A CBR-Inspired Approach to Rapid and Reliable Adaption of Video Game AI

Sander Bakkes, Pieter Spronck, and Jaap van den Herik

Amsterdam University of Applied Sciences (HvA), CREATE-IT Applied Research
P.O. Box 1025, NL-1000 BA Amsterdam, The Netherlands
Tilburg University, Tilburg center for Creative Computing (TiCC)
P.O. Box 90153, NL-5000 LE Tilburg, The Netherlands
s.c.j.bakkes@hva.nl, {p.spronck, h.j.vdnherik}@uvt.nl

Abstract. Current approaches to adaptive game AI typically require numerous trials to learn effective behaviour (i.e., game adaptation is not rapid). In addition, game developers are concerned that applying adaptive game AI may result in uncontrollable and unpredictable behaviour (i.e., game adaptation is not reliable). These characteristics hamper the incorporation of adaptive game AI in commercially available video games. In this article, we discuss an alternative to these approaches. In the case-based inspired approach, domain knowledge required to adapt to game circumstances is gathered automatically by the game AI, and is exploited immediately (i.e., without trials and without resource intensive learning) to evoke effective behaviour in a controlled manner in online play. We performed experiments that test case-based adaptive game AI on three different maps in a commercial RTS game. From our results we may conclude that case-based adaptive game AI provides a strong basis for effectively adapting game AI in video games.

1 Introduction

Traditionally, the artificial intelligence in video games (which we refer to as "game AI") is static, i.e., does not adapt to dynamic circumstances. This is a problem, because games are created for humans to interact with, and humans are notoriously unpredictable. Game designers desire adaptive techniques to allow the game AI to automatically correct mistakes, adapt to new tactics, and scale to the skill level of the human player. Such adaptation mechanisms, able to function within the time and resource restrictions inherent to video games, are seldom implemented, but have been the focus of several studies in the last decade.

In recent years researchers have increasingly adopted case-based reasoning (CBR) and case-based planning (CBP) approaches in their work in order to deal with the complexities of video games. Often, these case-based approaches are focused on decreasing the time required to learn effective behaviour in online play. For instance, Sharma *et al.* [1] developed an approach for achieving transfer learning in the MADRTS game, by using a hybrid case-based reasoning and reinforcement learning algorithm. Auslander *et al.* [2] used case-based reasoning

to allow reinforcement learning to respond as rapidly as possible to changing circumstances in the UNREAL TOURNAMENT domination game. For many of these approaches it is common that a game has finished before any effective behaviour has been established, or that the game characters do not live sufficiently long to benefit from learning. As a result, it is difficult for the players of a video game to detect and understand that the game AI is learning. This renders the benefits of online learning in video games subjective and unclear [3].

To deal with the relatively long learning times, in our research we focus on an adaptation mechanism that exploits game observations stored in a case base for the purpose of *instant* application to game circumstances. We build upon (1) the ability to gather and identify relevant game observations, and (2) the ability to effectively apply these observations in similar game circumstances. Corresponding case-based approaches have been applied to various game genres (see Aha *et al.* [4] for an overview). We observe that, in most research, relatively simple tasks in relatively simple environments are investigated, e.g., predicting the next action of a player in SPACE INVADERS [5]. However, more recently, case-based research has expanded into the domain of RTS games, albeit predominantly to relatively simple instances of the genre. For instance, Aha *et al.* [4] developed a retrieval mechanism for tactical plans in the WARGUS game, that built upon domain knowledge generated by Ponsen and Spronck [6]. Ontañón *et al.* [7] and Mehta *et al.* [8] established a framework for case-based planning on the basis of annotated knowledge drawn from expert demonstrations in the WARGUS game. Louis and Miles [9] applied case-injected genetic algorithms to learn resource allocation tasks in RTS games. Baumgarten *et al.* [10] established a mechanism for simulating human gameplay in strategy games using a variety of AI techniques, including, among others, case-based reasoning.

A functional requirement for adaptive game AI is that it should be reliable, i.e., adapt in a controlled and predictable manner. One way of increasing the reliability of an adaptation mechanism is to incorporate it in a framework for case-based adaptation. In such a framework, adaptations are performed on the basis of game observations drawn from a multitude of games. The effect of the game adaptations, therefore, can be inferred directly from previous observations that are gathered in the case base. This is what we do in the context of the research discussed in this paper. We implement a case-based mechanism in the RTS game SPRING, that uses a large case base of observations of games previously played between many different game AI's. We show that by using the case base an adaptive AI can take decisions that allow it to gain a high number of victories, and can endow game AI with a means for difficulty scaling.

