Robots and humans teaming up in pursuit-evasion games: modeling and learning issues

An *embryonic* version of a paper...

***Abstract*— If we aim to let humans and robots teaming up and effectively cooperate for a common utility, from both sides they should account for the possible behaviors of the others and act accordingly. On these premises, in this work we consider a game scenario, a pursuit-evasion one, to study how a human multi-robot team can effectively play the pursuit role, while the evaders are robots (playing rationally). Given that the human in the team can be assumed to play rationally but based on an imprecise representation model of the game, the robots need to adapt their (optimal) action policy to account for the non-optimality of the human player in the team. The pursuit robot players make use of a parametric trust model to represent human's behavior. While playing together, the model is refined and is used to make good guesses for the expected human moves, and act accordingly.**

I. Introduction

In a multi-agent team, to effectively cooperate for achieving a common utility, each agent should account for the behaviors of the other agents and act accordingly. The more the agents need a tight interaction and coordination while performing the common task, the more such a knowledge about other agents in the team is expected to be required to perform optimally. When the team is a mixed one, featuring the presence of both humans and robots, the situation is quite a challenging one, since it can be problematic for a human to have an effective and compact model of robots' behaviors, and, similarly, for a robot to have a reliable model of the behavior of its human teammates. The presence of multiple robots (and multiple humans) make the scenario even more complex, since also mutual interactions and specific individual differences should be accounted for.

Given these premises, the purpose of this work is to consider a simplified, yet challenging, game scenario to study how rational robotic agents can account for the presence of human teammates and adapt their action policy accordingly. Human agents are seen as rational agents, but relying on imprecise or partial knowledge representations, parametrized by the peculiar characteristics and history of the specific individual. Multiple robots can be part of the team, potentially leading to a human-swarm mixed team.

We address such a scenario by building on the notion of *trust*: we build empirical models that represent how "far from optimal" the human is expected to be when playing the game, and let the robots use this models to adapt to this information. In other words, we build heuristic models that say to which extent the rational robotic agent can "trust" the human, and we define a sound methodology to adapt accordingly the optimal robot policy to maximize the expected common team utility. The game scenario is modeled as an MDP.

The parametric trust models accounts for different ability "dimensions" of the human agent: ability to deal with large action spaces, ability to perform look-ahead, ability to represent the current state of the game. In addition, the trust model also includes a self-evaluation parameter directly input from the human regarding his/her ability to play the game optimally. Through this parameter a general trust model can be made more specific to the human player.

The humans' models are made available to the robots and are used to derive estimates about the next actions that would issued by the human teammates. Moreover, observing the human playing, new data about his/her abilities can be gathered by the robots and used to online revise the parameters of the trust models (as well as at the end of the game. This amounts to *learning both effective models for general human behavior in the game and for specific individual behavior*.

The *pursuit-evasion* (PE) problem is considered as game scenario. It consists of a team of players (pursuers) trying to capture all the members of another team (evaders) while minimizing the capture time. In our settings, humans play as pursuers together with other pursuer-robots. Evaders are all robots. The game has been set up both in a virtual environment (on the web) and in a real-world environment (in a very preliminary setting, see `https://youtu.be/Fjuya6wfDQ4`).

The considered game scenario is quite complex since it requires cooperation to achieve a common goal in a dynamic environment. For sake of simplicity, here we only consider settings such that the number of pursuers is sufficient to capture all the evaders. In general, pursuers robots are not able to capture all the evaders on their own without the help of the human pursuer. Therefore pursuer-robots need to adapt their strategy to take advantage of the "non-optimal" human collaboration to perform well and complete the game.

In general, human-robot cooperation systems try to recognize human intentions or requests through observation or explicit interaction (e.g., communication). Making optimal decisions under uncertain transitions is then achieved by learning the transition model, that is, the model of how the human acts. However, learning an effective transition model can require a large amount of observations and therefore a large number of games needs to be played before robots can effectively make use of human's help. Instead, we aim to be able to play "well" since the beginning, not necessarily waiting for learning the transition model. This is obtained by building a team policy based on a trust model of the "average" human player. The game is therefore directly

playable, but also adapts online to players with different abilities.

The originality of our approach resides in the fact that it combines a heuristic model for human decision-making in game scenarios with an online learning approach to make rational agents (i.e., robots) to adapt their policy to explicitly account for human non-optimality. Moreover, the proposed approach can quickly learn the high-level parameters influencing human decisions, without the need for building a model from scratch out of many games. A trust model is proposed which is based on the ability of the human to take an optimal decision taking into account the difficulty of the decision. From this model, the human transition (i.e., action) model is derived, allowing to compute the optimal pursuer-robots strategy by solving the corresponding Markov decision process (MDP). A second advantage of our approach compared to existing approaches is that significant parameters are used. Indeed, rather than learning the transition model without being able to extract any meaningful information from it, our approach learns parameters related to specific human abilities.

In the next section we give a description of the game and states the main assumptions of the scenario. Then we review related works before The describing the model for the human decision-making and the the way this is used to adapt the parameters of the robots' MDP, and how learning is implemented. Numerical results and evaluation of the proposed approach are presented in the final section.

