Exploring the emergence of learned behaviour from intelligent agents in a constrained 3D spatial environment

Sedar Olmez[1,3], Alison Heppenstall[1,3], and Daniel Birks[2,3]

[1] School of Geography, University of Leeds
[2] School of Law, University of Leeds
[3] The Alan Turing Institute

Abstract. The original Sugarscape agent-based model has been adopted by the modelling community as a simple yet very insightful model. It uses event-condition-action rules to simulate a society which ages over time and consumes sugar to survive. The model is well known and has been adopted in fields such as Economics, Computer Science and Biology therefore its utility will be explored in this research. The research will present an implementation of the original Sugarscape model (refined for 3D space) using Reinforcement Learning (a form of Machine Learning which enables autonomous agents to learn by rewards and penalties). A Reinforcement Learning algorithm such as Q-learning is implemented in this research as there is a lack of work being done to integrate Q-learning in conventional ABMs. Furthermore, Q-learning provides agents with the means to evolve overtime and adapt to changes without explicitly defining the actions to take under individual circumstances. On the other hand, agents that use event-condition-action rules do not adapt to change, they instead react to changes by applying actions that have been hard coded prior to model execution. The problem addressed is, can agents make more realistic decisions by learning from their environment using reinforcement learning? This research will explore reactive and adaptive behaviours to see if actions performed by agents within the given environmental circumstances would be as expected if they were humans. Furthermore, agents can anticipate future trends, for example; learning the fastest route out of a burning building while anticipating a fire alarm. This model will be implemented in Unity which is a video game development engine. Unity has not been widely adopted by the agent-based modelling community, yet does provide useful facilities for the creation of models.

Keywords: Agent-Based Model · Reinforcement Learning · Decision Making.

1 Background: Agents and Learning Algorithms

Learning algorithms such as Q-learning [12] have been around for many years. Signs of the emergence of these algorithms can be traced back to early video

games. Early ABMs applied condition-action-rules, then came BDI [9] and then hybrid BDI frameworks such as [5], PECS (Physical conditions, Emotional state, Cognitive capabilities and Social status) [11] and so on. It is clear that frameworks for ABMs are constantly evolving. During this period, Machine Learning algorithms are also being developed. Reinforcement Learning algorithms (a subset of Machine Learning algorithms) are not domain specific, meaning they can be adopted by any system that requires autonomous control of software that makes decisions overtime (mainly good decisions). There is a gap in the research being done to supply ABM frameworks with the ability to allow agents to adapt to changes during model execution (which is usually what happens in the real world, we constantly adapt to changes in our lives that we may never have expected i.e. adapting to the death of a loved one). Moreover we evolve overtime, we learn new skills and apply these skills that we never knew before. Reinforcement Learning provides agents with these traits. This research aims to apply a Reinforcement Learning algorithm to agents within a defined environment (Sugarscape [2]) then, analyse these agents to see if they can adapt to changes overtime and evolve.

1.1 Q-learning

Reinforcement Learning algorithms are used to provide agents in models with policies to execute within a given circumstance. These algorithms can be applied to any model that has a goal which needs to be fulfilled. For example; modelling driver behaviour where agents (cars) need to navigate from point A to point B without causing congestion. Large rewards will be given to cars that follow a route and penalties will be incurred if cars collide. As we run the model over many iterations, the agents start by making mistakes but each iteration they learn something not to do and something that is acceptable.

Q-learning has been utilised in various domains and is an option for agent based modelling researchers. [14] used Q-learning to calibrate an agent based supply network model by using agents to find optimal values for parameters in their operating policies. Moreover, the competitiveness of the electricity market is modelled using agents that utilise Q-learning to learn from past actions and deploy strategies against other competitors to ensure a fair distribution of electricity among suppliers [7]. [10] implemented the Q-Learning algorithm in order to model the bidding strategy of suppliers (agents) in electricity auctions. The authors examined the change in policy under various conditions of demand.

The Q-learning algorithm referred to throughout this research is from [13]. Q is initialised to a value provided by the programmer, and at each time step t an agent selects the action a_t. It then observes a reward r_t, and enters a new state s_{t+1} (this depends on a previous step s_t and the selected action a), finally Q is updated.

$$Q^{new}(s_t, a_t) \leftarrow (1 - a) \cdot Q(s_t, a_t) + a \cdot (r_t + \gamma \cdot \max_a Q(s_{t+1}, a)) \tag{1}$$

Where, a is the learning rate ($0 < a \leq 1$), $Q(s_t, a_t)$ is the old value, r_t is the reward at time step t, γ is the discount factor ($0 \leq \gamma \leq 1$) it values rewards

received earlier, higher than those received later. Finally $amax\, Q(s_{t+1}, a)$ is an estimated optimal future value [13]. The learning rate a is defined as; how much do you accept the new value compared to the value learned previously. The difference between the new value and previously learned value is multiplied by the learning rate a, let us call this S. Finally, S is added to the previous Q-value which moves the agent in the direction of the latest update.