2 Case-based adaptive game AI for SPRING

Here we describe case-based adaptive game AI. It is an elaboration of previous work with a focus on the CBR aspects [11, 12]. Particularly, we focus on an adaptation mechanism for SPRING. SPRING is a typical and open-source RTS game in which a player needs to gather resources for the construction of units

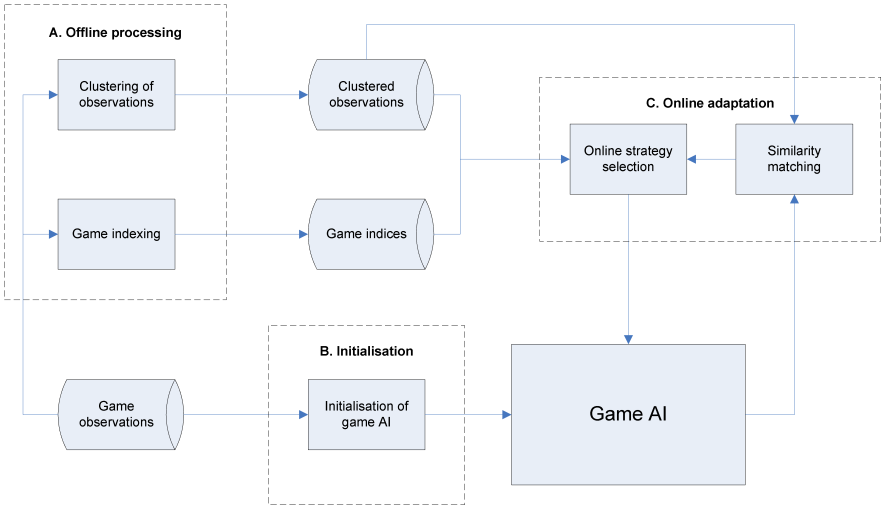

Fig. 1. General adaptation procedure of case-based adaptive game AI.

and buildings. The aim of the game is to use the constructed units and buildings to defeat an enemy army in a real-time battle. A SPRING game is won by the player who first destroys the opponent's 'Commander' unit.

General adaptation procedure. The adaptation procedure, illustrated in Figure 1, consists of three steps: (A) offline processing, (B) initialisation, and (C) online adaptation. In step A, game observations (values for a list of features for a particular game state) that are gathered in the case base are processed offline. The purpose of step A is to generalise over the gathered observations. The offline processing step incorporates components to (1) index gathered games, and (2) cluster observations.

In step B, the initialisation of the game AI is performed. The purpose of step B is to ensure that game AI is effective from the onset of a game. To this end, the step incorporates one component which initialises the game AI with a previously observed, effective game strategy. For the present experiments, we define a game strategy (or opponent strategy) as the configuration of parameters of the game AI that determine strategic behaviour. In the game AI that we will experiment with, we found 27 parameters that determine the game strategy of the game AI. The parameters affect the game AI's behaviour on a high, strategic level. For example, the parameter 'aircraft_rate' determines on a high level how often aircraft units should be constructed. How exactly the constructed aircraft units should be employed is decided by lower-level game AI. All 27 parameters are described in [11, 12].

In step C, the game AI is adapted online. The purpose of step C is to adapt the game AI in such a way that it exhibits behaviour that is effective in actual game circumstances. The online adaptation step incorporates components (1) to perform similarity matching, and (2) to perform online strategy selection. Besides

the similarity matching, online strategy selection exploits the game indices and the clusters of observations that were established offline in step A.

Game indexing. 'Game indexing' is employed in step A. As calculating the game indices is computationally relatively expensive, as all stored game observations need to be processed, it is performed offline. The calculated game indices are exploited for online strategy selection (in step C). We define a game's index as a vector of fitness values, containing one entry for each observed game state. The fitness values represent the desirability of the observed game states. To calculate the fitness value of an observed game state, we use an accurate evaluation function that was discussed in previous work [13].

Clustering of observations. 'Clustering of observations' is employed in step A. Clustering of observations being computationally expensive, as all stored game observations need to be processed, it as a consequence is performed offline. The established clustering of observations is exploited for online strategy selection (in step C). As an initial means to cluster similar observations, we apply the standard k-means clustering algorithm [14]. The metric that expresses an observation's position in the cluster space is determined by the composed sum of observational features, that also is applied for similarity matching.

