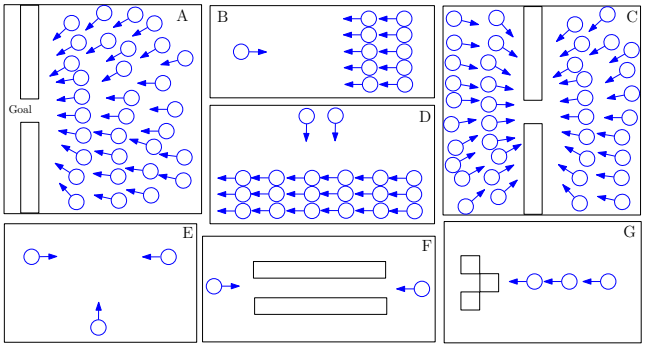

Figure 3: Simulated scenarios: (a) *Congested*, (b) *Crossing*, (c) *2-Sides*, (d)*Perpendicular Crossing*, (e) *Intersection*, (f) *Deadlock* and (g) *Blocks*.

7. EXPERIMENTAL RESULTS

We implemented our ALAN framework in C++ and tested its performance across a variety of simulation scenarios using pure greedy, ϵ-greedy, wUCB and our proposed context-aware algorithm as action selection techniques. The pure greedy approach corresponds to ORCA's goal-oriented motion. In all our experiments, we set the maximum speed $v^{\max}$ of each agent to 1.5 m/s and its radius r to 0.5 m. Agents could sense other agents within a 5 m radius, and obstacles within 1 m. The values of γ (Eq. 5) and T (for context-aware) were empirically determined (see Section 7.2), whereas a fixed value of 0.1 was used for β (Alg. 2). Whenever ϵ-greedy and wUCB were used, we also experimentally determined the best ϵ and T values respectively (typically between 0.1 and 0.2 for ϵ and 50 for T).

Action Space. After an experimental evaluation, a set of five velocity choices gave the best performance, as it provides the agent enough variety of behaviors while avoiding spending too much time in exploration. Specifically, the actions defined correspond to: (1) moving straight towards the goal, (2) left 45°, (3) right 45°, (4) backwards, all at a speed of $v^{\max}$, and (5) a complete stop.

Scenarios. Figures 1 and 3 summarize the different simulation scenarios (see `http://motion.cs.umn.edu/r/ALAN/` for videos). These include:

- *1-Exit*: 48 agents must travel through a large room, exit the room through a small doorway, and then traverse an open area to reach their goal (Fig. 1).
- *3-Exit*: Similar to *1-Exit*, but here the room has 3 exits.
- *Congested*: 32 agents are placed very close to the narrow exit of an open hallway and must escape the hallway through this exit (Fig. 3A).
- *Crossing*: A single agent interacts with a group of 10 agents moving in the opposite direction (Fig. 3B).
- *2-Sides*: Similar to *Congested*, but with two groups of 20 agents, each on the opposite side of the narrow exit (Fig. 3C).
- *Perpendicular Crossing*: Two agents have orthogonally intersecting paths with a crowd of 24 agents (Fig. 3D).
- *Intersection*: Three agents cross paths while moving towards their goals (Fig. 3E).

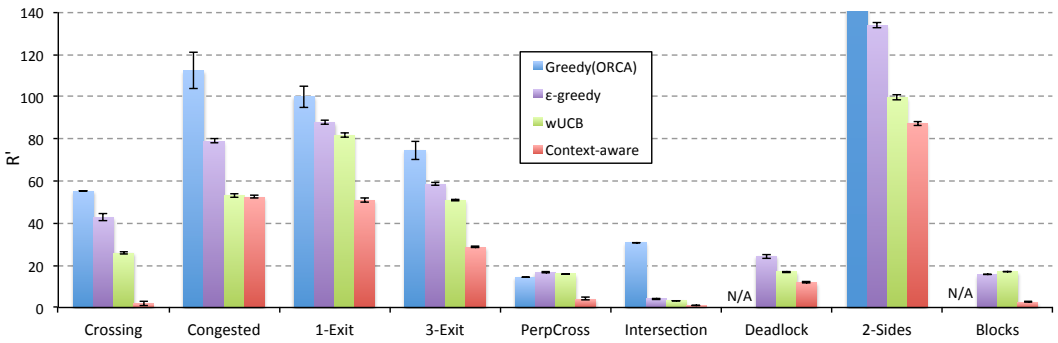

Figure 4: Performance comparison between Greedy (ORCA), ϵ-greedy, wUCB and our context-aware approach. In all scenarios, the context-aware agents have the lowest $R^{'}$. Reported numbers are the averages over 100 simulations.

