Behavior Modeling From Learning Agents: Sensitivity to Objective Function Details

(Extended Abstract)

Robert Junges
Örebro University
70182 Örebro, Sweden
robert.junges@oru.se

Franziska Klügl
Örebro University
70182 Örebro, Sweden
franziska.klugl@oru.se

ABSTRACT

The process of finding the appropriate agent behavior is a cumbersome task – no matter whether it is for agent-based software or simulation models. Machine Learning can help by generating partial or preliminary versions of the agent low-level behavior. However, for actually being useful for the human modeler the results should be interpretable, which may require some post-processing step after the actual behavior learning. In this contribution we test the sensitivity of the resulting, interpretable behavior program with respect to parameters and components of the function that describes the intended behavior.

Categories and Subject Descriptors

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

General Terms

Design, Experimentation, Performance

Keywords

Multiagent Simulation, Agent Learning

1. MOTIVATION

The basic idea behind agent-based simulation is that phenomena are generated from simulation of mostly local, low-level actions and interactions of agents. In such a bottom-up approach, a central issue concerns what behaviors the agents must exhibit so that the intended outcome is produced. Currently, dependent on the experience of the modeler, the development of an agent-based simulation may result in a painful trial and error process. The goal is to develop a systematic way of bridging the gap between agent behavior and macro-level outcome.

Our idea is to support the process of designing the agent behavior using self-adaptive agents [2]. A human modeler shall focus on describing the targeted phenomenon as well as the overall simulation settings, including the interfaces between environment and agents.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

A basic assumption hereby is that it is possible to characterize good performance in terms of an objective function, also in scenarios in which the features of the actually necessary behavior program are not exactly known. As in every learning approach, the definition of the objective function essentially drives the generation of the agent behavior. Our application of learning is special as we combine traditional Reinforcement Learning, using the given objective function, with a post processing step – a decision tree learner – in which the behavior model shall be abstracted to a human-readable representation. The central question in our contribution is therefore how robust the final behavior outcome is with respect to small changes in the objective function.

2. LEARNING FOR MODELING

As stated in the introduction, we propose a learning-driven analysis and design approach using self-adaptive agents in the behavior modeling task for simulation models. The actual optimality of the learnt agent control is only one relevant criterion; the interpretability of the outcome by a human is essential.

We start with the definition of an environmental model and a set of sensors and actuators that determine what the agents are able to perceive and manipulate. The second step is the definition of a learning architecture that is apt to connect perceptions and actions of the agent. After that, a reward function, providing feedback to the agents, is defined. The reward has to measure performance, as the agents will use it to explore how the environment reacts to their actions. The resulting behavioral model should then be analyzed by the human modeler for preventing artifacts that come from an improper environmental or reward model, or weak interfaces.

The chosen learning architecture for this contribution combines Reinforcement Learning with Decision Tree learning.

We selected Q-learning for our investigation because it is the simplest reinforcement learning technique directly producing situation-action pairs. However, even in simple scenarios the high number of pairs prevents a designer to oversee the actually learnt behavior. For tackling this readability problem we use a decision tree representation of the implicit behavior of the best situation-action pairs. In this contribution we selected the C4.5 algorithm to generate decision trees, which are a well-suited representation model for decision-making processes [1].

The best situation-action pairs are taken by first excluding those pairs that haven't been tested enough, as the confidence on their expected utility is lower. Then, we select for each situation the action with the highest Q-value, considering only those with non-zero, positive Q-values. The generated decision tree basically accomplishes the lacking abstraction that makes the resulting behavior description transparent for the designer.