Initialisation of game AI. 'Initialisation of game AI' is employed in step B. It concerns the selection of a game strategy that is adopted by the game AI at the start of the game. To select intelligently the strategy that is initially followed by the game AI, we need to determine which strategy the opponent player is likely to employ. To this end, we model opponent players based on actual game observations. In the current experiments, we construct opponent models on the basis of observations of the parameter values of the opponent strategies, which indicate the strategic preferences of particular opponents.

The considerations given above lead us to define the procedure to initialise the game AI as follows. First, determine the actual parameter values of the game strategy that is adopted by the opponent player. Second, determine in which parameter bands [15] the opponent strategy can be abstracted. We define three bands for each parameter, 'low', 'medium' and 'high'. Third, initialise the game AI with a strategy that was observed as effective against the most similar opponent. We consider a strategy effective when in previous play it achieved a predefined goal (thus, the game AI will never be initialised with a predictably ineffective strategy). Moreover, we consider opponents strictly similar when the abstracted values of the parameter bands are identical.

Similarity matching. 'Similarity matching' is employed in step C. It is supportive for the component to perform online strategy selection. For selecting an effective game strategy, the similarity matching component compares directly the strategic similarity of game observations. As a first step to match observations for similarity, the selection of the features and the weights assigned to each feature are determined by the researchers, to reflect their expertise with the game environment, considering further improvements a topic for future research. To compare a given observation with another observation, we use six observational features to provide measures for strategic similarity, namely (1) phase of the

game (i.e., opening, pre-midgame, midgame, pre-endgame, endgame [11]), (2) material strength, (3) commander safety, (4) positions captured, (5) economical strength, and (6) unit count. The first three features are also applied for establishing our evaluation function [13]. Features four to six are incorporated to provide additional measures for strategic similarity.

Our function to calculate the strategic similarity is defined by a composed sum. The terms concern the absolute difference in features values. By default, the features are assigned a weight of one. The first term is composed by the feature 'phase of the game' and 'unit count'. The feature 'unit count' is assigned a weight of 0.5, to reflect its lesser importance. The feature 'phase of the game' is incremented by a value of one, to enforce a positive feature value in the case that there is no difference in the phase of the game. As a result, this leads us to denote the function to calculate strategic similarity as follows.

$$\begin{aligned} \text{similarity}(obs1, obs2) = {} & (\ (1 + \text{diff\_phase\_game}(obs1, obs2)) \ * \ (0.5 * \text{diff\_unit\_count}(obs1, obs2))\ ) \\ & + \text{diff\_material\_strength}(obs1, obs2) \ + \ \text{diff\_commander\_safety}(obs1, obs2) \\ & + \text{diff\_positions\_captured}(obs1, obs2) \ + \ \text{diff\_ecc\_strength}(obs1, obs2) \end{aligned} \quad (1)$$

We noted that game observations are clustered. As a result, calculating the similarity between clustered observations is computationally relatively inexpensive. This is important, as similarity matching is performed online (in step C).

Online strategy selection. 'Online strategy selection' is performed in step C. It concerns selecting online which strategy to employ in actual play. Online strategy selection is performed at every phase transition [13] of the game. The selection procedure consists of three steps. First, we preselect the N games in the case base that are most similar to the current game. To this end, we use the computed game indices to preselect the games with the smallest accumulated fitness difference with the current game, up until the current game state. Second, from the preselected N games, we select the M games that satisfy a particular goal criterion. The goal criterion can be any metric to represent preferred behaviour. In our experiments, the goal criterion is a desired fitness value. For instance, a desired fitness value of one hundred represents a significant victory, and a fitness value of zero represents a draw situation for the players, which may be considered balanced gameplay. Third, of the selected M games, we perform the strategy of the game observation that is most similar to the current game state in terms of strategic features. We note that performing strategies associated with similar observations may not necessarily yield the same outcome when applied to the current state. It may happen that observations that are strategically similar may result from distinct circumstances earlier in the game. Therefore, to estimate the effect of performing the retrieved game strategy, we measure the difference in fitness values between the current and the selected observation, and adjust the expected fitness value by linear extrapolation.

3 Experimental setup

In our experimental setup, we perform three steps: A. Gathering feature data, B. Testing the adaptation mechanism, and C. Asssessing the performance. These three steps are discussed next.

