Case-Based Reasoning for Army Compositions in Real-Time Strategy Games

[1]*Martin ČERTICKÝ*, [2]*Michal ČERTICKÝ (4th year)*
Supervisor: [3]*Peter SINČÁK*

[1,3]Dept. of Cybernetics and Artificial Intelligence, FEI TU of Košice, Slovak Republic
[2]Dept. of Applied Informatics, FMPH Comenius University of Bratislava, Slovak Republic

[1]martin.certicky@tuke.sk, [2]certicky@fmph.uniba.sk, [3]peter.sincak@tuke.sk

***Abstract*—Over the years, there have been numerous successful applications of artificial intelligence techniques in the field of computer gaming. However, traditional graph-search based techniques often fail to perform at human level in real-time games with non-discrete game states. This fact encouraged the research of Case-Based Reasoning (CBR) and its applications to various aspects of computer game AI, especially in case of real-time strategies. We show how CBR can be used in the process of selecting the most effective army composition in a strategy game StarCraft, based on the game-related knowledge base designed by human experts (gamers).**

***Keywords*—Case-Based Reasoning, Agent, Real-Time Strategy, StarCraft**

I. Introduction

Players of adversarial computer games often need to adapt and react promptly and effectively to their opponent's unpredicted strategy. Most known artificial intelligence techniques have already been applied to computer games [11]. In this paper we will describe theoretical basics of an AI technique called Case-Based Reasoning (CBR) and show how it can be used to address one of the challenges from the field of game AI. CBR solves current, previously unseen, problems based on the solutions of similar past problems. The method includes the comparison process of current situation to previous similar cases. It chooses the most similar case from the past, and uses its corresponding remembered solution.

Different game genres present different challenges for artificial players (agents/bots). This work focuses on so-called Real-Time Strategy games (RTS). In RTS, as in other wargames, the participants position and maneuver units and structures under their control to secure areas of the map and/or destroy their opponent's assets. RTS games have shown to have huge decision spaces that cannot be dealt with search based AI techniques [11].

We will describe one possible application of CBR to playing RTS games, specifically for dynamic selection of the effective army composition in response to opponent's strategy. This is being done based on observation of opponent's actions and consequently comparing this knowledge with our case database. We have implemented an agent using CBR in this decision-making process within a real-time strategy game StarCraft: Brood War[1].

[1]StarCraft and StarCraft: Brood War are trademarks of Blizzard Entertainment, Inc. in the U.S. and/or other Countries.

The main goals of the paper is to present the practical application of CBR to an agent playing StarCraft and to demonstrate that this technique can help create intelligent, human-like behaviour in RTS games in general.

After the extensive overview of related work in section II, section III describes in detail how the CBR is used to solve our specific problem of selecting an optimal army composition. In section IV, we introduce our agent and give some insight into its implementation.

II. Related Work

Over the last years, case-based reasoning has grown from a rather specific and isolated research area to a field of widespread interest [1]. The number if its applications in various areas, including the game AI and opponent modelling, is rapidly growing.

Focusing only on the domain of game AI research and CBR applications relevant to this field, we were able to identify a considerable amount of published work. It therefore makes sense to categorize the games based on some kind of taxonomy, such as the one introduced by Aha, Molineaux and Ponsen in [2]. The games and corresponding CBR research is divided based on both the traditional player's viewpoint and on the degree of attracted research interest into following 7 categories:

A. Classic board games:

Typical board games like chess or checkers present a discrete, deterministic environment with two agents (players) affecting it in episodic turns. Several researchers have addressed classic board games, beginning with Arthur Samuels rote learning approach for playing checkers [15]. De Jong and Schultzs GINA instead memorized a partial game tree for playing Othello [4]. Chess has also been a popular topic. For example, Kerner described a method for learning to evaluate abstract patterns [10]. More recently, Powell et al.'s CHEBR learned to play checkers given only a paucity of domain knowledge [12].

B. Adventure games:

In adventure games, CBR has been used mainly for automated content generation. Fairclough and Cunningham described OPIATE [7], which uses a case-based planner and constraint satisfaction to provide moves for a story director agent so as to ensure that characters act according to a coherent plot. Also, Daz-Agudo et al. described a knowledge-intensive approach [5] that extracts constraints from a users interactively-provided specification, uses them to guide case retrieval and adaptation, and then creates a readable plot using natural language generation techniques.

C. Team sports:

Team sport games provide a challenging environment for the problems of real-time multi-agent coordination and planning, but do not involve complicating dimensions common to strategy games, such as economies, research, and warfare. Quite popular instance of such game is the RoboCup Soccer [14]. Wendler and Lenz described an approach for identifying where simulated agents should move [19], while Wendler et al. reported strategies for learning to pass [20]. Gabel and Veloso instead used a CBR approach to select members for a team [9].

