A robust MAS coordination mechanism for action selection

Dídac Busquets, Ramon López de Màntaras and Carles Sierra
Artificial Intelligence Research Institute (IIIA),
Spanish Council for Scientific Research (CSIC)
Campus UAB, 08193 Bellaterra, Barcelona, Spain
{didac, mantaras, sierra}@iiia.csic.es

1. INTRODUCTION

Several systems (usually robotics) are controlled by a multi-agent system (MAS). In such systems, the group of agents is responsible of deciding which action the system has to take. The different methods used to select the action are known as *action selection mechanisms*, and there is a wide range of them in the literature (a review of coordination mechanisms can be found in [2]).

We have focused on developing an action selection mechanism within the frame of a robot navigation system. One interesting characteristic of such a system is that there exists a clear ditinction between the agents that have a global view of the task to be performed (i.e. navigate to reach a target) and those that have local views (such as avoiding obstacles).

The agents with global views have a wider perception of the environment than those with local views, and they have knowledge for solving the task. On the other hand, the agents with local views have a very restricted perception and are only capable of solving local situations.

This distinction is also applicable to the kind of contribution the agents do to the system. The agents with global views contribute in reaching the goal and, talking in terms of rewards, maximise the benefit associated to the robot performance, while the agents with local views contribute in solving specific problems and minimise local costs that will affect the final benefit for the system.

In this paper we present an action selection mechanism based on this idea. The agents in the MAS are divided in two groups, global agents and local agents. Combining the different views of the agents, the action that is found to be the most beneficial in order to reach the goal is selected. The aim of this mechanism is not finding optimal behaviours, but robust ones.

2. ACTION SELECTION MECHANISM

As actions are performed, new states of the task to be solved are reached. Thus, solving the task results in going through a sequence of states, starting at an initial state and having to reach the goal state.

As in reinforcement learning models, we assume that accomplishing the task has a reward for the system. Reaching this goal is certainly not cost free, the actions performed by the system have an associated cost that will decrease the total final reward. Therefore, the main objective of the system is to reach the goal by incurring the minimum cost.

The mechanism works as follows. Each agent evaluates each possible action and computes the utility of performing that action (in a similar way as is done in [3]). These utilities are then send to a coordinator agent that combines the utilities from all the agents to determine which is the most useful action. This action is then performed, and a new state of the task is reached. Then the process starts again: each agent evaluates the current state of the task, sends its utilities and one action is selected.

3. TASK REPRESENTATION

As mentioned in the previous section, the system will go through several states. These state transitions show the evolution of the task resolution and are used by the agents to compute the utilities of the different actions.

Any state is defined as a vector of values over variables, which can be of any type (real values, interval values, qualitative values,...):

$$V = \{x_0 : \tau_0, x_1 : \tau_1, ..., x_n : \tau_n\}$$

the set of states is named S, and a concrete state $s \in S$ is a sequence of values of the form

$$s =< v_1, ..., v_n > s.t. \quad v_i : \tau_i$$

However, each individual agent does not perceive the whole state representation. It only perceives a subset of the state variables, depending on its specific task solving capability. So the state view by agent α would be based on:

$$V^\alpha = \{x_0, x_1, ..., x_m\}, V^\alpha \subseteq V$$

4. UTILITY COMPUTATION

We differentiate two types of agents in our system: agents with *global views* and agents with *local views*. Agents with global views over the environment, or global agents, are those whose goal is to accomplish the task (e.g. an agent responsible of moving a robot to a target position) and are in charge of maximizing the benefit. Agents with local views, or local agents, are those whose main goal is not to accomplish the task, but to satisfy certain properties when local state changes take place (e.g. an agent responsible of avoiding bumping into nearby obstacles).

Global agents can compute an expected gain on performing an action from the reward and the potential cost given its view on how to solve the problem. On the other hand, local agents can only have a notion of cost for reaching adjacent states, but are blind with respect to the final reward.

A global agent α computes the utility of state s_i as:

$$U^{\alpha}(s_i) = R - Cost^{\alpha}(s_i, G) \quad (1)$$

where R is the reward for accomplishing the task, common to all the agents, and $Cost^{\alpha}(s_i, G)$ is the cost of going from state s_i to the goal state, as seen by agent α.

