Artificial Intelligence for Playing Catan with Explanation
Project Proposal

Student: David Martin
Supervisor: Dr Lars Kunze

Catan is a modern European board game by Klaus Terber for between 3 and 4 players. It is both very popular, having sold over 18 millions copies (Raphel 2014), as well as being critically acclaimed and winning a number of awards (BoardGameGeek).

The game involves spending resources and trading in order to develop a set of tiles so that you gain victory points. When a player gains 10 of these they win the game. There is a random element to that game in that the resources that are produced each turn are decided by a dice roll. Victory points are gained by building settlements and cities, playing development cards or possessing the longest road. A more comprehensive version of the rules is available on the official website for the game *(http://www.catan.com/service/game-rules)*, which comprehensively covers the more involved elements of the game such as the starting sequence, board layout and trading rules.

From a research point of view Catan is interesting due to its difference from games more traditionally studied for AI purposes such as Chess and Go. There are multiple reasons for this degree of difference. One such reason is that there is an element of randomness due to the dice roles. Although this reason differentiates it from the games such as chess the problem of dealing with probability has been fairly well covered with algorithms such as expectiminimax being designed to take this particular factor into account (Melkó and Nagy 2007). Another way in which Catan presents an interesting problem is the nature of the moves that can be taken by players. In games such as chess, a move involves the lone action of moving a single piece, in Catan a players turn can include building structures, using development cards and trading. This contributes to a reasonably high branching factor given the wide array of possible moves available. Naturally this branching factor fluctuates based upon the available options dictated by the resources a player possesses but once taking into account the possible actions of all opposing players and the factor of trades it becomes, on average, relatively large. The trading in itself is another interesting feature that must be considered in an AI. Trading is an integral part of the game, a player that trades effectively has a much greater chance of getting the resources they need and avoiding the penalties the robber can inflict. This aspect of temporary collaboration with other players adds another factor that must be considered when playing the game. The final element that serves to complicate the AI somewhat is the presence of uncertain information that is introduced to the game when someone gets a card robbed from them. When a player is robbed they lose a card at random with the only the two participating players knowing which. Therefore a small amount of reasoning over uncertainty will be necessary.

For the above reason an AI to play Catan will be a worthwhile project to embark on. The ultimate objective of the project will be to develop an AI that can play the game competently. In addition to this a further objective will be to provide the AI the ability to explain the moves it has just made, allowing it to explain why it has made a move. This would allow the system to act like a tutor, not only possessing the ability to play the game but also being able to suggest moves alongside an explanation.

In order to carry out this task I have identified a number of tools and approaches that could be utilised in order to facilitate its successful completion. The task of implementing the game is not within the scope of this project. Therefore it makes sense to use a 3rd party program in order to do this. The specific software that will allow this is 'JSettlers' an opensource Java implementation of the game by Jeremy Monin that runs as a server and can be communicated to with sockets (Monin 2016); effectively making communication with it language independent. This particular software is often used for research and projects involving Catan. This is beneficial as not only does the game come packaged with "bots" that could be used for testing but other AI implementations exist using the same API giving the opportunity for benchmarking against other attempts at AI for the game.

As it is the programming language that I am most familiar with, I will program the majority of the game in Java. Other languages may be required for other parts of the program and these will be considered when the need for them becomes more apparent. Although this project will lean more towards the research side of things I will still aim to follow good principles of software engineering by documenting code thoroughly and aiming to develop the project in a modular and extensible manner such that it may be of use to future projects in this particular area.