D. Real-time individual games:

Real-time games with single agent, such as first-person shooters, leave little space for CBR application, but there have been a few applications anyway. For example, Fagan and Cunningham focused on a plan recognition task - they acquired cases (state-action planning sequences) for predicting the next action of a human player [6].

E. Real-time god/management games:

Single player management games require agents to mainly deal with planning and plan adaptation tasks in possibly non-deterministic environment. Fascianos MAYOR system [8] learns from planning failures in Sim City[2], a real-time city management game. MAYOR monitors planning expectations and employs a causal model to learn how to prevent failure repetitions, where the goal is to improve the ratio of successful plan executions.

F. Discrete/turn-based strategy:

Complex turn-based strategy games, like Freeciv (open-source Civilization clone), require players to solve a number of distinctive sub-tasks. An example of applying CBR to such sub-task is Ulam et al.'s approach to defending the cities from attackers [16].

G. Real-time strategy:

RTS games usually focus on military combat (versus one or more adversaries), although they also include decision dimensions concerning tasks such as exploration, economic development, or research advancement in a non-deterministic, partially observable environment. Aha et al. used a case-based system CaT [2] to select offensive and defensive actions, and Weber with Mateas [18] used CBR to select a build order in Wargus (open-source Warcraft 2 clone). Cadena and Garrido presented the combined approach using the Fuzzy sets and CBR to deal with strategic and tactical management in StarCraft [3].

[2]Sim City is a trademark of EA International Ltd.

III. Our Work

Generally, in RTS games, a player frequently faces a number of "typical situations". Experienced players know how to respond to these situations in optimal way thanks to knowledge acquired by playing the game a lot.

Our artificial agent needed to have some kind of knowledge representation structure, that would be able to hold such information, and allow an agent to use it during the gameplay.

An intuitive choice of method for representing and reasoning about "typical situations" is CBR, where every such situation is considered an individual case.

The classic definition of CBR was coined by Riesbeck and Schank [13]:

"A case-based reasoner solves problems by using or adapting solutions to old problems."

Conceptually CBR is commonly described by the CBR-cycle (Fig. 1). This cycle comprises four activities (the four REs):

1) Retrieve similar cases to the problem description.
2) Reuse a solution suggested by a similar case.
3) Revise or adapt that solution to better fit the new problem if necessary.
4) Retain the new solution once it has beed confirmed or validated [17].

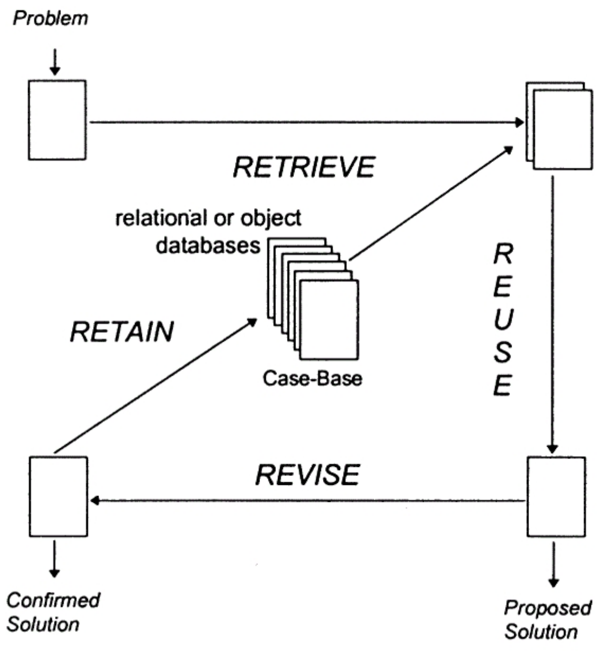


Fig. 1. Four activities of the CBR cycle as described in [17].

In our case, these problems (cases) are certain game situations. More specifically, a game situation is represented by a single *composition* of opponent's *army*.

Solutions to these problems are adequate counter-strategies, crafted by human players over time. A counter-strategy consists of our *desired army composition* and a collection of *desired upgrades* and *technologies*.

It is rare in RTS games that certain army composition has more than one equally good *"counter-composition"* (counter-strategy).

Generating such counter-compositions by some kind of rule-based system would be too difficult (near unrealizable in real time), because of the complexity of StarCraft. Within every composition, there exists a large number of synergic effects and the overall effectivity against other compositions depends on too many different factors.

