Towards case-based reasoning in real-time strategy environments with SEASALT

Jakob M. Schoenborn^{1,2} and Klaus-Dieter Althoff^{1,2}

 ¹ University of Hildesheim Universitätsplatz 1, 31141 Hildesheim, Germany schoenb@uni-hildesheim.de
² German Research Center for Artificial Intelligence (DFKI) Trippstadter Str. 122, 67663 Kaiserslautern, Germany kalthoff@dfki.uni-kl.de

Abstract. Real-time situations provide numerous different problems to solve. Starting with the requirement of finding a solution inside an acceptable time frame, the problem is to find the right balance between performance and precision of the system. One might not be able to wait multiple minutes for a solution, however, a too quickly made decision might entail a certain risk factor. Thus, it has to be decided when methods such as using a rule-based system is sufficient and when it is rather beneficial to take the cost of using methodologies for knowledge management. We are using StarCraft II as an example for decision making in a real-time environment using incomplete information with a finite set of buildings and units to control. We propose using agents to decide the proportion of command authority between using a rule-based and a case-based reasoning agent. Earlier stages of the game seem to be promising for immediate reactions while later stages of the game require more planning due to the increased rate of information, which have to be processed. By reusing past experiences, case-based reasoning may be able to help improving the planning process.

Keywords: Case-based Reasoning \cdot Realtime Strategy \cdot Knowledge Management

1 Introduction

To solve problems in a real-time situation is very difficult, especially with incomplete information. The general problem consists in finding a selection of complex processes in order to decrease the idle time of any given unit as much as possible. This is not only applicable in the gaming area, but similar problems can be found in the production area: using limited resources to obtain the highest possible output. The longer it takes to process the information, the more likely it is to lose value of the made decision, due to the delayed provision of the solution

Copyright © 2020 by the paper's authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



Fig. 1. Screenshot of StarCraft II during early stages of the game: Building the base, train combat units and manage resources. The panel in the bottom contains multiple information based in the selected unit such as health points, attack damage, unit-specific abilities, and a minimap with three different levels of fog of war (visible, visited but not currently visible (gray), and never visited (black).

or since the data as a decision foundation has been changed during the process - or a combination of both. This requires the decision making processes to be as effective as possible.

StarCraft II is a real-time strategy game developed and maintained by Activision Blizzard, including hosting and organizing tournaments which since 2010 have awarded \$33,003,549.29 in total prize money from 5839 tournaments, with a majority of the price money granted to players located in south korea [3]. The game, representative for any other games in its genre, evolves around two different aspects and can be seen in Fig. 1: macro- and micromanagement. The former considers the usage of resources (minerals, blue crystals) to build structures and combat units, the latter considers moving units for scouting and fighting the enemy player. Combat contains multiple different aspects to consider, such as differences in unit weapon- and armor types, ground and air units, and additionally in the kind of movement itself, for example, using a "hit- and run"-strategy to deal as much damage as possible while taking as less damage as possible. With these differences, it is important to scout the enemy to know about the enemies chosen strategy to counter it by building corresponding units. These receive a damage bonus based on their weapon type against the armory types of the enemies unit. This has also been tested in a CBR approach by Cadena and Garrido [2].

The usage of reinforcement learning (RL) and artificial neural networks (ANN) is in current research the way to solve seemingly any problem, for example, Vinyals et al. provided 2019 an AI, which is capable of defeating professional StarCraft II players [9] or earlier approaches by Mnih et al. using a similar approach [6]. However, ANN usually implies the unfair advantage of numerous training sessions, i. e., parallelizing games, or taking larger datasets into account than a human could process (971.000 replays of games played by human players have been used as data set) [9]. Since one of the strengths of case-based reasoning (CBR) is to use it even with a smaller set of cases, we investigate the possibilities of implementing CBR into the decision making process.

To consider the possibilities, we overview recent work which has been done in the real-time strategy games area, followed by describing our ideas on microand macromanagament agents, and ending with a conclusion including future work.

2 Related Work

Vinyals et al. developed AlphaStar, an AI which combines ANN and RL especially during training [9]. The API on which the AI has been created contains an observation object, which provides necessary information such as visible units. buildings, and the environment in general. Using these observations and the abilities of the owned units, the action state space can be transmitted via the monitoring layer. The monitoring layer processes the received observations by an artificial delay of 80ms and limiting the number of taken actions to approx. 22 per 5 seconds. This decision has been made to prevent the AI to gain an unfair advantage in contrast to the human player who is limited in the number of physical actions per second. The professional player Dario 'TLO' Wünsch credits AlphaStar to not be "superhuman", resulting in an overall fair feeling when playing against the AI [9]. In terms of learning components, the combination of RL and supervised learning (SL), multiple instances of RL agents are spawned by the SL layer, collect experiences, update the policy and value outputs. As a baseline, replays of human games have been used to learn from their behaviours and strategies and apply it to the current player.