Being able to automatically explain moves in Catan is not something, as far as my research shows, that has been attempted before. When this type of task is normally undertaken it falls under the categorisation of explanation generation, many examples of which are for expert systems where a system is desired to give an explanation of, for example, a complicated medical diagnosis in natural language. When identifying the expectations for the Catan system it becomes clear that this level of detail and dexterity in explanation generation will not be necessary. A common theme across systems that generate explanations and rationalise is the usage of a high level logical representation of the knowledge at hand. One way of utilising this approach would be via the usage of Answer Set Programming (ASP) in order to generate plans (Lifschitz 2002) for playing. In ASP the problem is represented using a logic program, the answer sets of which correspond to solutions which can be obtained using a solver. Colaco and Sridharan (2015) propose an architecture where the high level decision making and planning is undertaken using Answer Set Prolog to generate a plan. The execution of the actions in accordance to the plan is then undertaken using a probabilistic technique based upon the sensor inputs of the robots. A further paper by Sridharan et al. (2016) expands upon this architecture by introducing KRASP and UMBRA as explanation generators for a robotic system. This approach and its principals could be adapted to provide a game playing agent the ability to generate explanations over its actions and the subsequent state of the game.

In terms of game playing AI Catan can be considered to be example of a multi-player game with imperfect information and a non-deterministic element. As mentioned earlier this makes some traditional AI techniques that would normally be applied for game playing, such as minimax, extremely hard to utilize (Pfeiffer 2004), especially when combined with the high branching factor this game can present. An interesting technique for playing Catan is proposed by Szita et al. (2009) who use a Monte-Carlo Tree Search (MCTS). MCTS is a somewhat fashionable technique in terms of game playing at present, with Google's AlphaGo utilising it in conjunction with deep learning techniques in neural networks to achieve very impressive results (Silver et al. 2016). Monte-Carlo techniques, which involve random simulations, have also been used to play other games such as Scrabble (Sheppard 2002). Further techniques for playing Catan also exist. The AI's that are included with the JSettlers software are heuristically driven and rule based, aiming to imitate a human thought process and play the game according to a set of dictated rules. This approach generally results in AI that is capable of playing rationally and presenting a good but often lacks flexibility and the ability to adapt to its opponents. Programming an AI using this technique is considerably simpler than an approach such as MCTS but also relies on the designer being especially good at the game in order to impart this expertise into the system, a stark contrast with MCTS where no expert knowledge of the game at hand is required (Chaslot et al. 2008). Other machine learning techniques may also be worth exploring for the game such as Q-Learning developed by Watkins and Dayan (1992). This is a reinforcement learning algorithm that works by giving a score for the value of transitioning between different states based upon the score all future predicted steps from that state. This algorithm has also been effectively used as the basis of effective game playing applications (Mnih et al. 2015)

An interesting approach to an AI for this game would therefore be to take inspiration from the architecture proposed by Colaco and Sridharan (2015). This would involve formulating the state of the game using ASP and then utilising this to determine a high level action or goal. Outputs of this stage could be converted into explanations or instructions with possible examples when realised in natural language being *Build more roads to get the longest road* or *Build a city in a port location*. These high level commands represent the objective of the system being able to explain moves and act as a tutor. This output can then be used to influence a more general AI technique in order realise a set of moves within the game to work towards this objective. This particular technique could, for example, utilise an MCTS, rule driven or Q Learning technique. MCTS appears to be particularly worthy of further consideration given the results that Szita et al. (2009) achieved and the possibility of using the output of the ASP derived strategy to act as heuristics to influence the weightings of the search in order to favour the choice made in planning.

These most obviously apparent method of evaluation would be to play the developed system against pre-existing AI's and to measure its effectiveness based on the number of wins it achieves and the margin of victory and defeat it achieves. An interesting theme that could be explored when evaluating the project would be to explore whether the inclusion of the planning and explanation aspect of the project improves the results over using the classical AI part on its own. As well as evaluating the system via an analysis of its performance against other agents it would also be beneficial to play against it manually to provide a qualitative view of the style of play that the developed system possesses.