CBR allows us to abstract from the low-level game mechanics and in-depth origins of the effectivity of individual compositions, and take advantage of long-term experience of human players.

We are able to make use of predisposed database of game situations which we created. So far, we have not experimented with automatic modification of this database by an agent.

IV. Implementation

To test this approach, we implemented an agent playing a real-time strategy game StarCraft (Fig. 3). The agent was programmed in Java, using a JNIWBAPI interface, that allowed us to access in-game information and issue order to our units in real time, during the course of the game.

Simply put, our agent was constantly producing certain types of units during the game. The choice of unit types to produce is a result of evaluating the current game situation and selecting the adequate (most similar) case.

Each *case* is a vector consisting of doubles $\langle ratio, unitType\rangle$, where $unitType$ is a specific kind of opponent's unit and $ratio$ enumerates its percentage in his army.

A *solution*, corresponding to a case, is a similar structure. It also contains a collection of $\langle ratio, unitType\rangle$, but $unitType$s here correspond to units that we want to have in *our own army*. Additionally, a solution can also contain several $upgrades$ and $technologies$ that we want to research.

The case-solution list (database) is stored in simple text file, where every line contains one case and a corresponding solution.

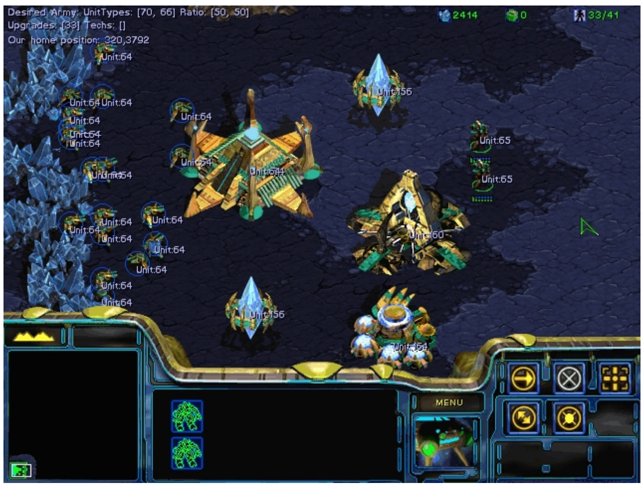
Few examples can be found in Fig. 2 (note that the original lines are divided, and some unit types are omitted to fit the paper format).

```
15;Medic,20;Firebat,65;Marine---
          40;Zealot,40;Dragoon,20;HighTemplar
          @LegEnhancements,PsionicStorm

60;SiegeTank,40;Vulture---
          50;Zealot,40;Dragoon,10;Arbiter
          @LegEnhancements,StasisField

20;Ultralisk,15;Defiler,65;Zergling---
          30;Archon,20;DarkArchon,50;Zealot
          @MindControl,LegEnhancements
```

Fig. 2. Example cases with corresponding solutions. Solutions are divided by "---" string.

Our agent uses a simple *similarity function* to determine which case resembles the current game situation the most. Specifically, similarity function $S(A,B)$ compares two cases (army compositions) A, B and returns the percentage of units that they have in common. Let n be a total number of unit types in the game and let $ratio(A,i)$ denote the percentage of $i-th$ unit type in the case A. Function $S(A,B)$ is then defined as:

$$S(A,B) = \sum_{i=0}^{n} MIN\big(ratio(A,i), ratio(B,i)\big)$$



```
Desired Army: UnitTypes: [70, 66] Ratio: [50, 50]
Upgrades: [33] Techs: []
Our home position: 320,3792
```

Fig. 3. In-game screenshot of our agent playing a StarCraft 1 vs. 1 match.

After we find the case that resembles current game situation the most, we use its solution as a parameter for our production function, which tells the agent what unit types (and in what ratio) to produce.

We evaluate game situations simply by observing the opponent's army composition. It is of course changing constantly during the game. Hence, in order to be successful, it is essential to properly monitor the game map (to "scout").

Consider the following example of army composition adaptation based on CBR: Thanks to scouting, we have noticed that the opponent started producing many air units (switched from *"100%zerglings"* to *"50%zerglings, 50%mutalisks"* army). Our own army (*"100%zealots"*) therefore becomes uneffective, since it cannot attack air units. However, using our similarity function, we determine that this situation resembles one of the typical cases from our database - the one with associated solution telling us to produce army effective against both air and ground enemy units (*"30%corsairs, 30%dragoons* and *40%zealots"*). When we tell our production function to produce this new composition, we should be able to deal with new opponent's army.

