Entertainment Computing 1 (2009) 27–37

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: ees.elsevier.com/entcom

Opponent modelling for case-based adaptive game AI

Sander C.J. Bakkes *, Pieter H.M. Spronck, H. Jaap van den Herik

Tilburg centre for Creative Computing (TiCC), Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands

ARTICLE INFO

Article history:
Received 9 July 2009
Revised 2 September 2009
Accepted 10 September 2009

Keywords:
Game AI
Adaptive behaviour
Opponent modelling
Rapid adaptation
Reliable adaptation
RTS games

ABSTRACT

In previous work we introduced a novel approach to adaptive game AI that was focussed on the rapid and reliable adaptation to game circumstances. We named the approach 'case-based adaptive game AI'. In the 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. In the research discussed in this article we investigate to what extent incorporating *opponent modelling* enhances the performance of case-based adaptive game AI. In our approach, models of the opponent players are generated automatically, on the basis of observations drawn from a multitude of games. We performed experiments that test the enhanced approach in an actual, complex RTS game, and observed that the effectiveness of case-based adaptive game AI increases significantly when opponent modelling is incorporated. From these results we may conclude that opponent modelling further improves the basis for implementation of case-based adaptive game AI in commercially available video games.

1. Introduction

Over the last decades, modern video games have become increasingly realistic with regard to visual and auditory presentation. However, game AI has not reached a high degree of realism yet. Game AI is typically based on non-adaptive techniques [1,2]. A major disadvantage of non-adaptive game AI is that once a weakness is discovered, nothing stops the human player from exploiting the discovery. The disadvantage can be resolved by endowing game AI with adaptive behaviour, i.e., the ability to learn from mistakes. Adaptive game AI can be established by using machine-learning techniques, such as artificial neural networks or evolutionary algorithms.

In practice, adaptive game AI in video games is seldom implemented because currently it requires 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). The general goal of our research is to investigate to what extent it is possible to establish game AI capable of adapting rapidly and reliably to game circumstances. To allow rapid and reliable adaptation in games, in previous research we introduced an approach to behavioural adaptation in video games that is inspired by the human capability to solve problems by generalising over previous observations in a restricted problem domain. This approach we named 'case-based adaptive game AI' [3,4]. The present research builds upon the aforementioned work by incorporating *opponent modelling*. Thereby we aim at increasing the effectiveness of case-based adaptive game AI in adapting rapidly and reliably to game circumstances.

The outline of this article is as follows. To provide context for the reader, we first provide an extensive overview of opponent modelling (Section 2). Subsequently, we outline concisely our approach to establish case-based adaptive game AI (Section 3). Next, we discuss how we incorporate opponent modelling in the approach (Section 4). Then, we report on the experiments that investigate to what extent incorporating opponent modelling enhances the performance of case-based adaptive game AI (Section 5), and discuss the experimental results (Section 6). Finally, we provide conclusions of the present research (Section 7).

2. Opponent modelling

Opponent modelling is an important research area in game playing. Opponent modelling concerns establishing models of the opponent player, and utilising the models in actual play. In general, an opponent model is an abstracted description of a player or of a player's behaviour in a game. The goal of opponent modelling is to improve the capabilities of the artificial player by allowing it to adapt to its opponent and exploit his weaknesses [5–8]. Even if a game-theoretical optimal solution to a game is known, a computer program that has the capability to model its opponent's behaviour may obtain a higher reward. A recent example that illustrates the

* Corresponding author. Tel.: +31 13 466 3558; fax: +31 13 466 2892.
E-mail addresses: s.bakkes@uvt.nl (S.C.J. Bakkes), p.spronck@uvt.nl (P.H.M. Spronck), h.j.vdnherik@uvt.nl (H. Jaap van den Herik).

1875-9521/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.entcom.2009.09.001

importance of opponent modelling, derived from Fürnkranz[9], is as follows.

Consider, the game of roshambo (also known as rock-paper-scissors), where if both players play their optimal strategies (i.e., randomly select one of their three moves), either player can expect to win one third of the games (with one third of the games drawn). However, against an opponent that always plays rock, a player that is able to adapt his strategy to always playing paper can maximize his reward, while a player that sticks with the 'optimal' random strategy will still win only one third of the games.

The general concept of modelling the opponent's strategy is regarded as important by many researchers [10–15]. In addition, researchers state that opponent models are sorely needed to deal with the complexities of state-of-the-art video games [16,**?**]. One of the grand challenges in this line of work are games like poker, where opponent modelling is crucial to improve over game-theoretically optimal play [18].

The remainder of this section is organised as follows. To provide context for the reader, we first give an overview of opponent modelling in classic games (2.1). Subsequently, we give an overview of opponent modelling in video games (2.2).