- *Deadlock*: Ten agents start at opposite sides of a long, narrow corridor. Only one agent can fit in the narrow space (Fig. 3F).
- *Blocks*: Five agents need to avoid 3 static obstacles in front of them to reach their goals (Fig. 3G).

Comparison Metrics. To measure the performance of ALAN in each scenario, we evaluated $R^{'}$ (see Definition 1) of the trajectories generated by each method. To analyze the effect that exploration has on the different action selection techniques within the ALAN framework, we also evaluated each simulation in terms of the average agent acceleration. For the evaluation of the individual parameters (T and γ) we measured the arrival time of the last agent in a simulation, denoted as $maxTravelTime$.

7.1 Results

Figure 4 compares the performance of ALAN's action selection techniques and ORCA alone in all of the scenarios. As can be inferred from the figure, our learning framework improves the overall time-efficiency of the agents in all but the *Perpendicular Crossing* scenario. Here, only our proposed context-aware technique performs better than ORCA, whereas ϵ-greedy and wUCB agents exhibit excessive exploration that diminishes any advantage from learning the best action. In fact, the context-aware technique consistently minimizes $R^{'}$ across all scenarios and outperforms the other approaches.

The only case where context-aware is not significantly faster than wUCB is the *Congested* scenario. Here, the agents start in a constrained configuration, hence they cannot take advantage of the goal-oriented action until the very last part of the simulation. In this scenario, implicit coordination (achieved by using the politeness component of the reward function) is more important.

We can also observe from Figure 4 that the travel time produced by our context-aware approach is close to the theoretically optimal value ($R^{'} \approx 0$) in several scenarios (*Crossing*, *Perpendicular Crossing*, *Intersection* and *Blocks*). As a result, this means that our generated paths are (nearly) as efficient as those that could have been computed by an optimal centralized planner.

Table 1 reports the average acceleration of the agents for each method in four of our scenarios. As can be seen from the table, ORCA agents have the lowest acceleration. This is attributed to the selection of a static goal-oriented

	Scenario			
Method	*Crossing*	*1-Exit*	*PerpCross*	*Intersection*
Greedy (ORCA)	0.013	0.004	0.008	0.021
ϵ-greedy	1.19	0.159	0.35	8.23
wUCB	1.73	0.22	0.53	8.08
Context-aware	0.045	0.052	0.047	1.18

Table 1: Average acceleration per agent, in ms^{-2}.

$\mathbf{v}^{\text{pref}}$ that leads to locally efficient motions at the expense of large travel times. In contrast, ϵ-greedy and wUCB agents are characterized by significantly higher acceleration than ORCA. Due to the unnecessary exploration, these agents very often choose different actions between consecutive learning cycles which results in an oscillatory behavior. By incorporating the '*win-stay lose-shift*' rule, our context-aware technique does not suffer from such issues, with its agents exhibiting low accelerations. Taking also into consideration their fast travel times, context-aware agents are ideal for time-efficient navigation.

Goal reachability. The diversity of motions exhibited by agents using our ALAN framework allows them to reach their goals, even when pure greedy motion fails to. This can be observed in the *Blocks* and *Deadlock* scenarios. In *Blocks* (Fig. 3G), pure greedy motion causes some agents to get stuck behind the static obstacles and can never reach their goals without using a roadmap [14]. Agents using ALAN learn that sideways motions are better and reach their goals by avoiding the obstacles. In particular, context-aware agents reach the goal faster than only using wUCB or ϵ-greedy (see Figure 4). In *Deadlock*, (Fig. 3F), agents face a deadlock situation inside the corridor, which ORCA cannot solve. Instead, context-aware agents are able to backtrack and quickly reach to their goals.