II. Assumptions and game description

The game is played in a discrete environment, which is represented through an *undirected graph G(V,E)* (no particular restrictions are imposed on the structure of G). G's vertices represent feasible discrete locations for the players, while the edges represent physical feasible moves between adjacent vertices. A small subset of the vertices is labeled as "Exit".

There are two teams and three types of players: the *evader* team, composed of evader-robots, and the *pursuer* team, composed of pursuer-robots and one (or more) human player. An initial configuration is provided giving the location of each player in the environment.

The game evolves according to *discrete time steps*. Players take alternating turns. On the pursuers turns, each pursuer either remains at its present location or moves to an adjacent vertex (all pursuers move simultaneously, with the common move being triggered by the human). Similarly, on the evaders turns, the evaders either remain at their present locations or move to an adjacent vertex. To increase the difficulty of the game, the evader could also be allowed to move to a vertex adjacent to an adjacent vertex (as it would have played twice on the same turn).

Capture is defined as states where an evader cannot move besides going into a cell where a pursuer or an evader stands. After capture the evader is removed from the game.

The game assumes *full knowledge*: at each time, all players know the locations of all participants to the game. The number of pursuers and evaders is fixed and known to the players. The approach for decision-making is *centralized*: a central process computes and communicates the actions that should be executed by each pursuer robot.

The pursuer team wins when no players of the evader team can make a move (or all have been captured). The evader team wins when one of its players reaches one of the vertices labeled with "Exit".

Evaders act selfishly, they do not cooperate. The action policy of an evader is fixed. It is first defined by computing the optimal policy of the pursuers (maximizing the reward of the pursuers), and then fixing the policy of an evader to the one that minimizes the reward of the pursuers.

The pursuer team player is said to have a *winning strategy* if no matter what choices the evaders make, it can eventually win the game. Otherwise, if the evader can indefinitely avoid capture no matter what choices the pursuer makes, then the evader player has a winning strategy.

In the current implementation, pursuers earns a reward of R=1000 when they capture the evader, and gets R=-2000 when the evader exits. Following each move, a negative reward R=-100 is also received, to promote an early catching of the evaders.

The pursuer performance of a game is measured with the gathered rewards, meaning in the number of time steps required to capture all the evaders. The better the cooperation between the pursuer-robots and the human, the less time will be needed to successfully finish the game.

A. Formal definition of the game

The game has 3 sets of players. Let H be the human pursuer players, P the set of pursuer robots, E the set of evader robots. A state s of the game, $s \in S$ is defined by the state of each of the players, s_h, s_e, s_p, that corresponds to the location of the player in the game environment. In turn, each $s_i, i = \{h, e, p\}$ is a joint configuration of the players in the specific set (e.g., $s_e = s_{e1}, s_{e2}, ..., s_{e|E|}$). At time t, the state of an individual player is determined by the vertex the member occupies at t. The feasible actions available to a player are the actions $A(s) = \mathcal{N}(v(s))$ where $\mathcal{N}$ is a neighborhood function and $v(s)$ the vertex occupied by the player in state s.

When the pursuer-robots execute an action, the human and the evader also execute an action at the same time. Therefore, from the point of view of a player, the effect of its action a at step t is only partially known in terms of what is the game state at $t+1$. In other words, an action of a player P in a given game state $s \in S$ results in the system passing to a set of possible outcome states $S' \subset S$. Each state-action pair outcomes a state s' that is associated to a *transition probability* of passing to that system state, which is based on the action policy employed by each player in the game. In order to make good plans, the transition probabilities should be as close as possible to real transition probabilities:

$$Pr(s'|s, a)\ \forall s, s' \in S,\ \forall a \in A(s).$$

Updating these probabilities is the key step to adapt to the human behavior because the pursuer-robots can act using a model predicting how the human would act.

III. Related work (*it's not clear how correct/good is the content below, but it can help to get some ideas*)

The literature on Pursuit-evasion on graphs can be roughly decomposed in approaches based on a deterministic analysis and approaches using a probabilistic analysis of the game. Deterministic analysis can be applied when pursuers and evaders move deterministically and everything about the game is known, such that the system can minimize capture time. Otherwise, the graph needs to be fully explored to guarantee that no evader remains in the environment (for example when the number of evaders is unknown or when the evaders move arbitrarily fast), hence the capture time is not minimized. On the other hand, probabilistic analysis aims at reducing the average capture time [1]. This type of methods can be applied when the evaders locations are unknown or when the evaders or the pursuers move stochastically (or the effect is stochastic from the point of view of a pursuer). Therefore, in our context, only probabilistic methods can apply and the main proposed approaches are detailed in the following.

A. Deterministic analysis

Deterministic planning is not feasible in the previously defined scenario because an action has multiple outcome states and our objective is to minimize the capture time. Yet, the approach is relevant because it handles the scalability issue also present in the probabilistic analysis.

In combinatorial analysis, we can distinguish monotonic search (that always decreases the size of the dirty set) to search that allows recontamination of a cell. Monotonic search is easier to implement but leads to solutions that require a larger number of pursuers.