2.1. Opponent modelling in classic games

In classic games, opponent modelling has as its main goal raising the game results of the own (artificial) player [16]. The objective is to exploit the opponent's weaknesses. Better game results are positively correlated with a higher playing strength. Computer programs that play classic games generally incorporate search techniques to find possible game actions by the opponent, of which a model can be constructed. As a result, the role of opponent modelling in classic games is to *guide the search process* towards improved results.

In the remainder of this subsection, we provide a concise history of opponent modelling in classic games (2.1.1), and describe the state of the industry with regard to incorporating opponent modelling techniques (2.1.2).

2.1.1. History

In the domain of classic games, opponent modelling is a research topic that was envisaged already a long time ago. Van den Herik [16] observe that, for instance, in the 1970s chess programs incorporated a contempt factor, meaning that against a stronger opponent a draw was accepted even if the player was +0.5 ahead, and a draw was declined against a weaker opponent even when the player had a minus score.

The first attempt to opponent modelling in classic games was taken by Slagle and Dixon [19], who incorporated rudimentary knowledge of the opponent in the search process. For instance, such knowledge can concern assumptions on the fallibility of an opponent [20]; game AI can consider the chance that the opponent performs a non-rational game action. In related work, Jansen [21,22] investigated using knowledge about the opponent in game-tree search.

Research specifically focussed on the topic of opponent-modelling search started in 1993. In that year, two research groups, one in Haifa, Israel and one in the Maastricht, Netherlands, simultaneously invented a search method that took knowledge of the opponent player into account. They both called it: opponent-model search. In Israel, Carmel and Markovitch [5] investigated in depth the learning of models of opponent strategies. In The Netherlands, Iida et al. [6] investigated potential applications of opponent-model search. An extensive description of the history of opponent modelling is given by Donkers [8].

In the year 1994, Uiterwijk and Van den Herik [23] invented a search technique to speculate on the fallibility of the opponent player. In the 2000s, Donkers et al. [7], Donkers [8] defined probabilistic opponent models, that attempted to avoid the pitfalls of opponent modelling by incorporating the player's uncertainty about the opponent's strategy.

2.1.2. State of the industry

The realisation of most ideas concerning opponent modelling is still in its infancy. There are three successful instances of actual implementation, viz. (1) roshambo [24], (2) iterated prisoner's dilemma [25], and (3) poker [26]. Still, there is a wealth of techniques that are waiting for implementation in actual games [16].

2.2. Opponent modelling in video games

Opponent modelling is of increasing importance in modern video games [9]. In video games, opponent modelling has as its main goal raising the entertainment factor (instead of raising the playing strength) [16]. In the remainder of this subsection, we subsequently describe the role of opponent modelling in video games (2.2.1), the challenges of opponent modelling in video-game environments (2.2.2), approaches applicable for opponent modelling in video games (2.2.3), and the state of the industry (2.2.4).

2.2.1. Role of opponent modelling

In order to raise the entertainment factor of a video game, game AI that incorporates opponent modelling may fulfil two roles: (1) as a companion, and (2) as an opponent. Each role entails distinct requirements for the game AI. A description of the two roles is given next. The description is derived from a review article by Van den Herik et al. [16], to which we refer the reader for more information on the topic.

Companion role : In the companion role, the game AI must behave according to the expectations of the human player. For instance, when the human player prefers an inconspicuous approach to dealing with opponent characters (e.g., by attempting to maintain undetected), he will not be pleased when the computer-controlled companions immediately attack every opponent character that is near. If the companions fail to predict with a high degree of success what the human player desires, they will likely annoy the human player, which is detrimental for the entertainment value of the game.

Opponent role: In the opponent role, the game AI must be able to match the playing skills of the human player, and respond adequately to the player's playing style. This is a difficult task. Research shows that when the opponent characters play too weakly a game against the human player, the human player loses interest in the game [27]. In addition, research shows that when the opponent characters play too strong a game against the human player, the human player gets stuck in the game and will quit playing too [28,29].

2.2.2. Challenges

Houlette [30], Charles and Black [31], Charles et al. [32], and Bohil and Biocca [33] discussed the challenges of opponent modelling in video-game environments, and suggested possible implementations of opponent modelling. A challenge for opponent modelling in video games is that models of the opponent player have to be established (1) in a relatively realistic and complex game environment, (2) with typically little time for observation, and (3) often with only partial observability of the environment.

Once opponent models are established, classification of the opponent player has to be performed in real time. Other computations, such as rendering the game graphics, have to be performed simultaneously. Researchers estimate that generally only 20% of all computing resources are available to the game AI [34]. Of these

20%, a large portion will be spent on rudimentary AI behaviour, such as manoeuvring game characters within the game environment. This implies that only computationally inexpensive approaches to opponent modelling are suitable for incorporation in the game AI.