A. Gathering feature data. We collect data of the defined observational features from SPRING games in which two game AIs are pitted against each other. As opponent player, we use the 'AAI (original)' game AI, as it is shipped with the game. As friendly player, we use the 'AAI (cb)' game AI; the original AAI opponent with as only difference it can utilise the case-base to adapt behavioural parameters. Feature data is collected on three different RTS maps. The three maps are (a) SmallDivide, (b) TheRing, and (c) MetalHeckv2. For more details on the maps we refer the reader to [11]. For gathering feature data, we simulate competition between different players. This is performed for each game by pseudo-randomising the defined 27 strategic parameters of each player of the game. The pseudo-randomisation results in the players following a randomly generated strategic *variation* of an effective strategy. All games from which observations are gathered are played under identical conditions.

B. Testing the adaptation mechanism. We perform two different experiments with the adaptation mechanism. In the first experiment, we test to what extent the mechanism is capable of adapting effectively to behaviour of the opponent player. The experiment is performed in play where the adaptive 'AAI (cb)' game AI is pitted against two types of opponent, (1) against the original AAI opponent, and (2) against a random opponent. For play against the latter type of opponent, the adaptive game AI is pitted against the original AAI opponent that is initialised with a randomly generated strategy. That is, in each trial of the experiment, a variation of an effective strategy is generated randomly. On each of the three RTS maps (i.e., SmallDivide, TheRing, and MetalHeckv2) we perform 150 adaptation trials for play against each two types of opponent. All adaptation trials are performed under identical conditions.

In the second experiment, we test to what extent the mechanism is capable of upholding a draw position. The experiment is performed in play where the adaptive 'AAI (cb)' game AI is pitted against the same two types of opponent as the first experiment. The second experiment is performed on the default map of the SPRING game, the map SmallDivide. On the map, we perform 150 adaptation trials for play against each two types of opponent. All adaptation trials are performed under identical conditions.

For offline clustering of observations, k is set to 10 per cent of the total number of observations. Before the game starts, the initial strategy is determined. Online (i.e., while the game is in progress) strategy selection is performed at every phase transition. The parameter N for online strategy selection is set to 50, and the parameter M is set to 5.

C. Assessing the performance. To establish a baseline for comparing the experimental results, both experiments are performed in a setting where the adaptation mechanism is disabled. In this setting, the game AI does not intel-

Table 1. Effectiveness of case-based adaptive game AI.

		Original opponent		*Random opponents*	
Adaptation mode	Trials	Goal ach.	Goal ach. (%)	Goal achv.	Goal ach. (%)
SmallDivide					
Disabled	150	59	39%	71	47%
Basic	150	115	77%	96	64%
OM	150	135	90%	136	91%
TheRing					
Disabled	150	90	60%	76	51%
Basic	150	122	81%	93	62%
OM	150	127	85%	93	62%
MetalHeckv2					
Disabled	150	70	47%	54	36%
Basic	150	124	83%	60	40%
OM	150	130	87%	79	53%

ligently determine the initial strategy, but instead randomly selects the initial strategy, and performs no online adaptation to game circumstances.

For the first experiment, where the adaptation mechanism is set to win the game, the effectiveness is expressed by the number of games that are won by the friendly player when it uses the adaptation mechanism. For the second experiment, where the adaptation mechanism is set to uphold a draw position, the effectiveness is expressed by the amount of time that a draw can be upheld by the player that uses the adaptation mechanism. We consider a game state strictly a draw when its fitness value is zero, with a small variance.

4 Results

Table 1 gives an overview of the results of the first experiment performed in the Spring game, obtained with enabled adaptation mechanism. The relevance of the 'OM' adaptation mode will be discussed in Section 5. Figure 2 displays the obtained median fitness value over all game trials against the original AAI opponent on the map SmallDivide, as a function over the relative game time.