In [2] evaders are adversarial and have, like the pursuers, full knowledge of the current state. The capture state is reached when all evaders reside on vertices occupied by at least one pursuer. The proposed method explores all the states and all the transition starting from the capture states, hence the algorithm is $O(v^{p+e})$ (with v, p, e denoting the number of **v**ertices, **p**ursuers and **e**vaders). The optimal policy is precomputed, then during execution all robots have access to the current state and its 'solution' (i.e., what it should be done to follow the policy). To solve the scalability problem, in [3] the problem is partitioned by assigning a group of pursuers to every evader and solve each multi-pursuer single-evader problem optimally with the previous described approach.

In [4] an algorithm is proposed for clearing a building (from intruders) using a human-robot team. The GSST algorithm that is proposed evaluates different minimum spanning tree to find a solution. Solutions gets better over time (solution with less pursuers). Given a sufficient runtime, the optimal number of pursuers is found along with a strategy. Humans follow waypoints on a laptop. It can handle only local visibility and also infinite evader speed.

Note that on a planar graph, 3 pursuers is sufficient to capture 1 evader even if its motion model is unknown. The strategy consists in approaching the evader and blocking its way out[].

B. Probabilistic analysis: Cooperation with agents with unknown policy

Probabilistic analysis is suitable for pursuit-evasion when evaders' locations are unknown and/or when evaders movements are unknown or probabilistic.

A probabilistic analysis of pursuit-evasion game is usually based on information space and Bayesian reasoning. Information space is the sequence of observation of the locations of pursuers and evaders up to the current time. According to the observation, pursuers estimate the position of evaders using Bayesian rules. In [], based on the estimation, two greedy algorithms are designed to provide pursuers policies to catch evaders. The local-max algorithm guides pursuers to their neighbor states which have the highest probability to catch an evader. The global-max algorithm guides pursuers one-step closer to states which have the highest probability over the whole state space to catch evaders.

Methods using Markov Decision Process (MDP) have been proposed to solve the pursuit-evasion problem. These approaches are described below.

1) Cooperation with AI agents: In [5], the authors have proposed the use of non-deterministic policies to find near-optimal policies in order to provide flexibility to the person executing the policy. Their approach adds "not-so-worse" actions besides the optimal action to the ones feasible by the agent. In other words, the set of actions that the person can execute in a given state to achieve is restricted to the ϵ-optimal solution. This is not applicable in our context, as the human player should be able to move autonomously.

Reinforcement learning (RL) can be used to learn the transition model of the other agents. In a multi-agent setting RL learns an internal representation of the other agents to estimate their policy and adapts the behavior [6]. In other words, the system learns about other agents in the environment so as to make good guesses of their expected behavior, and to act accordingly (to cooperate with them more effectively).

Making optimal decisions under uncertain transitions is usually achieved by learning the dynamics of the system, that is, the transition model (probability distribution over successor states for all state-action). Observing how the human acts would allow learning a model of how he/she behaves, that is, the transition model. For example, reinforcement learning (RL) can be used to learn the transition model of the other agents. However, learning this transition model would usually take a large amount of observations and therefore a large number of games needs to be played before the system interacts efficiently with the human. As the game should be directly playable, learning the transition model is not feasible.

2) Cooperation with humans: More closely related to approach that we propose in this work are the ones that model the human intention from a higher level perspective. Indeed, to enable adaptive decision making in games with a human player and AI players, the method proposed in [7] builds an MDP-partially observable Markov decision process (POMDP) couple. In this case, the MDP models the game world and the POMDP only models the players. The two policies are combined to adapt to player changes. Most of the computations are done offline.

In [8] and [9], respectively and MDP and a POMDP formulations are used to address the problem of a human player and a sidekick AI trying to catch multiple evaders. The human player needs to capture evaders but requires a sidekick to block an escape. Both approaches take into account the uncertainty on the subtask selected by the human (i.e., the evader the human is pursuing). In [8] the multi-evader MDP is decomposed into multiple single evader MDPs. Each of these MDPs is solved optimally using value iteration. The likelihood of a subtask to be selected by the human is based on the rewards of each MDP and on a probabilistic model estimating the probability of switching between subtasks. The action executed is the one that maximizes the expected reward given a belief on the selected subtask. Similarly, in [9] the human intention is used: the evader the human has chosen to pursue is a state variable, and solves the corresponding POMDP using a Monte Carlo tree search.

Note that our approach builds upon a similar methodology but we use a trust value to represent human ability playing the game. Instead of reasoning on a probabilistic model of the target pursued by the human, the proposed approach assumes the human is trying to make the optimal decision, and reasons on a model of this attempt. The model takes into account the human's ability and knowledge about the current state of the game. The originality of our approach resides in the fact that it quickly learns the high-level parameters influencing the human decisions. A trust model is proposed which is based on the ability of the human to take optimal decision while taking into account the difficulty of the decision.

C. Trust

In [10], trust is defined as "belief, held by the trustor, that the trustee will act in a manner that mitigates the trustor's risk in a situation in which the trustee has put its outcomes at risk". In the proposed method an outcome matrix is used to model the interaction between two agents. This matrix contains the reward for each agent depending on each agent action, which allows to identify trust situations. In our context, the reward/risk are the same for the trustor/pursuers and the trustee/human. It is a trust situation because if the pursuer suspect that the human is not cooperating, it should modify its policy to maximize its reward. Trust is then used to compute, in a given state, the probability that the human will make the best action among its possible actions.