Wender and Watson presented 2014 an approach to combine CBR with RL specifically for the micromanagement problem [10, 11]. Two kinds of agents are presented. One agent is observing the overall state of the game by creating so-called influence maps. These are areas, in which the system can take influence by executing actions such as attacking, building, or scouting to further increase the influence. Any other agent represents one unit object of the game, such as a marine soldier selected in Fig. 1. The casebase contains cases, which describe, for example, actions based on the current influence, such as sending certain agents to specific areas to increase the influence. These cases are not changed during the execution of the program. However, using Q-learning in the RL component, the solution of the most similar case may be adjusted to fit into the current situation.

Based on the perceived result after execution, the agent will be rewarded or punished [10, 11].

3 On the granularity level of control

The goal is to defeat the enemy by destroying every building and unit. Depending on the experience of the opponent, especially against beginners, a rule-based agent using only a set of few rules, such as building as soon as possible and collectively attacking after x units have been trained, is sufficient to fulfill this task. There are a few rules of thumb, which generally hold true and can easily be followed by any rule-based agent, such as not being supply blocked (meaning certain buildings have to be built before recruiting more units), using excess resources to build further production buildings for faster recruiting, and gathering combat units before heading towards the enemies base. This does not take the complexity of the game into account, which has been slightly mentioned in the introduction.

For the agent framework, we use the SEASALT of Kerstin Bach, which consists of multiple different layers dealing with knowledge presentation, -provision, -representation, -formalization and knowledge sources [1]. Fig. 2 shows the complete architecture layout. As a brief overview, the "Shared Experience using a Agent based System Architecture LayouT" uses one coordination agent who controls n topic agents. Each of those agents, including the coordination agent, contain a case factory, which in turn contains multiple agents for knowledge maintenance and formalization. A collector agent gathers information from a community of experts, for example, by using crawling technologies and extracting textual information into knowledge representations. Other kinds of knowledge representations are ontologies, taxonomies, similarity measures, constraints, vocabularies, and rules. These can be accessed by any layer, while knowledge formalization and knowledge sources may also modify these.

For our application, we propose to instantiate a coordination agent, macroand micro agent and an explanation agent. The coordination agent inhabits templates and a question handler in a knowledge map. One coordination agent can control other agents, which contain an own case factory and casebase. The casebase of these agents (including the coordination agent) are specially targeted for their individual needs, especially in terms of defined similarity measurements for the retrieval. For example, the micro agent might value the unit count for tactical combat decision making higher than the macro agents for prioritizing the reproduction of fallen units. To account for the complexity of the game and to inhabit a learning component, the following models are possible:

Centralized CBR. By only using one methodology, there is no interference and communication overhead between multiple agents to be expected, resulting in an overall faster decision making process. However, it seems questionable whether it is feasible to let each agent query the CBR system - and as such the casebase - based on numerous triggering events. This may create the necessity to repeatedly



Fig. 2. SEASALT, a domain independent architecture for knowledge management using multiple agents

evaluate the current plan, especially in very information heavy situations such as during fights between multiple armies.

Distributed problem solving. For distributed problem solving, a coordinator agent can be used as a first-level support to handle increasing complexity limits, such as during combat fights. The coordinator agent functions as the centralized CBR system, which has been partially covered by Wender and Watson [10]. By combining the information gathered by the micro- and macro agent, the coordinator agent combines these with the general information of the observation object (such as resources and visible areas) to create a new plan. A plan may consist out of multiple sequences, analogous to the approach of Kolbe et al. used in first-person shooter gaming scenarios [5].



Fig. 3. Instantiation of distributed problem solving using one macro agent, one micro agent, and one coordination agent.

CBR and *ANN* in combination. As discussed above, Vinyals et al. used ANN in combination with RL to train the agents with new strategies and planning capabilities [9]. For providing explanations for an ANN system, Keane and Kenny defined ANN-CBR twins. The combination of ANN and a CBR technique, which mostly used k-NN in their case, provided better results than considering each method separately [4]. The approach could be used analogously: CBR could provide most similar cases to the current situations while ANN interprets these and takes control over the game state and the overall game plan.

In Fig. 4, an explanation agent has been added to the knowledge provision layer. This agent will receive information of the macro- and micro agent directly whenever the agent queries those, in addition to the coordination agent. The coordination agent in turn may provide the solution to the graphical user interface. The explanation is targeted for the knowledge engineer to further understand why certain actions has been taken. This is helpful to understand the learning process of the micro- and macroagents and support further debugging of those, allowing a more targeted way for knowledge maintenance to increase the learning rate of the agents and to gain a more purposeful search through the casebases. Since it can be assumed that the knowledge engineer inhabits knowledge over the domain, providing a list with most similar cases and their features may serve as an explanation by itself - using the inherent explainability of CBR. Otherwise, for rather novice users, explanation patterns as introduced by R. Schank can be used for explanations [8]. These patterns can be filled, for example, with the rules that have been used, the applied similarity measures, the used vocabulary, or adaptation rules that have been triggered. These are the components of the knowledge containers as defined by Richter [7].