Knowing how to compute the utility of any state, the global agent α is now able to compute the utility gain in performing an action a that changes state s_i into state s_{i+1}:

$$Gain^{\alpha}(a, s_i) = U^{\alpha}(s_{i+1}) - U^{\alpha}(s_i) - Cost^{\alpha}(s_i, s_{i+1}) \quad (2)$$

We assume deterministic actions. The agents must have a function $nextState : state \times action \rightarrow state$, that gives the resulting state of performing a certain action in a given state. In formula (2), $nextState(s_i, a) = s_{i+1}$. By combining formulas (1) and (2) we obtain:

$$\begin{aligned} Gain^{\alpha}(a, s_i) &= Cost^{\alpha}(s_i, G) - Cost^{\alpha}(s_{i+1}, G) \\ &\quad -Cost^{\alpha}(s_i, s_{i+1}) \end{aligned} \quad (3)$$

that relates the gain to costs, without the need of knowing the reward.

The local agents, on the other hand, have no knowledge about the overall task, so they cannot compute any benefit nor state utility. Instead, they can compute their view of the expected cost of performing an action:

$$\begin{aligned} ExpCost^{\alpha}(a, s_i) &= P^{\alpha}(s_{i+1}) Cost^{\alpha}(s_i, s_{i+1}) \\ &\quad - \sum_{s' \neq s_{i+1}} P^{\alpha}(s') Cost^{\alpha}(s_i, s') \end{aligned} \quad (4)$$

where $P^{\alpha}(s)$ is the probability of the state s being in the optimal solution to reach the goal, from the point of view of agent α, and $s' \in \bigcup_{a'} nextState(s_i, a')$ represents any other reachable state from s_i by performing some other action a'.

Once the global agents have computed their estimated benefits and the local agents their estimated costs, they send this information to a coordinator agent that will determine the winning action.

5. COMBINING GAINS AND COSTS

The coordinator agent receives the information from each agent, and it has to combine them in order to decide which is the best course of action. For each action, it receives utilities from global and local agents. The action overall utility is computed as:

$$AU(a) = \max_{\alpha}\{Gain^{\alpha}(a)\} - \min_{\beta}\{ExpCost^{\beta}(a)\}$$

Taking the maximum gain and the minimum cost is an optimistic view. We could have the pessimistic view by swapping these operators.

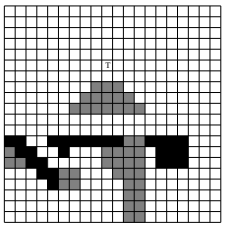

Figure 1: Map. Black squares are obstacles, grey squares represent bad terrain, white squares are free space and T is the target point.

The action with the maximum overall utility is the one selected to be performed:

$$selected\_action = \arg\max_{a} AU(a)$$

6. EXAMPLES

We have tested our approach on a toy example. The task to be solved is to move a robot from an initial point in a grid to a target point (see Figure 1), without bumping into the obstacles and with minimum cost.

The actions that can be performed are the movements of the robot from the current position to any of the four neighbouring non-diagonal cells (except obstacles and walls). The robot can make some observations of the environment: it can detect the characteristics of the adjacent areas (whether it is an obstacle, difficult terrain or free space).

The state represents the position of the robot in the grid. The current position of the robot and the position of the target are known at any time.

The MAS in control of the robot is composed by the following two agents:

- *Target Tracker*: its goal is to move the robot to the target point. It is able to construct a map of the environment, updated every time an obstacle is detected, and compute the Manhatan distance to the target taking into account these obstacles. It is able to perceive the position of the robot and the target. It is a global agent, and its gain function is:

  $$\begin{aligned} Gain(a, s_i) &= Dist(s_i, T) \cdot SC - Dist(s_{i+1}, T) \cdot SC \\ &\quad -SC \end{aligned}$$

  where $Dist(s, T)$ is the Manhatan distance from position s to the target, computed using the information stored in the map, and SC is the cost of moving the robot from one area to an adjacent one (and thus, $Cost(s_i, s_{i+1}) = SC$, for any s_i, s_{i+1}).

- *Terrain Assessor*: its goal is to avoid obstacles and difficult terrains. It can perceive the characteristics of the neighbouring areas. It is a local agent, and its cost function is:

  $$\begin{aligned} ExpCost(a) &= P(s_i) Cost(s_i, s_{i+1}) \\ &\quad - \sum_{s' \neq s_{i+1}} P(s') Cost(s_i, s') \end{aligned}$$

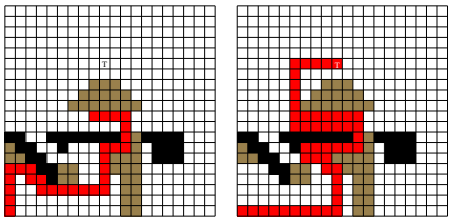

Figure 2: Sample run. The path is shown in darkest grey. The ellipsis shows the area of looping (left). Sample run with new probability function (right).

