CS 427: Term Paper

AI Strategy Selection in Real-Time Strategy Games With Emphasis on Starcraft II

Jacob Hurst, Richard Tran

April 7, 2019

1 Introduction

Real-Time Strategy (RTS) is a subset of strategy games in which the game is not turn-based but rather the participants control their resources and/or units in real time. In such a game, the players may play against time and/or diminishing resources in order to maximize their score and achieve a winning state. Given the complexity of RTS games, it is often very difficult to enumerate the state space of the game. The fast pace and abundance of tactical considerations in RTS games serve to further increase this complexity. This added complexity necessitates a shift in approach for artificial intelligence (AI) strategy selection and a rethinking of how to apply goal or data driven searches to the complex state space. This report will summarize an approach for applying goal-driven autonomy to the RTS game, StarCraft II, and summarize the common challenges and strategies unique to RTS games, specifically, StarCraft II.

1.1 StarCraft II

StarCraft II is a combat-based RTS game published by Blizzard Entertainment. The game revolves around three species (player types) each with their own unique abilities with respect to units and structures. There are three important resources to acquire which further the abilities of units and structures. To achieve a win, a player must maximize their units, resources, and structures and minimize the opposing players units, resources, and structures. This often results in a forfeit from the opposing player as they are left with no selection of moves.

2 Applying Goal-Driven Autonomy to StarCraft II

In RTS games, it is crucial for AI to react intelligently to unanticipated game states as players may unpredictably change their strategies and tactics as the game develops. In their 2014 report, UC Santa Cruz researchers, Mateas, M. et al., present an approach to developing such an AI agent, advocate the uses of AI agents that reason about their goals in response to unanticipated events, and present an implementation of the Goal-Driven Autonomy (GDA) conceptual model with a demonstrated application to StarCraft II.

2.1 Contributions

GDA models seek to address the problem of developing agents that respond to unanticipated failures and opportunities during plan execution in complex and dynamic environments. As the authors emphasize, "One of the main focuses of GDA is to develop agents that reason about their goals when failures occur, enabling them to react and adapt to unforeseen situations"[1]. This complexity is further increased when considering developing systems capable of concurrently reasoning on multiple goals. The authors present an implementation of a GDA model which allows a decoupling of the goal selection and goal execution logic in such an agent. This GDA model is accomplished by developing agents that reason about their goals.

The authors present a simplified version of the model. An important feature of the simplified model is the output from the planning component. The planning feature in the GDA system generates a set of actions, each action has a set of expectations for the expected

game state afterwards. If a failure occurs, this will be detected by means of detecting a difference in the current game state and the expected game state. This provides a mechanism for the agent to react to unanticipated events. This approach allows for autonomy to be integrated into the game playing agent.

2.2 Methods

The GDA model utilizes two common approaches in game playing AI; reactive systems and planning. Reactive systems do not look-ahead, rather, they map game states to actions. Reactive systems are commonly implemented using finite state machines and behavior trees. As reactive systems lack the look-ahead component, they are not able to hold expectations and as a result, detect failures. This limitation is overcome by using a blackboard to model an agents mental state. This enables the agent to reason about an expected game state. Planning is commonly implemented in a goal-oriented action based manner. In this manner, the agent has a set of goals, each of which map to a set of trigger conditions. When the trigger conditions for a goal become true, the system may begin planning for the activated goal. As a result, this form of planning maps game states to goals rather than actions.

The GDA model presented consists of 5 components; goal manager, planner, discrepancy detector, explanation generator, and goal formulator. At the start of a game, the goal manager provides the agent with an initial goal which is then passed to the planner. The planner generates a set of actions and expectations for achieving that goal. As these actions are executed in the game, the discrepancy detector monitors for any differences in the current game state and the expected game state. If a discrepancy is detected, the agent passes that discrepancy to the explanation generator. The explanation generator builds an explanation on why the failure occurred and passes that explanation to the goal formulator. The goal formulator evaluates the explanation and formulates a response goal. This goal is then passed to the goal manager. The goal manager selects which goal to execute and the procedure then repeats for the remainder of the game.

Using the presented GDA model, the researchers developed EISBot, an agent capable of playing StarCraft II effectively.

2.3 Results

In utilizing the presented GDA model, the researchers were able to separate the goal selection from the goal execution in their agent. Their agent, EISBot, was evaluated by applying it to the task of playing complete games in StarCraft II. The authors detail the results against StarCraft IIs built-in AI, stating "Overall, the agent achieved a win rate of 73% against the built-in AI. To ensure that a variety of scripts were executed by the built-in AI, 20 games were run on each map for each opponent race. In total, 300 games were run against the built-in AI." [1]. The agent also played 100 games in a competitive tournament and outranked 48% of the competitive players. The agent had a win rate of 37% against human competitors.

3 A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft II

RTS games involve many dynamic events that affect the game state, some of which are unknown to the player due to the incomplete and imperfect information presented to them

by the game. This, of course, leads to significant challenges for AI. In 2013, Ontañón et al. published their research on IEEE on the various challenges AI face with RTS games and describes the architectures of some created AI bots.

3.1 Contributions

The authors reflect that the research in RTS AI can further benefit into the real world. "Road traffic, finance, or weather forecasts are examples of such large, complex, real-life dynamic environments. RTS games can be seen as a simplification of one such real-life environment, with simpler dynamics in a finite and smaller world, although still complex enough to study some of the key interesting problems, such as decision making under uncertainty or real-time adversarial planning. Finding efficient techniques for tackling these problems on RTS games can thus benefit other AI disciplines and application domains, and also have concrete and direct applications in the ever growing industry of video games"[2].