In [11], the authors claim that the most commonly used trust model in *Ad Hoc* wireless networks to detect corrupted nodes is a Bayesian updating. The trust update value is computed as:

$$t = \frac{s+1}{s+f+2}$$

where t denotes the trust, s, the number of successful interaction and f the number of failures.

In [12], the authors differentiate trust and reputation but as we only use direct interaction to build the trust, there is no significant difference.

In [13] and [14], an entropy calculation is used to take into account the uncertainty (the more the confidence the more the trust gets updated):

$$T_{update} = \begin{cases} 1 - H(t) & \text{if } 0.5 \leq p \leq 1; \\ H(t) - 1 & \text{if } 0 \leq p \leq 0.5; \end{cases}$$

where: $H(t) = -p\log_2(t) - (1-t)\log_2(1-t)$.

IV. Overview of the method

The proposed approach aims at solving the MDP problem optimally while taking into account the uncertainty derived from human non-optimality, therefore requiring a trust model of how the human makes his/her decisions. The proposed model is based on multiple criteria trying to model the human's ability towards different aspects of the game's decision difficulty.

At the start of the game, the human inputs its self-assessment ability value in the trust model. This input sets the initial parameters of the human model. The model is then updated with the observations of the human actions and compared to the optimal policy (that would be played by a fully rational agent with complete information about the game).

In PE games, assuming the human player is honest (not trying to make the pursuer loose the game), its decisions are related to: (i) the ability to discriminate among the available choices, (ii) the ability to look-ahead, (iii) the ability to grasp a useful knowledge about the progress of the game. In the trust model, we represents these aspects in a parametric way, each value corresponding to a different aspect of the human ability or knowledge. *These values are then used to modify the probability distribution of human action selection in the transition model of the MDP*. The idea is that more "far from optimal" a human is expected to be according to the trust model, more "random" is expected to be the human action selection.

More specifically, the following three trust factors are considered to modify the probability distribution associated to human action selection:

- *difficulty of the decision*: we assume that the more difficult the decision is (i.e., more choices), the more random the human decision will be;
- *human ability to foresee the impact of its decisions*: smaller is the ability the less the human player is able to make good decisions, such that the more his/her choices will appear as random choices;
- *human knowledge about the game rules, how to play (experience) and knowledge about the current state of the game*.

The initial distribution of the probability of each human action is computed from the optimal state values computed with an MDP which is normalized to obtain a probability distribution.

V. Pursuit-evasion with human-robots pursuers as an MDP

A. MDP for a controllable pursuer

The goal here is to build an offline policy that can be used to evaluate the individual actions taken by the human player. Solving the MDP builds a policy for a deterministically controllable pursuer player (as in [4]), but it also computes Q-values for every state-action pair. To compute the trust we need to be able to evaluate a human action. In order to this, it is useful to compute an MDP as *if the human was controlled by the optimal policy*.

A Markov decision process (MDP) is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$, where:

- $\mathcal{S}$ is the set of possible states of the game. Here the state is defined by the position of the members of each player.
- $\mathcal{A}$ is the set of possible actions. Considering that G represents a *grid environment*, each member of the pursuer player set has 5 actions (N,S,E,W and no move), so the number of possible combinations can be large (5^n) for n pursuers. For simplicity we assume that $\mathcal{A}$ is not state dependent. Each individual action for each individual player is a deterministic one (i.e., selecting N result in moving in the N grid cell).
- $\mathcal{T}(s, a, s')$ is the transition function, which is equal to $P(s'|s, a)$, the probability that action a in state s will lead to state s'. In our case, evaders move semi-deterministically (moving away from a pursuer member if their inter-distance is below a threshold, breaking ties randomly). Pursuer-robots move deterministically.
- $R : \mathcal{S} \mapsto \Re$ is the reward function. The capture of pursuers are the only states that generate positive rewards. Negative rewards are generated for every non capture states.

A policy π determines the action to take when in state s_t. The objective is to find a policy that maximizes an utility function U, expressed here as the expected sum of rewards accumulated over time:

$$U(\pi) = \mathbb{E}\left(\sum_{t=0}^{T_{max}} \gamma^t R(s_t)|\pi\right) \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor.

The best value of the utility that can be obtained from a state s is represented by a function called the value function, $V(s)$. The value iteration algorithm [Bellman 1957] iteratively updates the value function the equation, until $V^*(s)$, the optimal value function is computed:

$$V_{t+1} = \max_a \left(\sum T_a(s, s')(R_a(s, s') + \gamma V_t(s')\right) \tag{2}$$

$V^*(s)$ is the expected cumulative reward of running the optimal policy π^* from state s.

Applying π^* consists in executing the optimal action a^* in each state:

$$a^* = \underset{a}{argmax}\left(\sum_{s'} T_a(s, s')V^*(s')\right) \tag{3}$$

B. MDP for human-robot pursuer teams

The main difficulty of having a human in the loop is that actions of the pursuer player are probabilistic. In the previously described MDP, the transition model $T_a(s, s')$ in equation 2 is unknown. We want to build and learn a human model to infer the transition model.