2.2.3. Applicable approaches

In video games, a common approach to establishing models of the opponent player is by modelling the actions of the player [35], for instance, by using *n*-grams [36]. An alternative approach is to model the *preferences* of opponent players, instead of the actions resulting from those preferences. This preference-based approach [35] identifies the model of an opponent by analysing the opponent's choices in predefined game states.

The opponent model can be either explicit or implicit. An opponent model is explicit in game AI when a specification of the opponents attributes exists separately from the decision-making process. An opponent model is implicit in game AI when the game AI is fine-tuned to a specific (type of) opponent, without the game AI actually referring that opponents attributes [16].

In the preference-based approach, opponent modelling can be seen as a classification problem, where an opponent is classified as one of a number of available models based on data that is collected during the game. Behaviour of the game AI is established based on the classification of the opponent. Modelling of preferences may be viewed as similar to approaches that regard known opponent models as (1) stereotypes, and (2) as an abstraction of observations [37]. In the remainder of the chapter, we follow the preference-based approach.

2.2.4. State of the industry

In recent years there have been several successful implementations of opponent modelling. For instance, Rohs [38] was able to model accurately the preferences of opponent players in the game Civilization IV. Yannakakis [39] investigated the modelling of opponent players, for the purpose of augmenting player satisfaction, and Sailer et al. [40] incorporated opponent modelling to enhance simulation-based planning in RTS games.

Van der Heijden et al. [41] applied opponent modelling to increase the effectiveness of strategies in a simple game mode of the ORTS game. In the card game MACHIAVELLI, which shares numerous characteristics with modern video games, Bergsma [42] was successful in establishing and utilising effectively models of the opponent player. In the game GHOSTS researchers were able to enhance the game AI by allowing it to learn the opponent's playing style [43,44]. In the game of GUESS IT researchers showed that opponent modelling could be used to learn effective game strategies [45]. In the complex SPRING game, Schadd et al. [46] were able to generate automatically accurate models of the opponent player.

In addition, researchers incorporated techniques to predict sequences of user actions [47]; such as the position of opponent players in first-person shooters [48,49], and in the game WORLD OF WARCRAFT [50]. Wong et al. [51] investigated player modelling for a simple 2D shooter game. In the related domain of interactive storytelling, Thue et al. [52] investigated how models of the opponent player can be applied to create stories that can be adapted to fit individual players.

3. Case-based adaptive game AI

In this section we outline concisely our approach to achieving rapidly and reliably adaptive game AI. The approach was coined 'case-based adaptive game AI'. We first provide background information on the topic of adaptive game AI (3.1). Subsequently, we describe the general design of case-based adaptive game AI (3.2). Next, we summarise the results obtained with case-based adaptive game AI that does not incorporate opponent modelling (3.3). A detailed discussion of case-based adaptive game AI is available in previous work [3,4].

3.1. Adaptive game AI

As modern video games present a complex and realistic environment, one would expect characters controlled by game AI in such an environment to behave realistically ('human-like') too. An important feature of human-like behaviour of game AI is the ability to adapt to changing circumstances. Game AI endowed with this ability is called 'adaptive game AI', and is typically implemented via machine-learning techniques. Adaptive game AI may be used to improve the quality of game AI significantly by learning effective behaviour while the game is in progress. Adaptive game AI has been successfully applied to uncomplicated video games [53–55], and to complex video games [56].

To deal with the complexities of video games, in recent years researchers have sometimes adopted case-based reasoning (CBR) and case-based planning (CBP) approaches in their work. For instance, Sharma et al. [57] developed an approach for achieving transfer learning in the MADRTS game, by using a hybrid case-based reasoning and reinforcement learning algorithm. Aha et al. [58] developed a retrieval mechanism for tactical plans in the WARGUS game, that builds upon domain knowledge generated by Ponsen and Spronck [59]. Ontañón et al. [60] established a framework for case-based planning on the basis of annotated knowledge drawn from expert demonstrations in the WARGUS game. Auslander et al. [61] uses case-based reasoning to allow reinforcement learning to respond more quickly to changing circumstances in the UNREAL TOURNAMENT domination game.

Generally, we observe that learning effective behaviour while the game is in progress (i.e., 'online'), typically requires such a large number of learning trials that the process becomes too inefficient for practical use. It is not uncommon that a game has finished before any effective behaviour could be established, or that game characters in a game do not live sufficiently long to benefit from learning. As a result, it is difficult for players to perceive that game AI in fact is learning. This renders the benefits of online learning in video games subjective and unclear [17]. In addition, we observe that even with advanced approaches to game AI, it often is difficult to establish effective behaviour in a controlled and predictable manner. Therefore, the focus of the present research lies on establishing effective behaviour of game AI in a rapid and reliable manner.