The authors enumerated the common challenges associated with AI in RTS. A summary of StarCraft AI competitions was also included that described AI architectures used by the participants. Altogether, the authors provided a centralized overview of the research being done in the area of RTS game AI.

3.2 Challenges in StarCraft AI

There are a significant amount of challenges in the domain of RTS AI. The authors have generalized them into the following six areas:

1. **Planning**

   Due to the nature of RTS games the state space will be much larger than traditional board games such as Checkers or Chess. The number of actions on a single "piece" is also much larger than their board game counterparts. Thus, common approaches to mapping out the state space such as search trees are not suitable.

2. **Learning**

   Given that the state space is so large and search trees not being applicable, learning techniques seems to be next alternative. However, that comes with its own challenges that the authors categorized into "Prior learning, In-game learning, and Intergame learning"[2].

   Prior learning would be the utilization of available data that exists before a game is played such as replays of prior matches or information about specific strategies that are dependent on the terrain of the map. In-game learning is in regards to how an AI would be able to improve their own strategies and game play while playing their game. The difficulty in this is once again attributed to the nature of RTS: the state space is large and there is imperfect information. Intergame learning refers to how an AI can learn from their strategy in order to improve their chances of winning the next.

3. **Uncertainty**

   The authors state that there are two types of uncertainty in RTS games. The first of which would be the fact that the game contains imperfect information, meaning that the players cannot view the whole game map. In order to update or gain new information

the player must send a unit for reconnaissance in order to see what the opponent is up to. This in itself is a challenge as the AI needs to utilize previous information on where the opponent's army is positioned so that they can maneuver their scout to avoid it while trying to find information from key areas. The second type of uncertainty arises from the fact that the player cannot predict the actions of their opponent.

4. **Spacial and Temporal Reasoning**

   In StarCraft, many maps are small which makes terrain exploitation is a worthwhile consideration in any strategy. Spacial reasoning is used in building placement, which is building one's structures in such a way that it protects themselves against large scale attacks as well as efficient building placement such that it does not interfere with their own unit pathing.

   Temporal reasoning is an important aspect in engaging the opponent, such as attacking before the opponent finishes a critical upgrade or defending yourself at key times when your own upgrade has not yet completed.

5. **Domain Knowledge Exploitation**

   In regards to traditional board games, the authors state, "In traditional board games such as Chess, researchers have exploited the large amounts of existing domain knowledge to create good evaluation functions to be used by alpha–beta search algorithms, extensive opening books, or endgame tables"[2]. However it remains unclear how significantly large amounts of domain knowledge can be used by the bots to improve their chances of winning.

6. **Task Decomposition**

   Taking a specific goal and breaking it down to smaller executable steps. This arises some complexity since the smaller steps may change depending on the situation. The higher-level decision being overall strategy, which is then broken down to tactics, which is dependent on the position of the units at that moment. Reactive control is also key to the chosen tactic as some tactics may not be used in areas such as a choke point or an open field.

3.3 Bot AI for Competitive StarCraft

The researchers attended the most popular and longest running StarCraft AI competition called the AIIDE StarCraft Competition, where participants submit their AI bot to compete against others for prize money funded by Blizzard Entertainment, the creators of StarCraft. At this event they interviewed and received information about the architecture of the top seven placing bots. The architecture of these seven bots were analyzed and the researchers were able to place these bots into two paradigms of design: Abstraction and Divide & Conquer.

The abstraction approach involves each module of a bot to have a desired goal, and decisions are made in which to reach this goal is communicated upwards from lower tiered modules whose goal is to provide a decision. This approach allows each module to focus on its own goals which overall benefits the entire system thus allowing the game of StarCraft to become simpler to understand. The divide & conquer design revolves around having one module that focuses on each task specifically such as separate modules for resource gathering, attacking, building, etc. In this way, many actions can be done at simultaneously.

Out of the seven chosen AI bots, five followed the Divide & Conquer design with one of them being a hybrid of both paradigms. If we looked at their architectures a more closely, it is specifically the combat aspect of the game that Divide & Conquer becomes the more popular choice. This seems to make sense as combat relies on many maneuvers of separate army groups in various locations doing multiple actions simultaneously. Creating an AI that learns from domain knowledge is still a difficult task and all bots instead have a list of hard-coded strategies in which they simply choose from. While this approach has a higher win rate in competition, the downsides reveal what needs to be worked on if we would like this research to be applied in real world applications. As the authors state, "These hard-coded approaches struggle to encode dynamic, adaptive behaviors, and are easily exploitable by adaptive opponents"[2].

4 Conclusion

For AI researchers, RTS games is a very challenging undertaking. However, it is progress towards other fields of AI research such as finding efficient actions in a factory or finding the optimum path a robot can take to a destination. We can hope that one day that the research put into these competitive game bots will go on to benefit our real world society in the future.

References

[1] Mateas M. Weber, G. and A. Jhala. Applying Goal-Driven Autonomy to StarCraft. *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2014 (accessed April 2, 2019).

[2] Synnaeve G. Uriarte A. Richoux F. Churchill D. Ontanon, S. and M. Preuss. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games*, 2013 (accessed April 2, 2019).