Reinforcement learning techniques can be naturally employed to learn the transition model. However, the approach is potentially too slow, since it requires a large amount of observations. This is not really feasible during a game or a small number of games. Therefore, our approach is to start from a value of trust that immediately allows to "play", and that can be improved later on. To tackle this issue, we a trust model to compute the transition model that takes into account the uncertainty on the decisions of human player. This model outputs trust value(s). It is set initially with an input from the human player (for example the human tells the system how good it think he can perform). Trust is updated during the game using the observations of human actions.

The general trust τ reflects the ability of the human to take optimal decisions. Its value is in [0,1] interval:

- A trust value of 1 indicates the human will always take optimal actions.
- A trust value of 0 indicates the human will take a random choice in between actions.
- A trust value of 0.5 indicates the human is the average human.

In the MDP described in section V-A, we can add an intermediate state between the pursuers action and the human action. Indeed, even though they play simultaneously, the pursuers need to determine their actions without knowing what the human will do. An action a in the MDP can be decomposed in a_p, the pursuers actions, and a_h, the human action ($a = a_p \cup a_h$). The optimal policy (π^*), and therefore optimal Q values (Q^*) can be computed off-line for the pursuer player because the action are deterministic. Human actions are not known a priori, therefore when the human is included as a pursuer, action of the pursuer-player become probabilistic. The probabilities of human actions are required to compute the MDP i.e., a human transition model is required.

1) Human model: The transition matrix is a $|S \times A| \times |S|$ size matrix, which is very sparse. This makes efficient to store for each state-action the possible outcome states and their respective transition probabilities.

The action of the pursuer-robots is found by computing an MDP with the new transition model. It is optimal in regards to the human action selection model.

Learning the transition model requires a prohibitively large number of game playing. Assuming the human's intention are good, he plays cooperatively and tries to execute the

best actions. The probability of each action then depends on the difficulty of the decision and on the human perception of Q-values.

This difficulty depends on the difficulty of the decision coming from the game state and on the human ability and current knowledge of the state of the game, including the number of feasible actions and their respective expected rewards. The expected reward computation should be discounted because long term rewards are more difficult to "see".

The transition matrix is then updated depending on this trust model.

C. Operational scheme and complexity issues

Figure 1 illustrates the proposed structure of the game and of the MPD.

The main problem with this structure is the computational complexity required for solving the problem to optimality. Unfortunately, MDPs scale exponentially with the number of state variable. Here, if n_p is the number of pursuers, n_e the number of evaders and N_{cells} is the number of cells of the grid world, then the MDP state space is of size $\mathcal{O}(N_{cells}^{n_e+n_p+n_h})$. Therefore, the size of the MDP space needs to be reduced in order to be practical.

D. Factors influencing human decision

As we have already pointed out, in PE games, human decisions are related to his/her "ability" (look-ahead, ability to discriminate the best choice) and to current knowledge of the progress of the game. To model these parameters we use a trust value τ which is decomposed into different components, each one corresponding to a parameter of the human ability or knowledge. These components are then used to progressively modify the probability distribution of human action selection.

The initial distribution of the probability of each action is computed from the Q-values which are normalized to make them a probability distribution. More specifically, the following three general trust factors modify the probability distribution associated to human action selection:

- *Number of feasible actions* compared to the ability of the human to manage multiple actions;
- *Distance-to-go* (for action, minimum time that lead to a capture) compared to human look-ahead (how far in time does the system thinks the human is thinking);
- *Positions of the other agents* (measure of knowledge about the location and intentions of the other players);

These trust factors are then translated into a transition matrix allowing to decide pursuers moves according to the human decisions.

E. From trust to transition matrix

In this section, we describe the process by which the full transition matrix used to compute the MDP for the pursuer-robots is derived from the initial trust value and how it is updated.

The transition model is updated using the observation of the human action compared to the optimal action. A new policy is then computed for the pursuer-robots optimal giving the next action to execute.

Factors that influence the human decision are detailed in the following subsections. For each factor, the initial parameters and their update is shown. Updates of these parameters are done by comparing the human actions observed to the current belief in the human's ability with initial parameters given by the human at the beginning of the game

Figure 2 illustrates the states transition of an example where a the robot-pursuers have the choice between two actions. This numerical example points out the impact of the transition model used. Notably its policy will be different depending on the transition model, which is computed from the trust.