3.2. Design of the approach

Case-based adaptive game AI is an approach to game AI where domain knowledge is gathered automatically by the game AI, and is exploited immediately (i.e., without trials and without resource-intensive learning) to evoke effective behaviour. Case-based adaptive game AI is expected to be particularly successful in games that have access to the Internet to store and retrieve samples of gameplay experiences. For instance, in Massive Multiplayer Online Games (MMOGs), observations from many games played against many different opponents are available to the game AI. The approach is illustrated in Fig. 1. It implements a direct feedback loop for control of characters operating in the game environment. The behaviour of a game character is determined by the game AI. Each game character feeds the game AI with data on its current situation, and with the observed results of its actions (see bottom of Fig. 1). The game AI adapts by processing the observed results, and generates actions in response to the character's current situation. An adaptation mechanism is incorporated to

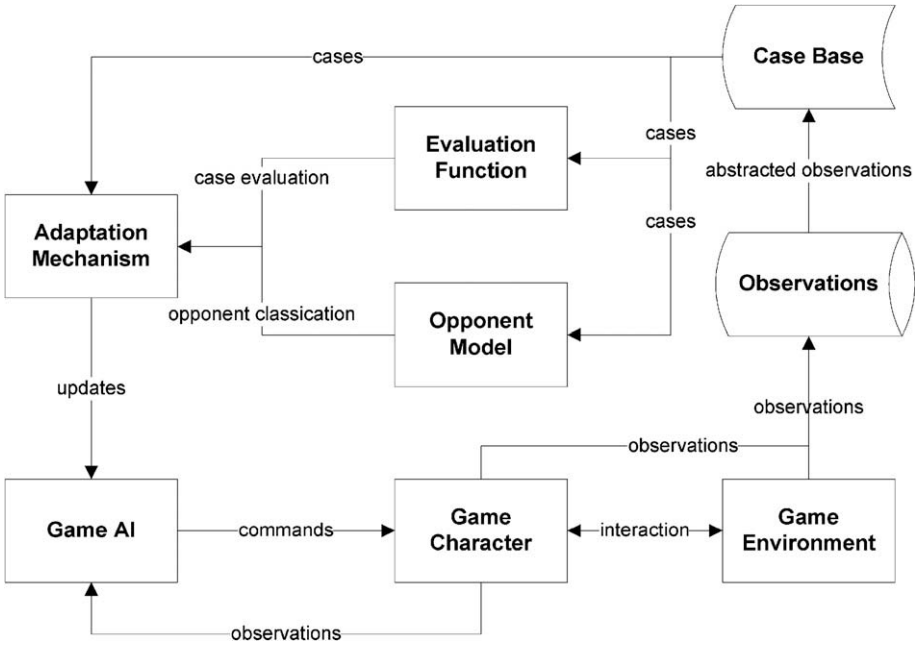

Fig. 1. Case-based adaptive game AI (see text for details).

determine how to adapt the game AI in the best way. For instance, reinforcement learning may be applied to assign rewards and penalties to certain behaviour exhibited by the game AI.

In Fig. 1, for rapid adaptation we have extended the feedback loop by (1) explicitly processing observations from the game AI, and (2) allowing the use of attributes which are not directly observed by the game character (e.g., observations of team-mates). Inspired by the case-based reasoning (CBR) paradigm, the approach collects character observations and game-environment observations, and extracts from those a case base. The case base contains all observations relevant for the adaptive game AI, without redundancies. The observations are time-stamped and structured in a standard format for rapid access. To adapt rapidly to circumstances in the current game, the adaptation process is based on domain knowledge drawn from observations of a *multitude* of games. The domain knowledge gathered in a case base is typically used to extract models of game behaviour, but can also directly be exploited to adapt the AI to game circumstances. In our proposal of case-based adaptive game AI, the case base is used to extract an evaluation function and opponent models. Subsequently, the evaluation function and opponent models are incorporated in an adaptation mechanism that directly exploits the gathered cases during online play.

3.3. Obtained results

In experiments that test case-based adaptive game AI in an actual Real-Time-Strategy (RTS) game [3,4], we observed that the adaptive game AI could play a strong game. We noticed that the case-based adaptive game AI was able to find in the case base strategies that could effectively defeat its opponent AI. As in these experiments the opponent AI was not able to adapt its behaviour, the case-based adaptive game AI could exploit its discoveries indefinitely. Exact results are provided in Section 5. In addition, we observed that even in play with randomised strategic parameter values, the case-based adaptive game AI was often able to find effective strategies in the case base, and was thereby able to improve on the established baseline performance. As randomised play may be considered a simulated way to test the game AI against previously unobserved opponents, this is a satisfactory result.

We noted that the final outcome of the 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 whom it will be pitted. Therefore, in the present research we investigate how our approach to rapidly and reliably adapt game AI can be improved by incorporating opponent modelling techniques.

4. Incorporating opponent modelling

This section discusses how opponent modelling may be incorporated to enhance the game AI of an actual, complex video game. To provide context for the reader, we first describe the game environment in which we implement case-based adaptive game AI (4.1). Subsequently, we describe how we implemented the three main components of case-based adaptive game AI, viz. (1) an evaluation function (4.2), (2) an adaptation mechanism (4.3), and (3) opponent modelling (4.4).