V. Conclusion & Future Work

Case-Based Reasoning offers a fine way to increase efficiency of intelligent agents playing strategy games. In future we are planning to create a system where our case database will be able to adapt by itself. This will be done based on game results and even on results of particular game sections (fights, economic situations etc.). We are also planning to include our CBR module in a student project *MontyBot*. The main goal of this project, solved by several students from different universities, is to create an intelligent StarCraft-playing agent using a wide variety of methods of artificial intelligence.

In our paper we described step-by-step the CBR technique, along with one of its possible application. We are certain this method has a great potential not only in this field.

References

[1] A. Aamodt, E. Plaza, *"Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches,"* AI Communications 7, 1994, pp. 39-59.

[2] D. W. Aha, M. Molineaux, M. Ponsen, *"Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game,"* Case-Based Reasoning Research and Development, Lecture Notes in Computer Science Volume 3620, 2005, pp. 5-20.

[3] P. Cadena, L. Garrido, *"Fuzzy Case-Based Reasoning for Managing Strategic and Tactical Reasoning in StarCraft,"* Advances in Artificial Intelligence, Lecture Notes in Computer Science Volume 7094, 2011, pp. 113-124.

[4] K. De Jong, A. C. Schultz, *"Using experience-based learning in game playing,"* Proceedings of the Fifth International Conference on Machine Learning, 1988, pp. 284-290.

[5] B. Daz-Agudo,P. Gervs, F. Peinado, *"A case based reasoning approach to story plot generation,"* Proceedings of the Seventh European Conference on Case-Based Reasoning, 2004, pp. 142-156.

[6] M. Fagan, P. Cunningham, *"Case-based plan recognition in computer games,"* Proceedings of the Fifth International Conference on Case-Based Reasoning, 2003, pp. 161-170.

[7] C. R. Fairclough, P. Cunningham, *"AI structuralist storytelling in computer games,"* Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, 2004.

[8] M. J. Fasciano, *"Everyday-world plan use,"* Technical Report TR-96-07, The University of Chicago, Computer Science Department, 1996.

[9] T. Gabel, M. Veloso, *"Selecting heterogeneous team players by case-based reasoning: A case study in robotic soccer simulation,"* Technical Report CMU-CS-01-165, Pittsburgh, PA: Carnegie Mellon University, School of Computer Science, 2001.

[10] Y. Kerner, *"Learning strategies for explanation patterns: Basic game patterns with applications to chess,"* Proceedings of the First International Conference on Case-Based Reasoning, 1995, pp. 491-500.

[11] S. Ontaňón, K. Mishra, N. Sugandh, A. Ram, *"Case-Based Planning and Execution for Real-Time Strategy Games,"* Lectuire Notes in Computer Science Volume 4626, 2007, pp. 164-176.

[12] J. H. Powell, B. M. Hauff, J. D. Hastings, *"Utilizing case-based reasoning and automatic case elicitation to develop a self-taught knowledgeable agent,"* In D. Fu and J. Orkin (Eds.) Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop, 2004.

[13] C.K. Riesbeck, R. Schank, *"Inside Case-based Reasoning,"* Erlbaum, Northvale, NJ, 1989.

[14] J. Ruiz-del-Solar, E. Chown, P. G. Ploeger, *"RoboCup 2010: Robot Soccer World Cup XIV,"* Lecture Notes in Computer Science, Vol. 6556, 2011.

[15] A. Samuel, *"Some studies in machine learning using the game of checkers,"* IBM Journal of Research and Development, 3(3), 1959, pp. 210-229.

[16] P. Ulam, A. Goel, J. Jones, *"Reflection in action: Model-based self-adaptation in game playing agents,"* In D. Fu and J. Orkin (Eds.) Challenges in Game Artificial Intelligence: Papers from the AAAI Workshop, 2004.

[17] I. Watson, *"Case-based reasoning is a methodology not a technology,"* AI-CBR, University of Salford, Salford M5 4WT, UK, 1999.

[18] B. G. Weber, M. Mateas, *"Case-Based Reasoning for Build Order in Real-Time Strategy Games,"* In Proceedings of the Fifth Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE), 2009, pp. 106-111.

[19] J. Wendler, M. Lenz, *"CBR for dynamic situation assessment in an agent-oriented setting,"* In D.W. Aha and J.J. Daniels (Eds.), Case-Based Reasoning Integrations: Papers from the AAAI Workshop, 1998.

[20] J. Wendler, G. A. Kaminka, M. Veloso, M. *"Automatically improving team cooperation by applying coordination models,"* In B. Bell and E. Santos (Eds.) Intent Inference for Collaborative Tasks: Papers from the AAAI Fall Symposium, 2001.