Fig. 4. Instantiation of distributed problem solving extended by adding an explanation agent, which gathers information from the macro- and micro agent and communicates with the coordination agent, transferring explanations to the graphical user interface.

Full multi-agent system. Each unit is treated as an own agent. However, these agents need to willingly work together as a team to defeat the opponent. Typically, in a multi-agent system, each agent may interact with any other agent as well as the coordinator agent who coordinates single agents to fulfill the overall goal (which may be split into separate lower level goals). The architecture is analogous to SEASALT in Fig. 2. From a knowledge management perspective, it may be interesting to evaluate on which granularity the level of control should be settled. As stated before, each unit could be treated as an own agent. A typical army usually consists out of 20-40 units, thus, leading to 20-40 agents in need to communicate and coordinate with each other and the coordination agent. Another possibility may be to designate one CBR agent to a specific unit type, since usually an army only consists out of 2-4 different unit types. This would decrease the level of communication needed drastically. However, experiences are then also limited to a single agent.

4 Conclusion

Developing an artificial intelligence in real-time strategy fields provide multiple challenges. However, there are also chances for increasing the efficiency by including CBR. Here, we shared a few thoughts on the granularity level of control by different approaches. With having positive and negative aspects, the question remains open whether there is an overall "best" approach or whether a hybrid architecture might be feasible as well. Regarding this question, another important question is the granularity of the case structure in terms of choosing attributes and how to model the local and global similarity without ending up in overfitting problems. Based on the case structure, the case instances themselves are another challenge on their own. For a given sequence of events, a starting point and an end point has to be defined. Especially in terms of learning from single fights, the start and the end of a fight has to be determined and saved inside the case structure. Furthermore, the transferability of knowledge learned by a single agent to another agent using the same (or parts of) casebase remains to be an open research question, which may help to structure the casebases of a single agent. These challenges may be considered in future work.

References

- Bach, K.: Knowledge Acquisition for Case-Based Reasoning Systems. Ph.D. thesis, University of Hildesheim (2013), http://www.dr.hut-verlag.de/978-3-8439-1357-7.html
- Cadena, P., Garrido, L.: Fuzzy case-based reasoning for managing strategic and tactical reasoning in starcraft. In: Batyrshin, I., Sidorov, G. (eds.) Advances in Artificial Intelligence. p. 113–124. Springer Berlin Heidelberg (2011)
- Earnings, E.: StarCraft II Top Players & Prize Pools Esports Tracker :: Esports Earnings (2020), https://www.esportsearnings.com/games/151-starcraft-ii, last validation: 06/14/2020
- 4. Keane, M.T., Kenny, E.M.: How case-based reasoning explains neural networks: A theoretical analysis of XAI using post-hoc explanation-by-example from a survey of ANN-CBR twin-systems. In: Bach, K., Marling, C. (eds.) Case-Based Reasoning Research and Development 27th International Conference, ICCBR 2019, Otzenhausen, Germany, September 8-12, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11680, pp. 155–171. Springer (2019)
- Kolbe, M., Reuss, P., Schoenborn, J.M., Althoff, K.D.: Conceptualization and implementation of a reinforcement learning approach using a case-based reasoning agent in a FPS scenario. In: LWDA 2019, Workshop on Knowledge Management, Berlin (2019)
- Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
- Richter, M.M.: Fallbasiertes Schließen. Görz, Günther; Rollinger, Claus-Rainer; Schneeberger, Josef (Hrsg.): Handbuch der Künstlichen Intelligenz 4, 407–430 (2003)
- 8. Schank, R.C.: Explanation Patterns: Understanding Mechanical and Creatively. L. Erlbaum Associates Inc., USA (1986)
- Vinyals, O., et al.: Grandmaster level in StarCraft II using multiagent reinforcement learning. Nature 575(7782), 350–354 (2019). https://doi.org/10.1038/s41586-019-1724-z
- Wender, S., Watson, I.: Combining case-based reasoning and reinforcement learning for unit navigation in real-time strategy game AI. In: Lamontagne, L., Plaza, E. (eds.) Case-Based Reasoning Research and Development. p. 511–525. Springer International Publishing (2014)
- Wender, S., Watson, I.: Integrating case-based reasoning with reinforcement learning for real-time strategy game micromanagement. In: Pham, D.N., Park, S.B. (eds.) PRICAI 2014: Trends in Artificial Intelligence. p. 64–76. Springer International Publishing (2014)