From Global Selective Perception to Local Selective Perception

Sébastien Paquet, Nicolas Bernier and Brahim Chaib-draa
DAMAS Laboratory, Laval University, Canada
{spaquet;bernier;chaib}@damas.ift.ulaval.ca

Abstract

This paper presents a reinforcement learning algorithm used to allocate tasks to agents in an uncertain real-time environment. In such environment, tasks have to be analyzed and allocated really fast for the multiagent system to be effective. To analyze those tasks, described by a lot of attributes, we have used a selective perception technique to enable agents to narrow down the description of each task, enabling the reinforcement learning algorithm to work on a problem with a reasonable number of possible states.

1. Introduction

Analyzing many possible goals and choosing the best one is obviously a difficult task for an agent, particularly in a dynamic and uncertain environment. In this case, an agent must choose a goal with only a partial view of the situation and do so quickly, because the environment is constantly changing. To do that, the solution proposed here is to use a reinforcement learning algorithm in which the agent learns the expected reward it should get for every possible goal it could be faced with. To reduce the number of states, we have used a selective perception technique to enable the agent to learn which are the most useful characteristics to perceive in all possible situations [2].

We have tested our algorithm in the RoboCupRescue simulation environment in which the goal is to build a simulator of rescue teams acting in large urban disasters [1]. More precisely, this project takes the form of a competition in which participants are designing rescue agents trying to minimize damages, caused by a big earthquake, such as civilians buried, buildings on fire and blocked roads. In this paper, we focus only on the *FireBrigade* and *FireStation* agents' task which is to extinguish fires.

To accomplish the extinguishment task, we have used the selective perception algorithm at two different levels. First, the *FireStation* agent is using the algorithm to allocate a zone on fire for each *FireBrigade* agent. Then, those agents are choosing the best fire to extinguish in this zone. This approach enabled the multiagent system to learn how to extinguish fires by taking advantage of the global view of the *FireStation* agent and at the same time, the local and more precise view of the *FireBrigade* agent. Furthermore, during the simulation, each agent tries to maximize its reward which depends on the success of the extinguishment and the number of buildings and civilians saved.

2. Selective Perception

With the selective perception reinforcement learning algorithm, the *FireStation* agent and the *FireBrigade* agents learned by themselves to reduce the number of possible states, by distinguishing only states that need to be distinguished. To be more precise, the algorithm uses a tree structure similar to a decision tree. At the beginning all states are considered to be the same, so there is only the root of the tree. After some experiences, the agent tests if it would be interesting to divide the different states, represented as the leaves of the tree. To divide a leaf, it tries to divide its experiences, stored in that leaf, by making a test on an attribute. By doing so, the agent creates new states, by expanding a leaf, and thus it refines the view it has about the state space.

The algorithm presented here is an instance-based algorithm in which a tree is used to store all instances which are kept in the leaves of the tree. To find the leaf to which an instance belongs, we simply start at the root of the tree and head down the tree choosing at each center node the branch indicated by the result of the test on the instance's attribute value. Each leaf of the tree also contains a Q-value indicating the expected reward if a fire that belongs to this leaf is chosen. In our approach, a leaf l of the tree is considered to be a state for the reinforcement learning algorithm.

At each time step t, the agent records its experience captured as an "instance" that contains the observation it perceives (o_t) and the reward it obtains (r_t). Each instance also has a link to the preceding instance and the next one, thus making a chain of instances, one for each building an agent chooses to extinguish. The agent keeps all those instances until the end of the simulation, since there is no time for learning during the simulation.

After a simulation, all agents put their new experiences together. All those new experiences are then added to the leaf of the tree they belong to. Afterwards, we update the Q-values of each leaf node to take into consideration the new instances which were just added:

$$Q(l) \leftarrow R(l) + \gamma \sum_{l'} Pr(l'|l)Q(l') \qquad (1)$$

where $R(l)$ is the estimated immediate reward if a fire that belongs to the leaf l is chosen, $Pr(l'|l)$ is the estimated probability that the next instance would be stored in leaf l' given that the current instance is stored in leaf l. Those values are calculated directly from the recorded instances. $R(l)$ is the average reward obtained when a fire belonging to this leaf was chosen and $Pr(l'|l)$ is the proportion of next instances that are in leaf l'.