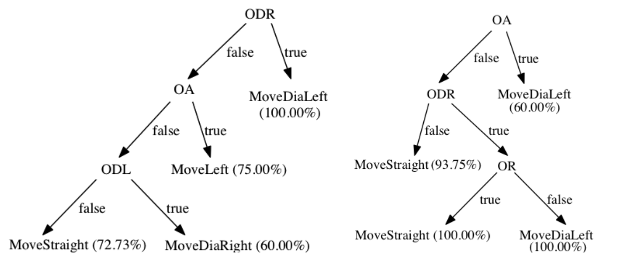
3. EXPERIMENTAL SETUP

Our test scenario is a pedestrian evacuation model. The environment is represented by a room, a number of round obstacles and one exit. The agents are randomly placed in the half on the opposite to the exit. They have two objectives: leave the room as fast as possible and do not collide with obstacles or other agents. Perception is discretized into sectors, actions according to movement directions. A reward is given to each agent individually after each step containing the following components: (1) *Exit Reward* indicating whether the exit was reached; (2) *Collision Reward* punishing collisions; and (3) *Distance Reward* indicating whether the agent came closer to the exit. More details can be found in [3]. For testing the consequences of different setups to the overall outcome we focussed on the relation between the different components indicating the pressure towards/against particular situations: a) when one or more elements are not considered; b) when one elements contributes half or twice as much as the others; c) when all contribute with the same weight.

4. RESULTS AND DISCUSSION

We systematically run experiments with different configurations for the objective function. The agents learnt to sufficiently perform, that means move directly to the exit while avoiding fixed and dynamic obstacles. Only little differences in the performance measured in terms of number of collisions were observable. Naturally, the best performance was measured when the weights of the *Collision Reward* were higher than for *Distance Reward.* After that, using the decision tree learner for generalization, we tested how well the classification result from the decision tree resembles the originally learnt behavior. It turned out that the best preservation of information at the generalization step occurred when the *Distance Reward* was weighted higher than the *Collision Reward.*

Although performance and accuracy measures were almost the same for the different settings, the resulting decision trees were quite different. With a higher *Distance Reward*, we can see that the agent tend to develop actions that lead to shorter paths, at the same time as they try to avoid collisions. A shorter path means the selection of movements that direct the agent to a sector closer to the obstacle. This is different in the case with higher *Collision Reward*: the decision tree points to a selection of perceptions and actions that lead to the development of a wider collision-avoidance path. This comes from the fact that is hard to predict other agents' movements as agents cannot distinguish between pedestrians and columns. If the weight on the collision avoidance is higher, the agents learn to be more "cautious". They take wider deviations from the direct route to avoid eventually colliding with another agent, at the expense of evacuation time. That means finally that those trees are more elaborated than when learning with the other configurations - this can be seen in Figure 1. The codes in the nodes correspond to different perceptions: *O* for Obstacle, *D* for Diagonal, *A* for Ahead, *L* for Left and *R* for Right.



(a) Collisions (b) Evacuation

Figure 1: Behavior trees for different objective function details putting more focus on collision avoidance or fast evacuation.

5. CONCLUSION AND FUTURE WORK

Testing the robustness of the finally abstracted behavior output with respect to variations of the initial objective function showed a high sensitivity. Due to the abstraction and generalization done using the decision tree learner, we expected more robustness. However this result shows the relevance of a careful formulation of the criteria for valid agent behavior. Really surprising is that the complexity of the resulting behavior program is not mirrored in the classical numerical metrics that describe learning performance. This makes the formulation of the objective function describing what the agents shall achieve even more critical than describing how they should achieve it.

These results lead to next steps for establishing agent learning as a tool for agent simulation design. First, we have to analyze the learnt behavior more directly by controlling agents using the decision trees generated. This will show whether they actually perform in the intended way or whether too much information has been lost. This may lead us to testing other generalization techniques than the simple decision tree learner that we used. There is a lot of research going on for state abstraction in reinforcement learning. Although not aiming at readability of the abstracted program, they might be applicable in our case. A second future direction of our research directly addresses the formulation of the objective function: Instead of formulating an objective function, we may use learning by demonstration and imitation techniques for directly mapping observable actor or stakeholder behavior to generate a behavior program for the corresponding simulated agent.

6. REFERENCES

[1] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51:141 – 154, 2011.

[2] R. Junges and F. Klügl. Generating inspiration for agent design by reinforcement learning (to appear). *Information and Software Technology*, 2011.

[3] R. Junges and F. Klügl. Programming agent behavior by learning in simulation models (to appear). *Applied Artificial Intelligence*, 2012.