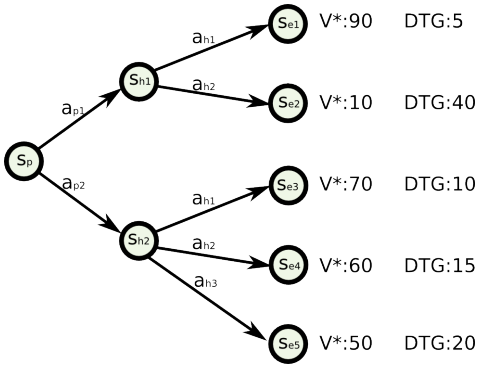

Figure 2*: Illustration of state transition. Sp is a state where the pursuer has to play and Sh a state where the human plays. V* values are the values of the states using the optimal policy.*
If a random transition model is used for the human, the best action will be action a_{p2}:
$V_{sp1} = 0.5 * 90 + 0.5 * 10 = 50 \quad < \quad V_{sp2} = 0.33 * 70 + 0.33 * 60 + 0.33 * 50 = 59.4$
But if the best action has 0.8 *probability of being chosen then the best action is action a_{p1}:*
$V_{sp1} = 0.8 * 90 + 0.2 * 10 = 74 \quad > \quad V_{sp2} = 0.8 * 70 + 0.1 * 60 + 0.1 * 50 = 67$

1) Initial trust: The initial trust is given by the human at the beginning of the game along with a confidence in this trust. Trust is a value between 0 and 1 and confidence between 10 and the maximum number of turns. Trust serves as parameters for the initial distribution of transformation parameters (ability in the number of actions, look-ahead). The second servers as hyperprior or pseudo-counts of the distribution probability of these transformation parameters.

2) Initial distribution: The initial distribution (before modifications) is influenced by the difficulty of the decision is given by the optimal Q-values from the MDP with a controllable human. The Q-values are normalized to give a probability distribution. This probability distribution function (PDF) is modified in regards of the difficulty of the decision, the current experience, and the ability of the human player.

3) Number of actions: The more actions are available to the human, the more likely he/she will make a random choice. Therefore, the more the actions the more the PDF

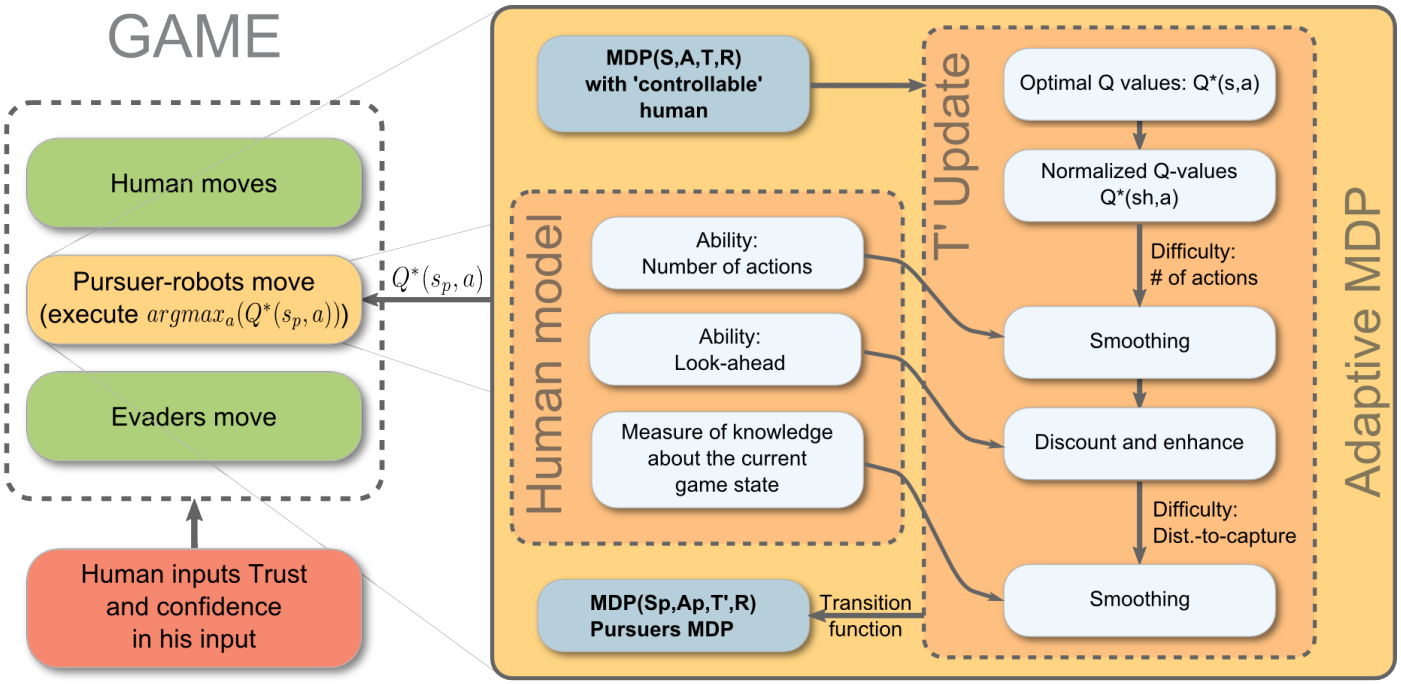

Figure 1: *Operation scheme*

should be smoothed. τ_{NA} denotes the ability of the human regarding the number of action.

The probability of success for a given number of action is given by the following equation (chosen in a ad hoc manner:

$$\tau_{NbA} = \sqrt[4]{\tau_{NA} * (1 - \frac{Nba}{MaxNba + 1})} \quad (4)$$

The initial τ_{NA} is set to the global trust value $\tau_{NA} = \tau$.