As the project may not go fully to plan it is important to maintain a contingency plan. The most obvious act of contingency in order to reduce the scope of the project and pull it back within a more achievable zone would be to scale down either the AI or explanation aspect of the project. The choice of which of these to reduce would be taken pragmatically, based upon which is the most promising at the time. I predict that the hardest part of the project will be applying the explanation to the AI system, though this will obviously depend on the techniques and overall architecture chosen for the system. On the other hand there may time to expand the scope of the project. If this is the case then it would be possible to expand the rationalisation so that it becomes more general, being able to not only explain its own moves but any given to it, furthering its "tutoring" ability and allowing it to make judgements on moves from other sources. The likelihood of reaching this expansion phase must realistically be seen as slim however, given the ambitious scope of this project.

Feasibility of this project should not be an issue due to the low requirements. The only things that are required are a Java development environment which can easily be obtained for no cost. The API that I am using in order to provide the game engine is open source and can therefore be freely downloaded and, if necessary, modified. No special hardware should be required for this development with the only possible exception to this being the amount of RAM that may be required for the AI aspect of the project, though it should not be an issue with the systems I have at my disposal.

In order to stay on track for this project it will be necessary to devise a timetable in order to schedule the work that needs doing. There will be time over the Christmas break and before the end of the autumn term to carry out some work. This time will be best spent setting up the development environment and ensuring that communication with the 3rd party API is well understood ready for interfacing with the AI. It would also be beneficial to spend this time carrying out further reading to gather information regarding the various topics related to this project. A specific timetable for the project will be developed towards the end of the Christmas break when a clearer picture of the work is gathered. It is important in this timetable to factor in time for evaluating the project and writing the report, which will be an ongoing task throughout the project. I will also ideally set up weekly meetings with my supervisor, Dr Kunze, so that progress on the project and ideas can be discussed. The mandatory weekly progress logs will also be submitted.

To conclude, it can be seen that the presented project is suitably ambitious and provides ample opportunity to create an innovative and original piece of work. This project proposal hopefully demonstrates that the key issues and considerations necessary for this task have been addressed and I believe that, with support, I will be able to produce excellent results.

References

BoardGameGeek (2016). *Catan*. URL: `https://boardgamegeek.com/boardgame/13/catan` (visited on 11/08/2016).

Chaslot, Guillaume et al. (2008). "Monte-Carlo Tree Search: A New Framework for Game AI." In: *AIIDE*.

Colaco, Zenon and Mohan Sridharan (2015). "What Happened and Why? A Mixed Architecture for Planning and Explanation Generation in Robotics". In: *Australasian Conference on Robotics and Automation*. Citeseer, pp. 2–4.

Lifschitz, Vladimir (2002). "Answer set programming and plan generation". In: *Artificial Intelligence* 138.1, pp. 39–54.

Melkó, Ervin and Benedek Nagy (2007). "Optimal strategy in games with chance nodes." In: *Acta Cybern.* 18.2, pp. 171–192.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518.7540, pp. 529–533.

Monin, Jeremy (2016). *JSettlers2*. URL: `http://nand.net/jsettlers/`.

Pfeiffer, Michael (2004). "Reinforcement learning of strategies for Settlers of Catan". In: *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*.

Raphel, Adrienne (2014). "The Man Who Built Catan". In: *The New Yorker*. URL: `http://www.newyorker.com/business/currency/the-man-who-built-catan` (visited on 11/08/2016).

Sheppard, Brian (2002). "World-championship-caliber Scrabble". In: *Artificial Intelligence* 134.1, pp. 241–275.

Silver, David et al. (2016). "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587, pp. 484–489.

Sridharan, Mohan, Ben Meadows, and Zenon Colaco (2016). "A tale of many explanations: towards an explanation generation system for robots". In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, pp. 260–267.

Szita, István, Guillaume Chaslot, and Pieter Spronck (2009). "Monte-carlo tree search in settlers of catan". In: *Advances in Computer Games*. Springer, pp. 21–32.

Watkins, Christopher JCH and Peter Dayan (1992). "Q-learning". In: *Machine learning* 8.3-4, pp. 279–292.