The results reveal that when pitted against the original AAI game AI, the adaptation mechanism improves significantly on the established baseline effectiveness on the map SmallDivide (77%, compared to the baseline 39%) (cf. chi-square test, Cohen [16]). In addition, the adaptation mechanism improves substantially on the established baseline effectiveness on the map TheRing (81%, compared to the baseline 60%). Subsequently, the adaptation mechanism improves significantly on the established baseline effectiveness on the map MetalHeckv2 (83%, compared to the baseline 47%). These results indicate that the adaptation mechanism is generically effective in play against the original AAI game AI. In addition, the results reveal that in play against randomly generated opponents, the adaptation mechanism obtains an effectiveness of 64% on the map SmallDivide. Thereby, it improves on the established baseline effectiveness of 47%. This improvement in effectiveness is consistent with our findings on the map TheRing, where the adaptation mechanism obtains an effectiveness of 62%

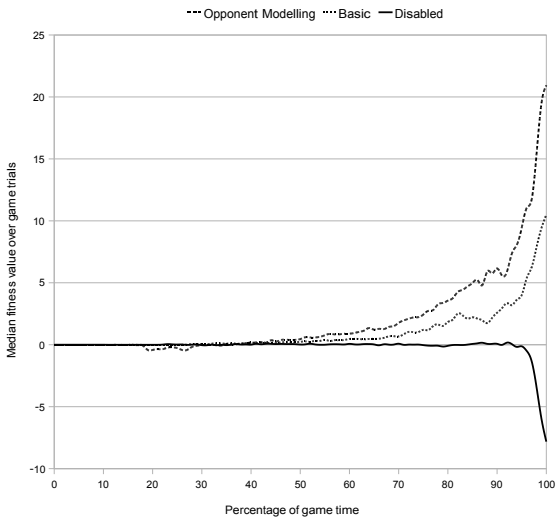

Fig. 2. Median fitness value over all game trials against the original AAI opponent on the map SmallDivide, as a function over the relative game time.

(compared to the baseline 51%). Against randomly generated opponents on the map MetalHeckv2, the adaptation mechanism obtains an effectiveness of 40% (compared to the baseline 36%). These results indicate that even in randomised play, the adaptation mechanism is able to increase the effectiveness of game AI.

Results with difficulty scaling reveal that when pitted against the original AAI opponent, the adaptation mechanism improves significantly on the time in which a draw is upheld (37 minutes, compared to the baseline 27 minutes) (cf. t-test, Cohen [16]). At a certain point in time, inevitably, the game AI will no longer be able to compensate play of the opponent, and the game will either be won or lost by the game AI. Comparable performance is obtained when the adaptation mechanism is pitted against opponents with randomly generated strategies. The results reveal that when pitted against opponents with randomly generated strategies, the adaptation mechanism improves significantly on the time in which a draw is upheld (36 minutes, compared to the baseline 28 minutes).

5 Extending the approach to player modelling

We observed that the final outcome of a SPRING game is largely determined by the strategy that is adopted in the beginning of the game. This exemplifies the importance of initialising the game AI with effective behaviour. In order to do so, a player needs to determine accurately the *opponent* against which it will be pitted. We assume that in video-game practice, (human) game opponents

do not exhibit behaviour as random as in our experimental setup, but will exhibit behaviour that can be abstracted into a limited set of opponent models. Therefore, on the condition that accurate models of the opponent player can be established, game AI should focus on effectively applying models of the opponent in actual game circumstances rather than directly exploiting current game observations [3].

Our general proposal for incorporating opponent modelling into the case-based adaptive game AI is to make the component an integral part of *case-based* adaptation, i.e., let it exploit the case base that is built from a multitude of observed games, for the purpose of automatically establishing models of the opponent player. It should then use the established models to allow the adaptation mechanism to adapt more intelligently the game AI to game circumstances. The adaptation process, we propose, is performed in two steps. First, classify in online play the opponent player. Second, exploit the classification together with previous observations that are gathered in the case base, to reason on the preferred game strategy.

In previous work we discussed how opponent modelling was incorporated in the approach to case-based adaptive game AI [12]. We observed that by incorporating opponent modelling techniques, the case-based adaptive game AI was generally able to increase its effectiveness (cf. the 'OM' entries in Table 1). However, in some circumstances, the increase in effectiveness was relatively modest, and in one situation the effectiveness remained stable. An analysis of this phenomenon shows that our proposed use of opponent modelling works best in circumstances where gameplay is highly strategic (e.g., the map SmallDivide), compared to circumstances where strategic gameplay is a matter of less importance (e.g., the map TheRing). We therefore conjecture that to increase the effectiveness in these circumstances, (1) the opponent models should incorporate additional features that model in more detail facets of the opponent behaviour. In addition, improved results may be established (2) by incorporating knowledge on how heavily the features of the models should be weighted, and (3) by investigating the principal features for certain tasks, e.g., by applying PCA [17].