2 Methodology

2.1 Sugarscape

Sugarscape is an agent based model that simulates artificial societies. The idea was originally developed in [3] and several implementations of the model have been produced since [1, 4, 8]

The original sugarscape contains a heterogeneous population of autonomous agents that compete for renewable resources which are unequally distributed over a 2-dimensional environment [3]. Agents are autonomous, meaning there is no central guidance that makes them behave in specific ways. They are also heterogeneous, meaning each agent has unique traits and understanding of the environment (i.e. initial location and wealth). The environment contains randomly distributed sugar, some cells may contain no sugar and are classed as empty cells.

2.2 Unity

Unity supports many software engineering packages such as OpenAI's ml-agents [6]. These packages provide a wide range of learning algorithms that can be deployed in video games but can also be used in agent based models. Unity also allows for the development of 3D environments with a physics engine. The model will be presented using this platform.

3 Results

The application of the sugarscape model will be made up of several C# (programming language) scripts. These are;

- SugarCollector - agent framework.
- SugarArea - a script that randomly distributes sugar in the environment.
- SugarLogic - a script that describes how often sugar is distributed and where it is distributed.
- TFModelBrain - the Q-learning algorithm applied to each agent.

The TFModelBrain script is trained over 500 - 1000 iterations as this is the limit of computing power provided but can be increased if a more powerful computer is used. Once the training process is finished, the model can be executed and the behaviour of each agent is traced.

In Unity, to trace behaviour of individual agents, one can deploy "gizmos". This facility contains various widgets that can be applied to agents and the environment. The widgets used to trace behaviour in the model are;

- BoxCollider - sensor that detects how close an agent is to a physical object.
- Camera - used to allow agents to see.
- RayPerceptionSensor - a custom sensor that allows agents to compete with one another by shooting lasers at each other to disable one another for a given time period.
- AnimationTrack - used to track the movement of each agent during the simulation.

The behaviours that are of interest in this domain are;

- Do sugar collectors evolve overtime to hide behind obstacles from the snatcher (hostile agent)?
- How will the sugar collectors behave when the snatcher chases them?

To trace the behaviour on an individual level, first person cameras can be used to see what agents are doing (refer to Figure 1).

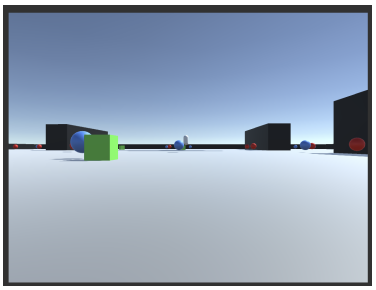

Fig. 1. First person view from agents perspective

4 Discussion

The motivation for this work stems from the lack of research carried out to address emergent human behaviours in agent based models. If the ABM community is to simulate human behaviour in models then it is necessary to test those methods that have already been developed by researchers to see if they really can be applied to much larger domains such as the simulation of people in cities, planning problems that require complex decision-making and so on. Sugarscape is a simple model, and it contains environmental features that can be used to test behaviours of agents. This research should hopefully provide agent-based modellers with new avenues to explore regarding intelligence representation and how deep learning can be a viable option when simulating human behaviours.

References

1. Bigbee, A., Cioffi-Revilla, C., Luke, S.: Replication of Sugarscape Using MASON. In: Agent-Based Approaches in Economic and Social Complex Systems IV (2007). https://doi.org/10.1007/978-4-431-71307-4_20
2. Epstein, J.M., Axtell, R.: Artificial societies and generative social science. Artificial Life and Robotics **1**(1), 33–34 (3 1997). https://doi.org/10.1007/bf02471109
3. Fukuyama, F., Epstein, J.M., Axtell, R.: Growing Artificial Societies: Social Science from the Bottom Up. Foreign Affairs (1997). https://doi.org/10.2307/20048043
4. Lysenko, M., Rahmani, K.: Sugarscape on Steroids : Simulating over a Million Agents at Interactive Rates. Proceedings of Agent2007 conference (2007)
5. Nair, R., Tambe, M.: Hybrid BDI-POMDP framework for multi-agent teaming. Journal of Artificial Intelligence Research (2005). https://doi.org/10.1613/jair.1549
6. OpenAI: OpenAI (2019), https://openai.com/
7. Rahimiyan, M., Mashhadi, H.R.: An adaptive Q-Learning algorithm developed foragent-based computational modeling of electricity market. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews (2010). https://doi.org/10.1109/TSMCC.2010.2044174
8. Rahman, A., Setayeshi, S., Zafarghandi, M.S.: An analysis to wealth distribution based on sugarscape model In an artificial society. International Journal of Engineering, Transactions B: Applications (2007)
9. Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. In: Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95) (1995)
10. Tellidou, A., Bakirtzis, A.: A Q-learning agent-based model for the analysis of the power market dynamics **2006**, 228–233 (11 2006)
11. Urban, C., Schmidt, B.: PECS – Agent-Based Modelling of Human Behaviour. Operations Research (2001)
12. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8**(3-4), 279–292 (5 1992). https://doi.org/10.1007/bf00992698
13. Watkins, C.J., Dayan, P.: Technical Note: Q-Learning. Machine Learning (1992). https://doi.org/10.1023/A:1022676722315
14. Zhang, Y., Bhattacharyya, S.: Effectiveness of Q-learning as a tool for calibrating agent-based supply network models. Enterprise Information Systems (2007). https://doi.org/10.1080/17517570701275390