Scalability. To analyze the performance of our approach as the number of agents increases, we varied the number of agents in the *3-Exit* and *1-Exit* scenarios (Figure 5). We can observe that our context-aware approach maintains a near-linear growth in travel time, while greedy agents show a super-linear growth. In fact, the difference in scalability increases as more agents are added into the system. With our approach, each added agent introduces only a fixed burden to the travel time, as it does not slow down the motion

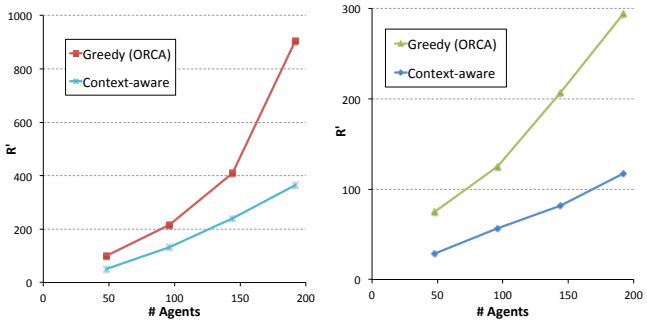

Figure 5: Scalability of Greedy (ORCA) and context-aware in the (a) 1-Exit and (b) 3-Exit scenarios

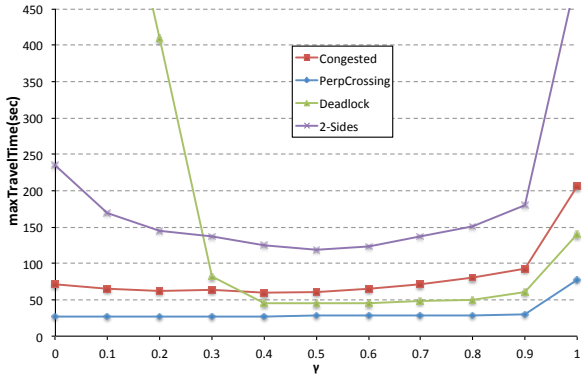

Figure 6: Performance of context-aware agents, with different values of γ.

of all other agents. With the greedy approach, extra agents introduce more constraints to the system, which significantly slows down the global motion of the crowd.

7.2 Sensitivity Analysis

Behavior parameter γ. We evaluated how the balance between the goal-oriented and politeness components of our reward function (Eq. 5), controlled by the parameter γ, affects the performance of our context-aware approach. We varied γ from 0 to 1 in four scenarios and report the results in Figure 6. We can observe that evaluating actions by either pure goal-oriented or polite motion ($\gamma = 0$ or $\gamma = 1$) can reduce performance. Pure goal-based evaluation forces agents to choose motions that are constrained by the environment. On the other hand, evaluation based only on the politeness component promotes the selection of less constrained motions, which may take agents far from their goals. Hence, as can be seen from Figure 6, equally weighting both goal-oriented and polite behaviors consistently produces more efficient motion. For example, in situations with many conflicting constraints, such as in the *Deadlock* and *2-Sides* scenarios, agents benefit from the fact that some of them choose polite motions, as it allows other agents to follow goal-oriented paths. Based on this analysis, in all of our experiments we set $\gamma = 0.5$.

Time Window. Figure 7 shows how different sizes of the time window T used in the wUCB component of our algorithm affect the performance in four scenarios. As can be seen from the figure, having no memory at all (T= 5) forces the agent to keep learning potential new values for the actions, which clearly affects the performance. On the other hand, keeping long histories of rewards has only a moderate negative effect on the performance, and reflects the extra time that it takes for agents to learn new reward distributions. Large time window sizes also increase the memory requirements for our approach. We can see that the best performance is achieved with a time window of sizes between T= 30 and T= 70. Therefore, for all our experiments, we set T= 50.