4.1. Game environment

The game environment in which we implement case-based adaptive game AI, is the complex video game SPRING [62]. SPRING, illustrated in Fig. 2, is a typical and open-source RTS game, in which a player needs to gather resources for the construction of units 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.

Modern RTS games typically progress through several distinct phases as players perform research and create new buildings that provide them with new capabilities. The phase of a game can be straightforwardly derived from the observed traversal through the game's tech tree. A tech tree is a directed graph without cycles that models the possible paths of research that a player can take within the game. Traversing the tech tree is (almost) always

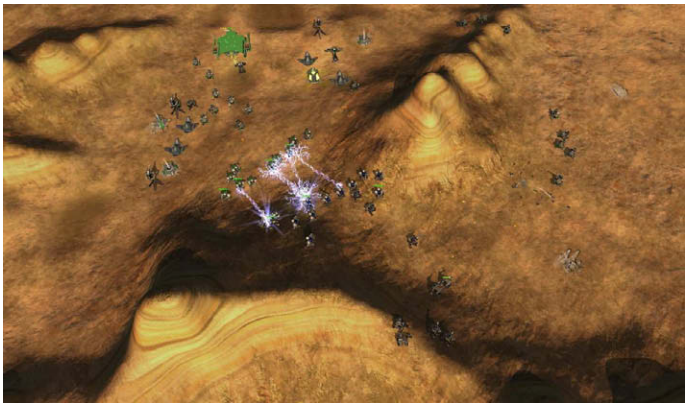

Fig. 2. Screenshot of the SPRING game environment. In the screenshot, the base on the top left is being attacked via a narrow cliff passage.

advantageous, yet there is a cost for doing so in time and game resources. In SPRING, three levels of technology are available. At the start of the game, a player can only construct Level 1 structures and Level 1 units. Later in the game, after the player has performed the required research, advanced structures and units of Level 2 and Level 3 become available.

4.2. Evaluation function

To exhibit behaviour consistent within the game environment presented by modern video games, game AI needs the ability to assess the current situation accurately. This requires an appropriate evaluation function. The high complexity of modern video games makes the task to generate such an evaluation function for game AI a difficult one.

In previous research we discussed an approach to generate automatically an evaluation function for game AI in RTS games [63]. The approach to generate an evaluation function incorporated TD (Temporal Difference) learning [64] to learn unit-type weights, which reflect the actual playing strength of each unit type in the game. Our evaluation function incorporates a parameters to reflect the phase of the game (e.g., opening, end game), and two evaluative terms, one term that represents the material strength and another term that represents the Commander safety.

Results of experiments to test the established evaluation function showed that just before the game's end, the function is able to predict correctly the outcome of the game with an accuracy that approaches 100%. Considering the suddenness in which a SPRING game may be won (i.e., destroying the Commander unit), this is a satisfactory result. In addition, experimental results showed that the evaluation function predicts ultimate wins and losses accurately before half of the game is played. From these results, we concluded that the established evaluation function effectively predicts the outcome of a SPRING game and that the proposed approach is suitable for generating evaluation functions for highly complex video games, such as RTS games. Therefore, we incorporate the established evaluation function in the implementation of our case-based adaptive game AI.

4.3. Adaptation mechanism

In our approach, domain knowledge collected in a case base is exploited for adapting game AI. To generalise over observations with the problem domain, the adaptation mechanism incorporates an offline means to index collected games, and performs an offline clustering of observations. To ensure that game AI is effective from the onset of a game, it is initialised with a previously observed, successful game strategy. For online strategy selection, a similarity matching is performed that considers six experimentally determined features. A detailed description of this procedure is provided in [3,4].

We define the game strategy as the configuration of parameters that determine strategic behaviour. The term 'opponent strategy' is used analogous to game strategy, to reflect that it concerns a game strategy that is employed by the opponent player. In the game AI that we experiment with, we found 27 parameters that determine the game strategy of the game AI. The concerning parameters affect the game AI's behaviour on a high, strategic level, rather than on a low, tactical level. For example, the parameter AIRCRAFT_RATE determines at 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 the Appendix.

4.4. Opponent modelling

Here we discuss how we incorporate opponent modelling into the case-based adaptive game AI; the former being a technique that enables game AI to establish and utilise models of the opponent player. In our approach to case-based adaptation of game AI, opponent models are established *automatically*, on the basis of a case base of game observations. Our goal of utilising the opponent models is to improve the effectiveness of the adaptive game AI. We first describe how we establish models of opponent players in the complex SPRING game (4.4.1). Subsequently, we discuss how we utilise models of the opponent player for the purpose of adapting AI of the game (4.4.2).