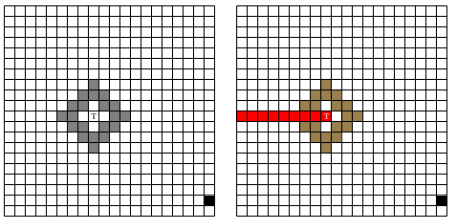

Figure 3: Target point surrounded by a difficult area (left). Sample run with the *Target Tracker* agent (right).

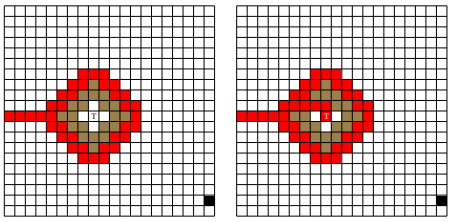

Figure 4: Sample run with both *Target Tracker* and *Terrain Assessor* (left). *Terrain Assessor* with modified probability function (right).

These costs depend on the perceived characteristics of the neighbouring areas. We have set all the probabilities $P(s)$ to $\frac{1}{4}$ since there are four possible states to be reached and any of them could be the best one.

A sample simulation is shown in Figure 2. The starting point is the bottom left corner.

As it can be seen, the robot has not reached the target. The problem is that the robot enters a loop (see the encircled region) by moving from left to right and vice versa indefinitely.

To avoid this behaviour, we have changed the probability function of the *Terrain Assessor* agent. Instead of giving the same probability to every state, it will depend on how many times the robot has visited that state. The more visits, the less probable. This way we are favouring the exploration of new paths. A sample run with this new function is shown in Figure 2. As it can be seen, the robot has now reached the target, after having done some exploration.

It should be noted that the robot did not find the optimal path. This could only be done having complete information about the environment, which is not the case. What we are trying to do is having a robust behaviour, able to cope with any situation the robot may encounter while trying to accomplish the task.

Another example run is shown in Figures 3-4. The target is surrounded by a difficult area. If we only had the *Target Tracker* agent, it would go straight to the target (Figure 3). If we add the *Terrain Assessor* with equal probabilities for each state, the robot is constantly going around the difficult area, but never enters it (Figure 4). If we use the probability function depending on the visits, the robot goes once around the difficult area, and then it realizes that there is no other way to go to the target than going through the difficult area (Figure 4). Although in this case the final reward would be less than going straight to the target, it would have done better if a clear path existed somewhere around the target.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a new action selection mechanism that takes into account the different roles of the agents in a MAS, with the main objective of adding robustness to the system. These roles can be either *benefit-maximiser* or *cost-minimiser*, and can be found in many systems governed by a MAS. As a typical system of this type, we have focused on a robot navigation system.

Although we have achieved successful results, it has been on a very simple example. We would like to use the mechanism presented in this paper on a more complex and real scenario. In particular, we want to use it in the robot navigation system described in [1]. In this previous work we used the idea of action selection through bidding, but the bids were not defined in terms of utilities, gains or costs, but specific "urgency" functions for each agent.

However, the addition of the new mechanism is not straightforward. The difficulty lies in the state representation. In the example shown previoulsy, the robot had perfect knowledge about its location and the target's positions, and the effects of the actions were clearly defined. In the new robot system there is no precise knowledge about the location of the robot, and the environment is not a grid (the navigation system uses a topological map instead), which adds more difficulty to the definition of states transition.

8. REFERENCES

[1] R. López de Màntaras C. Sierra and D. Busquets. Multiagent bidding mechanisms for robot qualitative navigation. In *Intelligent Agents VII. Lectures Notes in Artificial Intelligencee (Proceedings of ATAL-2000).* Springer, Verlag, In press.

[2] G. Weiss (ed.). *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence.* The MIT Press, Cambridge, MA, 1999.

[3] Julio Rosenblatt. Optimal selection of uncertain actions by maximazing expected utility. *Autonomous Robots*, 9:17–25, 2000.