6 Conclusions

In this paper we discussed a CBR-inspired adaptation mechanism for complex video games. The approach was validated in SPRING, a complex RTS game with imperfect information. We observed that case-based adaptive game AI provides a strong basis for adapting rapidly and reliably the player's behaviour. In addition, we managed to improve the effectiveness of the AI further by incorporating opponent modelling techniques. We conclude that case-based adaptive game AI, especially when enhanced with opponent modelling, can be a worthwhile technique to implement in actual video games.

Acknowledgement. The research reported in this paper was supported by the SIA project 'Smart Systems for Smart Services, the NWO project 'ROLEC', and the Dutch Ministry of Economic Affairs project 'ICIS'.

References

1. M. Sharma, M. Holmes, J. Santamaria, A. Irani, C. Isbell, and A. Ram, "Transfer learning in real-time strategy games using hybrid CBR/RL," in *Proc. of the 20th Int. J. Conf. on AI (IJCAI 2007)*, M. M. Veloso, Ed., 2007, pp. 1041–1046.
2. B. Auslander, S. Lee-Urban, C. Hogg, and H. Muñoz-Avila, "Recognizing the enemy: Combining reinforcement learning with strategy selection using case-based reasoning," in *Proc. o. t. 9th Eur. Conf. on CBR (ECCBR 2008)*, 2008, pp. 59–73.
3. S. Rabin, "Preface," in *AI Game Programming Wisdom 4*, S. Rabin, Ed. Charles River Media, Inc., Hingham, Massachusetts, USA, 2008, pp. ix–xi.
4. D. W. Aha, M. Molineaux, and M. J. V. Ponsen, "Learning to win: Case-based plan selection in a real-time strategy game," in *Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR 2005)*, H. Muñoz-Avila and F. Ricci, Eds. DePaul University, Chicago, Illinois, USA, 2005, pp. 5–20.
5. M. Fagan and P. Cunningham, "Case-based plan recognition in computer games," in *Proceedings of the 5th International Conference on Case-Based Reasoning (ICCBR 2003)*, K. D. Ashley and D. Bridge, Eds. Springer-Verlag, Heidelberg, Germany, 2003, pp. 161–170.
6. M. J. V. Ponsen and P. H. M. Spronck, "Improving adaptive game AI with evolutionary learning," in *Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*, Q. H. Mehdi, N. E. Gough, and D. Al-Dabass, Eds. University of Wolverhampton, Wolverhampton, UK, 2004, pp. 389–396.
7. S. Ontañón, K. Mishra, N. Sugandh, and A. Ram, "Case-based planning and execution for real-time strategy games," in *Proc. of the 7th Int. Conf. on CBR (ICCBR 2007)*, R. O. Weber and M. M. Richter, Eds., 2007, pp. 164–178.
8. M. Mehta, S. Ontañón, and A. Ram, "Authoring behaviors for games using learning from demonstration," in *Proceedings of the Workshop on Case-Based Reasoning for Computer Games, 8th Int. Conf. on Case-Based Reasoning (ICCBR 2009)*, L. Lamontagne and P. G. Calero, Eds., 2009, pp. 107–116.
9. S. J. Louis and C. Miles, "Playing to learn: Case-injected genetic algorithms for learning to play computer games," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 669–681, dec 2005.
10. R. Baumgarten, S. Colton, and M. Morris, "Combining AI methods for learning bots in a real-time strategy game," *Int. J. on Computer Game Technologies*, vol. 2009, 2009, article ID 129075. Special issue on AI for Computer Games.
11. S. C. J. Bakkes, P. H. M. Spronck, and H. J. Van den Herik, "Rapid and reliable adaptation of video game AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 2, pp. 93–104, 2009.
12. ——, "Opponent modelling for case-based adaptive game AI," *Entertainment Computing*, vol. 1, no. 1, pp. 27–37, 2009.
13. S. C. J. Bakkes and P. H. M. Spronck, "Automatically generating a score function for strategy games," in *AI Game Programming Wisdom 4*, S. Rabin, Ed. Charles River Media, Inc., Hingham, Massachusetts, USA, 2008, pp. 647–658.
14. J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 28(1), pp. 100–108, 1979.
15. R. Evans, "Varieties of Learning," in *AI Game Programming Wisdom*, S. Rabin, Ed. Charles River Media, Inc., Hingham, Massachusetts, USA, 2002, pp. 571–575.
16. P. R. Cohen, *Emperical Methods for Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, USA, 1995.
17. K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, vol. 2, no. 6, pp. 559–572, 1901.