4.4.1. Establishing opponent models

We noted that in our approach to case-based adaptation of game AI, opponent models are established automatically, on the basis of game observations gathered in the case base. We establish models of the opponent players as follows. We first define 10 features of an opponent's high-level strategic behaviour. The features are selected by the researchers, to reflect their expertise with the game environment. Naturally, we acknowledge that by manually defining and selecting the game features we potentially limit the accuracy of the models. The investigation of further improvements is considered a topic for future research.

Strategic behaviour, e.g., the opponent's preference of unit type, the focus of an opponent's technological development, the strength of his economy, and the aggressiveness of the opponent, can generally be inferred from observing the feature values during actual play. As SPRING is a typical RTS game, the defined features may be generalised to similar strategic games. The ten defined features are given next.

1. Number of observed k-bot units.
2. Number of observed tank units.
3. Number of observed air units.
4. Number of technologically advanced constructions (i.e., level 2 or higher).
5. Number of metal extractors.
6. Number of solar panels.
7. Number of wind turbines.
8. Time of first attack on one of the metal extractors.
9. Time of first attack on one of the solar panels.
10. Time of first attack on one of the wind turbines.

The first three features express the global strategic preference of an opponent, which is important in determining placement of units and constructions. For instance, in the map SmallDivide (see Fig. 3(a)), the mountain passes can be crossed by k-bot units, but not by tanks. If the game AI can deduce that the opponent is not constructing k-bot units, it can safely distribute resources for a purpose other than defending the mountain passes.

The fourth feature expresses the technological development of a player. If the game AI can deduce that the opponent is constructing advanced units, it can respond by also construction such units.

The fifth, sixth and seventh feature express the strength of an opponent's economy, and by implication, the strength of the opponent's army. If the game AI can deduce that the opponent's economy is relatively weak, it can safely focus on increasing the strength of its own army before launching an attack on the opponent.

The 8th, 9th and 10th feature express the aggressiveness of the opponent player. If the game AI can deduce that the opponent follows an aggressive playing style, which implies that it launches many relatively small attacks, it can attempt to build defenses to withstand the small attacks, and in parallel constructing a relatively strong army. The time of the first attack is expressed in terms of game states that have passed since the game started.

In establishing the opponent models, we consider that we wish to utilise the models in a relatively early state of playing the game, when adapting the game AI still has a relatively strong effect on the final outcome of the game. Therefore, we aim to gather feature data of observed opponent behaviour in a specific state that is relatively early in the game, but that is not too early for observing strategic choices of the opponent player. In our experiments, opponent models are established after 150 game states of play, which amounts to approximately 10 min of real-time play. Based on the feature data gathered after observing numerous games, opponent models are generated by clustering the feature data via the standard *k*-means clustering algorithm [65]. To determine automatically the difference in opponent behaviour expressed by the feature data, the Euclidean distance measure is incorporated in the applied clustering algorithm.

4.4.2. Utilising opponent models

We utilise models of the opponent player for the purpose of adapting AI of the SPRING game. The established opponent models are utilised as follows. We extend the offline processing phase of case-based adaptive game AI, by labelling each game in the case base with information about the opponent that the game AI was pitted against. This process is performed by a classification algorithm, which classifies the game AI's opponent on the basis of observed feature data of the concerning game. As we established opponent models by a clustering of feature data, the classification of an opponent is straightforward; namely by calculating the nearest neighbouring cluster of opponents for the given gameplay sample. The classification of the opponent player is utilised for (1) initialisation of game AI, and (2) online strategy selection in actual play. This is discussed below.

1. Initialisation of game AI (with OM): The procedure to select intelligently the strategy initially followed by the game AI is as follows. First, based on previous observations, we determine which opponent the game AI is likely to be pitted against in a new game. In our experiments, we consider it most likely that the game AI will be pitted against the opponent that over the course of numerous games has been observed the most. Subsequently, we initialise the game AI with the game strategy that in previous games has proven most effective against this particular opponent. We consider a strategy effective when in previous play it achieved a set goal criterion (thus, the game AI will never be initialised with a predictably ineffective strategy). 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 100 represents a significant victory, and a fitness value of 0 represents a situation where both players are tied, which may be considered balanced gameplay.

2. Online strategy selection (with OM): Here we select online which strategy to employ when a transition in the phase of the game takes place. When opponent models are available, the procedure of online strategy selection extends that of online strategy selection without opponent modelling.