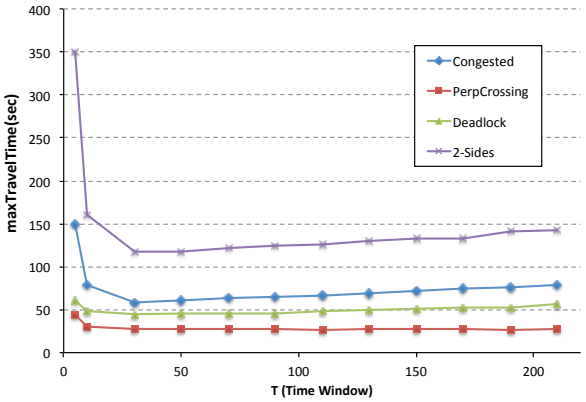

Figure 7: Travel time for context-aware agents with different sizes of time window T, for the wUCB component.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced ALAN, an adaptive learning framework for agent navigation. ALAN formulates the problem of time-efficient decentralized navigation of multiple agents as a multi-armed bandit problem, where the action space corresponds to a set of preferred velocities. In our framework, agents evaluate their actions through a reward function that encourages goal-oriented motion and polite behavior. By exploiting action selection techniques, ALAN agents dynamically adapt to changing local conditions and find new optimal actions. To balance the exploration versus exploitation tradeoff and further improve the navigation efficiency of the agents, we also proposed a context-aware action selection technique that blends in a probabilistic manner the selection of the best known action with wUCB. Our technique promotes exploration only when it is necessary by adapting the '*win-stay lose-shift*' learning strategy from game theory. We experimentally validated its performance across a variety of scenarios and demonstrated that it significantly reduces the travel time of the agents, as compared to other action selection techniques.

Looking forward, we are interested in developing methods to incorporate an even broader range of agent behaviors without involving excessive exploration. Additionally, we would like to explore our framework to simulate the motion of human-like agents, which may need additional constraints on their motion in order to look realistic.

Acknowledgment: Partial support is gratefully acknowledged from NSF grant IIS-1208413.

REFERENCES

[1] J.-Y. Audibert, R. Munos, and C. Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[2] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *The Journal of Machine Learning Research*, 3:397–422, 2003.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[4] O. Bayazit, J.-M. Lien, and N. Amato. Better group behaviors in complex environments using global roadmaps. In *8th International Conference on Artificial life*, pages 362–370, 2003.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, Mar. 1986.

[6] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using Velocity Obstacles. *International Journal of Robotics Research*, 17:760–772, 1998.

[7] A. Garivier and E. Moulines. On upper-confidence bound policies for switching bandit problems. In *Algorithmic Learning Theory*, pages 174–188. Springer, 2011.

[8] S. Guy, S. Kim, M. Lin, and D. Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–52, 2011.

[9] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[10] D. Hsu, R. Kindel, J. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.

[11] I. Karamouzas and M. Overmars. Simulating and evaluating the local behavior of small pedestrian groups. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):394–406, 2012.

[12] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[13] S. Kim, S. J. Guy, W. Liu, R. W. Lau, M. C. Lin, and D. Manocha. Predicting pedestrian trajectories using velocity-space reasoning. In *Algorithmic Foundations of Robotics X*, pages 609–623. Springer, 2013.

[14] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[15] W. G. Macready and D. H. Wolpert. Bandit problems and the exploration/exploitation tradeoff. *IEEE Transactions on Evolutionary Computation*, 2(1):2–22, 1998.

[16] F. Martinez-Gil, M. Lozano, and F. Fernández. Multi-agent reinforcement learning for simulating pedestrian navigation. In *Adaptive and Learning Agents*, pages 54–69. Springer, 2012.

[17] F. Martinez-Gil, M. Lozano, and F. Fernández. MARL-Ped: A multi-agent reinforcement learning based framework to simulate pedestrian groups. *Simulation Modelling Practice and Theory*, 47:259–275, 2014.

[18] M. Nowak and K. Sigmund. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, 364(6432):56–58, 1993.

[19] M. A. Nowak. Five rules for the evolution of cooperation. *Science*, 314(5805):1560–1563, 2006.

[20] J. Ondřej, J. Pettré, A.-H. Olivier, and S. Donikian. A synthetic-vision based steering approach for crowd simulation. In *ACM Transactions on Graphics (TOG)*, volume 29, page 123. ACM, 2010.

[21] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 99–108, 2007.

[22] H. Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1985.

[23] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69(5-6):246–274, 2007.

[24] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[25] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[26] M. Tokic. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI 2010: Advances in Artificial Intelligence*, pages 203–210. Springer, 2010.

[27] L. Torrey. Crowd simulation via multi-agent reinforcement learning. In *Artificial Intelligence and Interactive Digital Entertainment*, pages 89–94, 2010.

[28] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR*, volume 70 of *Springer Tracts in Advanced Robotics*, pages 3–19. Springer-Verlag, 2011.

[29] C. Zhang and V. Lesser. Coordinated multi-agent learning for decentralized pomdps. In *7th Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (MSDM) at AAMAS*, pages 72–78, 2012.

[30] C. Zhang and V. Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *12th International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1101–1108, 2013.