When the Q-values have been updated, we check all leaf nodes to see if it would be useful to expand a leaf, thus dividing the instances more finely. To do so, we try all possible tests on all attributes and we choose the attribute that maximizes the error reduction as shown in equation 2 [3], if it is greater than a certain threshold. The error measure considered is the standard deviation ($sd(I_l)$) on the instances' expected rewards. The expected error reduction obtained when dividing the instances I_l of leaf l is calculated with the following equation where I_d denotes the subset of instances in I_l that have the d^{th} outcome for the potential test:

$$\Delta error = sd(I_l) - \sum_{d} \frac{|I_d|}{|I_l|} \times sd(I_d) \qquad (2)$$

During the simulation, the agents are using the tree created offline to choose the best fire zone and the best building on fire to extinguish. Since that the *FireStation* agent has a better global view of the situation, it is its responsibility to suggest fire zones to *FireBrigade* agents. And because those agents have a better local view, they are choosing which particular building on fire to extinguish. By doing so, we can take advantage of the better global view of the *FireStation* agent and the better local view of the *FireBrigade* agent at the same time.

To allocate the fire zones, the *FireStation* agent has a list of all fire zones. For each fire zone it knows about, the center agent retrieves from the tree all expected rewards for all possible number of agents. It begins with one agent and it adds one agent until the expected reward becomes greater then a specified threshold. The *FireStation* agent does the same thing with all fire zones, ending up with an expected reward and a number of agent for each zone. Then, it chooses between all zones, with an expected reward greater than the threshold, the one that minimizes the distance of the required *FireBrigade* agents.

When choosing a fire to extinguish, the *FireBrigade* agent has a list of buildings on fire it knows about in the fire zone specified by the *FireStation*. For each building on this list, the agent finds the leaf of the tree to which this building belongs and records the expected reward associated with this leaf. The chosen building is the one that has the greatest expected reward.

3. Experimentations

In our experiments, the state space for the task of choosing a fire zone contained 150 000 states and for the task of choosing a building on fire in the fire zone it contained 875 000 states. The results we have obtained show a drastic reduction of the state space, since the agents have learned a tree with 67 leaves for choosing a fire zone and with 1719 leaves for choosing a building on fire.

With those trees that are small compared to the possible number of states, the agents were extinguishing fires efficiently. However, we have observed some variations in the results from one simulation to another, partly due to the uncertainty in the simulation. In short, the learning algorithm has helped our agents to learn how to choose a building on fire, but the uncertainty in the environment is still hard to manage and it slows down the learning process.

4. Related Works and Conclusion

The approach presented is inspired by the work of McCallum in his thesis [2]. In our work, we use approximately the same tree structure to which we have made some modifications. Firstly, we do not use the tree to calculate Q-values for every possible basic actions. Another difference is in the way the leaves are expanded, because instead of generating all possible subtrees, we use an error reduction estimation measure. Like Uther and Veloso [4] with their Continuous U-Tree algorithm, we also support continuous attributes, but with a different splitting criteria. We also have more than one chain, therefore our concept of instance chain is closer to the concept of *episode* described in [5].

To summarize, this article has presented an efficient algorithm to enable the agent to learn the best decision to take at the goal decision level. We have shown how we can adapt a reinforcement learning algorithm to the problems of task allocation in a real-time and the uncertain environment. Our results show that agents are able to obtain good results with trees having a very small number of leaves.

References

[1] H. Kitano. Robocup rescue: A grand challenge for multi-agent systems. In *Proceedings of ICMAS 2000*, Boston, 2000.

[2] A. K. McCallum. *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

[3] J. R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of ICML'93*, pages 236–243, 1993.

[4] W. T. B. Uther and M. M. Veloso. Tree based discretization for continuous state space reinforcement learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 769–774. AAAI-Press/MIT-Press, 1998.

[5] P. Xuan, V. Lesser, and S. Zilberstein. Modeling Cooperative Multiagent Problem Solving as Decentralized Decision Processes. *Autonomous Agents and Multi-Agent Systems*, 2004.