The smoothing factor (SF) is inversely proportional to the trust. Also, the smoothing parameter should be proportional to the number of actions available NbA. The smoothing factor SF is then computed with the following equation:

$$SF = (\frac{1}{\tau_{NA}} - 1) \times NbA \times P_{NA} \quad (5)$$

where P_{NA} is the parameter of the average player which is updated after each game (see *Updating* paragraph).

For example, with extreme values of trust:

- when $\tau_{NA} = 1$ there is no smoothing because the human is trusted to take good decision among a large number of choices;
- when $\tau_{NA} = 0$ the smoothing factor is large and the distribution becomes uniform, the human is assumed to take a random action.

The distribution is then smoothed. For each action the probability of this action is:

$$Pr_a(a_i) = \frac{Pr_b(a_i) * (Confidence * \tau_{NA} + NbSuccess) + SF}{(Confidence * \tau_{NA} + NbSuccess) + SF * Nba}$$

Figure 3 illustrates the relation between the smoothing factor and the number of actions for a 0.5 τ_{NA}.

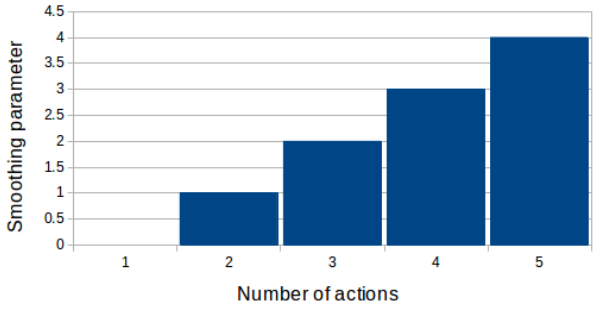

Figure 3: *Smoothing factor and number of actions with* $\tau_{NA} = 0.5$

4) Updating: An observation is counted as success or failure depending on whether the human executed the optimal action or not. This count is kept for every category i.e. for every number of actions.

An observation of a success adds 1 to successes in its category and categories with less actions.

An observation of a failure adds 1 to failures in its category and categories with more actions.

This count serves for the computation of the expected value for the probability of success for a given number of action:

$$E(Z - opt|Nba) = \frac{s(Nba) + \tau_{NA} * Confidence + 1}{n(Nba) + Confidence + 2}$$

where $Z - opt$ is an observation of the human executing the optimal action, $s(Nba)$ the number of successes i.e. observations where the human chose the optimal action among Nba actions and $n(Nba)$ the number of observations in the Nba category.

5) Look-ahead and distance-to-go: The distance-to-go (DTG) of an action corresponds to the minimum number of turns before a capture can happen after taking an action. The look-ahead (LA) corresponds to the number of turns the system thinks the human is thinking ahead.

τ_{LA} denotes the ability of the human regarding the distance to the goal (the number of action before success can be achieved). As the distance to the goal is directly proportional to the value in the Optimal-Human MDP, we simply use the MDP value divided into a finite set of buckets.

Then, similar to the τ_{NbA} we compute the probability of success for a given distance with the following equation:

$$\tau_{LA} = \sqrt[4]{\tau_{LA} * (1 - \frac{DTG}{MaxDTG + 1})} \tag{6}$$

The initial τ_{NA} is set to the global trust value $\tau_{NA} = \tau$.

An observation is counted as success or failure depending on whether the human executed the optimal action or not. This count is kept for every bucket.
An observation of a success adds 1 to successes in its category and categories with larger values (closer DTG).
An observation of a failure adds 1 to failures in its category and categories with lower values (longer DTG).

This count serves for the computation of the expected value for the probability of success:

$$E(X|DTG) = \frac{s(DTG) + \tau_{LA} * Confidence + 1}{n(DTG) + Confidence + 2}$$

where X is an observation of the human executing the optimal action, $s(Nba)$ the number of successes, i.e. observations where the human chose the optimal action among Nba actions, and $n(Nba)$ the number of observations in the Nba category.

6) Experience of the game:

F. Human-Adapted MDP

Optimal-human MDP Random-human MDP Worse-human MDP

$$\begin{aligned} E(X|NBA, DTG, XP) = \alpha_{NBA} * E(X|NBA \\ + \alpha_{LA} * E(X|DTG) \\ + \alpha_{XP} * E(X|XP) \end{aligned} \tag{7}$$

Finally, to decide on the best action we use a Monte Carlo approach to solve the MDP using the transition probabilities obtained the aforementioned method.

G. Trust update

For this player θs are stored. But update the average lookahead updated. Update the average player that should be the model for a trust of $\tau = 0.5$.

After each game, the system learns the parameters of the average player.

