Controlling co-incidental non-player characters

Joseph Walton-Rivers
Computer Science Department
University of Essex
Colchester, CO4 3SQ UK
jwalto@essex.ac.uk

Abstract

This document describes some of the problems encountered when trying to create believable Non-Player Characters (NPCs). It also describes the tiny coop domain and its use for testing of co-operative agents. This paper also describes the use of modelling the other agent as part of the agent's strategy and its effects for the Tiny Coop domain.

Introduction

Non-Player Characters in Role Playing Games have been criticised by both researchers and players for their lack of social capabilities (Afonso and Prada 2008).

Believable agents are agents which act like they are part of the world (a.B. Loyall et al. 1997). Livingstone chose to define believable agents along the lines act, plan and sense (Livingstone 2006). Both of these definitions feature components relating to social capabilities of the agent. To be more believable, the agents need to be able to co-ordinate and co-operate with each other and the player.

Multi-agent domains

Multi-agent domains can be divided into 'distributed' and 'centralised' planning. In the first case, each agent is controlled by their own controller. The second case describes agents that are are controlled by a single over-arching controller (Kovacs 2012).

Problems with multi-agent planning

Complexity Agents add complexity, state space is $agent1 \times agent2$ actions. The complexity of the multi-agent version of a planning problem depends on the largest number of actions which that agent must execute.

Plan Invalidation Another agent may invalidate the actions of our agent by accident, requiring us to re-plan and come up with better ways of doing things.

Co-ordination Between Agents co-ordination between agents If the agents actions depend on the actions of another agent, then the agent must ensure that the other agent has executed its part of the joint plan before it can continue. This also requires that the other agent follows the same plan and is able to complete its part of the joint plan.

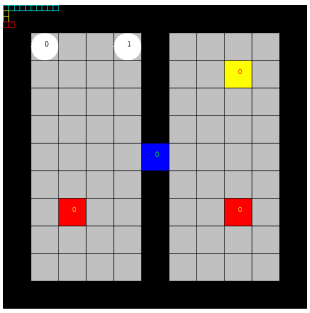

Figure 1: TinyCoop: Single Door Level

Tiny Coop

Role playing games are complex domains which usually feature a large number of agents, have a lot of features and are extremely large state spaces. In order to study co-operative behaviours we created a simple 2 player puzzle game and showed that Monte Carlo Tree Search (MCTS) was capable of solving the puzzles (Williams et al. 2015).

The domain features 2 individually controlled agents (white circles) which both must get to all the goals (marked in bright yellow) within the level. The agents need to navigate though doors (marked in blue), the doors will only open if the other agent is standing on the corresponding button (marked in red).

The agents can chose to move in one of the four cardinal directions, chose not to move or chose to communicate a direction to the other agent. This communication of a direction is the only explicit form of communication the agents have available. Agents are given 40 milliseconds to make their move, else they will stay at their current position.

The game provides a forward model to allow the agent to simulate the effect of a move. As the agents moves are made simultaneously, the forward model requires that both agents moves be provided (the policy of the other agent) when attempting to advance the state.

Policies

A number of different agents were used with the Tiny Coop domain. These were paired with different agents for the evaluation.

Random Policy

The simplest policy is to assume that the other agent will select random moves. This is not overly effective because it does not allow the agent to exploit the behaviours of the other agent in its plans.

Predictor Policy

This policy attempts to simulate accurately modelling the paired agent by using a copy of the other agent to the agent (*the predictor policy*). The copy and the 'real' paired agent cannot communicate and is initialised with a different random seed in the event the agent has random element to its action selection process. It therefore mimics the decision making process of the agent without access to its internal state.

Paired Agents

There are a number of paired agents which I am using for my research. The graph (fig. 2) shows the results of pairing the predictor and random policies.

Random This agent selects an action at random from the set of legal actions. The random policy and the predictor agent both assume (correctly) that the paired agent moves randomly, as a result, both agents have the same performance.

Follow the flare This agent moves in the direction indicated by the other agents communication action from the last game tick. The agent which utilised prediction was able to out-perform the standard MCTS controller. The predictor agent is able to exploit the behaviour of the flare follower agent whereas the standard MCTS approach is not able to do so and was therefore able to obtain a better score.

MCTS The paired MCTS controller used a fixed roll-out budget and therefore made the predictor MCTS agent overspend its time budget. As a result, the predictor was penalised in the form of no-operations being applied as its actions. The random predictor does not have this problem and as a result is able to achieve much better scores.

Bias Random This agent selects a random move, but biases the selection towards movement actions. The random policy agent is still a fairly accurate model of this agent's behaviour and therefore the agent performs quite well. The predictor policy agent knows that the agent is less likely to issue a communication action than move and therefore can factor that into its plans.

Future work

The use of accurate modelling shows promise but requires more work. The technique used at present is not a viable technique in a role-playing game environment.

State space abstraction

If the agents were to work with a more abstract, high-level description of the problem it would be able to explore more of the state space during its roll-outs.

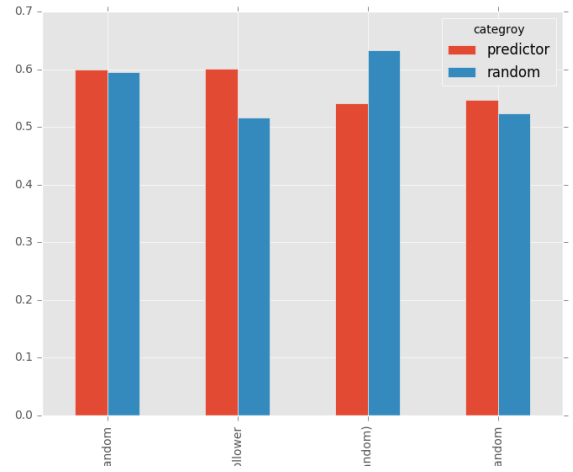

Figure 2: Scores of the predictor vs random policies

This reduces the number of states which exist within the world and makes it easier to find a valid solution to the problem. This high level abstract plan can then be used to guide the agent though the level.

Agent abstraction

The agent currently has a model of the other agent which features the actions the other agent will do in a given situation. The predictor requires advance knowledge of the agent it is paired with. This is not always possible. The agent should be able to generalise about the strategies it has encountered and predict what strategy the paired agent is using.

Acknowledgments

This work was funded by the EPSRC Centre for Doctoral Training in Intelligent Games & Game Intelligence (IGGI) [EP/L015846/1].

References

a.B. Loyall; Loyall, A.; Bates, J.; and Bates, J. 1997. Personality-rich believable agents that use language. (May):106–113.

Afonso, N., and Prada, R. 2008. Agents that relate: Improving the social believability of Non-Player Characters in Role-Playing Games. In *7th International Conference, Pittsburgh, PA, USA, September 25-27, 2008. Proceedings*, volume 5309 LNCS, 34–45. Springer.

Kovacs, D. L. 2012. A Multi-Agent Extension of PDDL3. 19.

Livingstone, D. 2006. Turing's test and believable AI in games. 4(1):6.

Williams, P. R.; Walton-rivers, J.; Perez, D.; and Lucas, S. M. 2015. Monte Carlo tree search applied to co-operative problems. In *Proceedings of the IEEE Computer Science and Electronic Engineering Conference (CEEC)*, 219–224.