In the case that no opponent models are available, the procedure is as follows. 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 indexes to preselect the games with the smallest accumulated fitness difference with the current game, up until the current game state. Second, of the preselected *N* games, we select the *M* games that satisfy a particular goal criterion (e.g., obtain a positive fitness value). 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. The features for determining the strategic similarity are (1) material strength, (2) commander safety, (3)

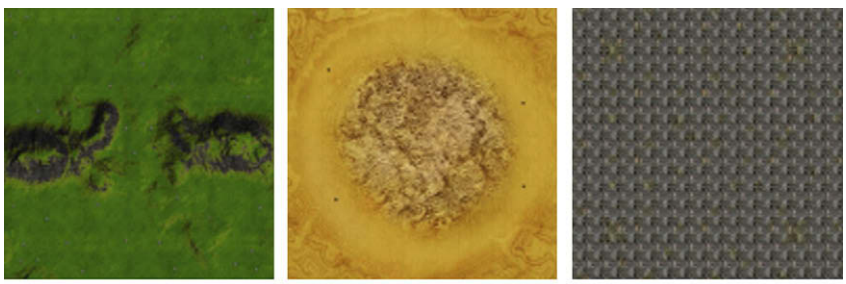

Fig. 3. The three maps that were used in our experiments.

phase of the game, (4) positions captured, (5) economical strength, and (6) unit count. The strategic features and the calculation of similarity is discussed in more detail in previous work [3].

Naturally, we have to consider that performing strategies associated with similar observations may not yield the same outcome when applied to the current state. 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 straightforwardly compensate the expected fitness value. For instance, consider that after playing the game for a certain amount of time, the fitness value of the current game is -5, and that the fitness value of a similar game at that same time was +5, and resulted ultimately in a fitness value of +10 when the game had finished. In this situation, we estimate that applying the game strategy under consideration will result ultimately in a fitness value of 0.

In the case that opponent models are available, there is an additional moment for game adaption besides at the occurrence of phase transitions, namely at the moment at which the opponent player can be classified accurately. That is, the same moment at which the opponent models were established previously. Recall that the game AI is initialised with a game strategy on the basis of a prediction on who the opponent player will be. The game strategy is adapted if the initial prediction of the opponent player is different from the actually observed opponent.

Furthermore, to compare with an increased reliability the similarities between the current game, and games that are gathered in the case base, the classification of the opponent player is incorporated in the process of selection of the game strategy, that is described above. The classification of the opponent player is incorporated as a means to narrow down the preselection process, by preselecting the N games with the smallest accumulated fitness difference with the current game, that as an additional requirement have also been played against the actually observed opponent. We assume that the proposed application of opponent modelling allows for utilising game strategies more effectively. Our experiments investigate our assumption.

5. Experiments

This section reports on the experiments that investigate to what extent incorporating opponent modelling enhances the performance of case-based adaptive game AI. We first describe the experimental setup (5.1). Subsequently, we discuss the performance evaluation (5.2). Next, we report on the automatically generated opponent models (5.3). Finally, we discuss the results obtained with game adaptation (5.4).

5.1. Experimental setup

To test our implementation we start collecting observations of games where two game AIs are pitted against each other. Multiple Spring game AIs are available. We found one open-source game AI, which the author named 'AAI' [66]. AAI is under active development, and is regarded stable and effective in general play. We enhanced this game AI with the ability to collect game observations in a case base, and the ability to disregard radar visibility so that perfect information on the environment was available.[1] As opposing player, we used the original AAI game AI.

[1] Game developers choose generally to provide the game AI with perfect information of the environment. Seen from a purist perspective, this is regarded as a form of cheating behaviour. However, one of the goals for providing entertaining game AI is to obtain effectiveness without cheating *obviously*. Game AI which executes actions that are in principle unavailable, will not be regarded as entertaining. By contrast, game AI that exploits discreetly perfect information in order to provide challenging gameplay, will be regarded generally as entertaining.

Table 1
Contents of the case base.

Map	Games in case base	Obs. in case base	Data size (MB)
SmallDivide	325	213.005	650
TheRing	325	128.481	341
MetalHeckv2	325	107.081	201

For collecting observations, we simulate different players competing with other distinct players, by pseudo-randomising the strategic parameters of both players for each game. This results in randomly generated strategic variations of predictably reasonable behaviour. The collection process was as follows. During each game, game observations were collected every 127 game cycles, which corresponds to the decision-making frequency of AAI. With the SPRING game operating at 30 game cycles per second, this resulted in game observations being collected every 4.233 s.

We acknowledge that the amount of offline storage should be low for our approach to be considered practical for implementation in a game-production setting. We therefore store game observations in a lightweight fashion, by only abstracting the position and unit-type of each unit for each game observation. This abstraction, of approximately 3 KB per observation, provides a powerful basis for deriving observational features. Accordingly, a case base was built from 448,567 observations of 975 games, resulting in 1192 MB of uncompressed observational data. Approaches are available to keep reducing the size of the case base, such as offline data compression and subsequent online data decompression [67], and automatic condensation of the case base [68]. However, incorporating these approaches lies outside the scope of the present research.