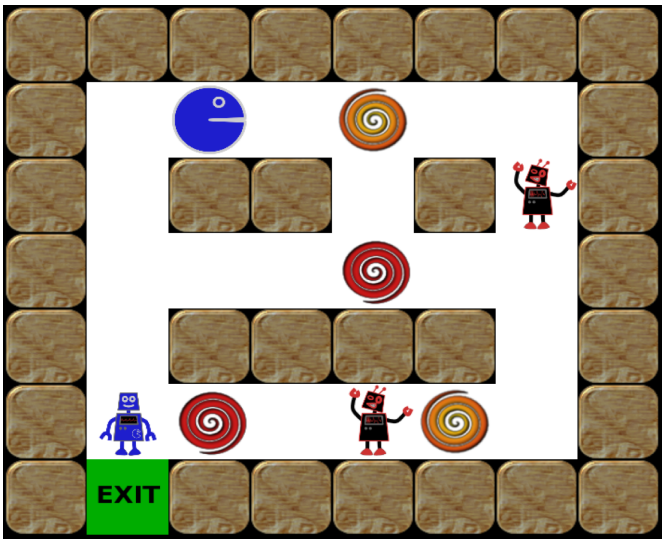

Figure 4: *Screen capture of the developed web-based game*

VI. Results

In order to validate the described approach along with the models, framework and algorithms, presented in the following, we developed a web-based game. To evaluate our approach, we developed the game "BlueMan and the bluebots" (see figure 4).

VII. Conclusion

Most of the computation is performed off-line. Online learning of the human ability. This ensures a game AI that work well out of the box, but allows for online adaptation to take place. For evaluation of the method we can compare performances of our method with the performance when the pursuer-robots execute one of the following policy:

- optimal policy considering the human also executes the optimal policy,
- optimal policy considering the human executes a random policy,
- optimal policy using the transition model of the human coming from the normalized Q-values.

Also, the quality of the parameters can be evaluated by comparing how the predicted transition model differs from real human action selection.

As pointed out, the game has been set up both in a virtual environment (on the web) and in a real-world environment (in this case the setting is quite preliminary setting). Other than studying modeling and learning aspects for human-swarm peer cooperation, the "real" game environment is expected to be engaging and providing a different experience compared to the virtual game. One of the final goals of this work is precisely to set up a an exciting game environment to let humans play together and against swarms of robots. Once the robots move fast enough (with the help of a tracker) and adequate lighting and sound effects are in place, a number of interesting cognitive and decision aspects can be studied. For instance, such a settings could be used to "train" children to perform complex decision making in 3D environments

and develop trust and adapting to robot behaviors. Moreover, the game environments could reflect complex situations like emergency support and used to train professionals to get acquainted to robots and act cooperatively and in synergy with them.

REFERENCES

[1] T. Chung, G. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Autonomous Robots*, vol. 31, no. 4, pp. 299–316, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10514-011-9241-4

[2] M. A. Vieira, R. Govindan, and G. S. Sukhatme, "Optimal policy in discrete pursuit-evasion games," Tech. Rep., 2008.

[3] ——, "Scalable and practical pursuit-evasion with networked robots," *Intelligent Service Robotics*, vol. 2, no. 4, pp. 247–263, 2009. [Online]. Available: http://dx.doi.org/10.1007/s11370-009-0050-y

[4] G. Hollinger, A. Kehagias, and S. Singh, "GSST: anytime guaranteed search," *Autonomous Robots*, vol. 29, no. 1, pp. 99–118, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10514-010-9189-9

[5] M. M. Fard and J. Pineau, "Non-deterministic policies in markovian decision processes." *Journal of Artificial Intelligence Research (JAIR)*, vol. 40, pp. 1–24, 2011.

[6] Y. Nagayuki, S. Ishii, and K. Doya, "Multi-agent reinforcement learning: an approach based on the other agent's internal model," in *MultiAgent Systems, 2000. Proceedings. Fourth International Conference on*. IEEE, 2000, pp. 215–221.

[7] C. T. Tan and H.-l. Cheng, "An automated model-based adaptive architecture in modern games." in *AIIDE*, 2010.

[8] T.-H. D. Nguyen, D. Hsu, W. S. Lee, T.-Y. Leong, L. P. Kaelbling, T. Lozano-Perez, and A. H. Grant, "Capir: Collaborative action planning with intention recognition." in *AIIDE, Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

[9] O. Macindoe, L. P. Kaelbling, and T. Lozano-Pérez, "Pomcop: Belief space planning for sidekicks in cooperative games," in *Proceedings of the 8th Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, October 2012, best Paper Award. [Online]. Available: http://lis.csail.mit.edu/pubs/macindoe-aiide12.pdf

[10] A. R. Wagner, "The role of trust and relationships in human-robot social interaction," Ph.D. dissertation, Georgia Institute of Technology, 2009.

[11] J. Gonzalez, M. Anwar, and J. Joshi, "Trust-based approaches to solve routing issues in ad-hoc wireless networks: A survey," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, Nov 2011, pp. 556–563.

[12] L. Mui, M. Mohtashemi, and A. Halberstadt, "A computational model of trust and reputation," in *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, Jan 2002, pp. 2431–2439.

[13] Y. L. Sun, W. Yu, Z. Han, and K. R. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 305–317, 2006.

[14] S. Bhattacharya and A. Ghosh, "Article: Entropy trust based approach against ip spoofing attacks in network," *International Journal of Computer Applications*, vol. 67, no. 23, pp. 27–32, April 2013, published by Foundation of Computer Science, New York, USA.