To determine to what extent case-based adaptive game AI can be applied generically, we tested it while operating in three different RTS maps. To this end, for each map we collected observations from numerous games played on the particular map, and exploit these observations in adaptation trials. The three maps are (a) SmallDivide, (b) TheRing, and (c) MetalHeckv2. All maps are virtually symmetrical and have no water areas. The map SmallDivide, illustrated in Fig. 3(a), is the default map of the SPRING game, and has one choke point in the centre of the map. The map TheRing, illustrated in Fig. 3(b), is a map with an impassable mountain in the centre of the map. The map MetalHeckv2, illustrated in Fig. 3(c), is a map without significant obstacles, that in addition is abundant with metal resources.

All training games and adaptation trials are played under identical starting conditions. An overview of the contents of the case base is given in Table 1. We observe that the amount of gathered observations depends on the structure of the map. For instance, due to the choke point in the centre of the SmallDivide map, games on this map generally take a relatively long time to finish.

Opponent models are established after 150 game states of play, which amounts to approximately ten minutes of real-time play. The parameter k for k-means clustering of opponents is set to ten percent of the total number of games. Empty clusters are removed automatically, in case the particular value of k was set too large.

Before the game starts, offline processing of the case-base, as well as selecting the initial strategy, is performed according to the procedure described in previous work [3,4].[2] Online (i.e., while the game is in progress), strategy selection is performed at every phase transition. The parameter N for online strategy selection (dis-

[2] Offline processing of the case-base takes about 2 min, excluding clustering of observations. One-time only clustering of observations takes about 36 min. Online strategy selection takes about 0.1 s.

cussed in Subsubsection 4.4.2) is set to 50, and the parameter M is set to 5.

5.2. Performance evaluation

To evaluate the performance of the case-based adaptive game AI, we determined to what extent it is capable of adapting effectively to game circumstances. We performed two different experiments. First, we tested to what extent the case-based adaptive game AI is capable of adapting to the original AAI game AI, set to play in a medium playing strength. Second, we tested to what extent the case-based adaptive game AI is capable of adapting to previously unobserved opponents, which is simulated by pitting the game AI against the original AAI game AI, initialised with randomly generated strategies.

To establish a baseline for comparing the experimental results, all experiments are performed in a mode where the case-based adaptation mechanism is *disabled*. In this mode, the game AI no longer intelligently determines the initial strategy, but instead randomly selects the initial strategy, and performs no online adaptation to game circumstances. For each experiment with case-based adaptive game AI, we performed trials where the case-based adaptive game AI was set to win the game (i.e., obtain a positive fitness value). The experiment is performed in a basic mode, and in a mode in which opponent modelling techniques are incorporated. All experimental trials were repeated 150 times.

5.3. Generated opponent models

Opponent models were generated automatically based on observations gathered from play on three distinct maps. Each map was observed over 325 games. On the map SmallDivide, nine opponent models were generated automatically. On the map TheRing and MetalHeckv2, 8 and 9 opponent models were generated automatically, respectively.

The generated models reveal that opponents observed on the map SmallDivide typically employ a defensive playing style, have a preference for constructing advanced buildings, and have a preference for constructing tank units. Opponents observed on the map TheRing are typically similar to those observed on the map SmallDivide, with the difference that they have a preference for constructing k-bot units, instead of tank units. Opponents observed on the map MetalHeckv2 typically employ an aggressive playing style, do not have a preference for constructing advanced buildings, and have a preference for constructing tank units.

5.4. Results of game adaptation

Table 2 gives an overview of the results of the first experiment performed in the SPRING game. In the experiment, the case-based adaptive game AI was pitted against the original AAI game AI on the three different maps. The first column of the table lists the adaptation mode of the case-based adaptive game AI. The second column lists how often the trial was repeated. The third and fourth column list how often the goal was achieved in absolute terms, and in terms of percentage, respectively.

The results reveal that when pitted against the original AAI game AI on the map SmallDivide, the effectiveness of case-based adaptive game AI increases significantly when opponent modelling techniques are incorporated (90%, compared to 77% without opponent modelling). Also on the maps TheRing and MetalHeckv2 the effectiveness increases when opponent modelling techniques are incorporated (85%, compared to 81%, and 87% compared to 83%, respectively). These results indicate that incorporating opponent modelling techniques indeed increases the effectiveness of case-based adaptive game AI. Fig. 4 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.

Table 2
Effectiveness of case-based adaptive game AI against the original opponent.

Adaptation mode	Trials	Goal achv.	Goal achv. (%)
SMALLDIVIDE			
Disabled	150	59	39
Basic	150	115	77
OM	150	135	90
THERING			
Disabled	150	90	60
Basic	150	122	81
OM	150	127	85
METALHECKV2			
Disabled	150	70	47
Basic	150	124	83
OM	150	130	87

Tables 3 and 4 give an overview of the results of the second experiment performed in the SPRING game. In the experiment, the case-based adaptive game AI was pitted against the original AAI, initialised with randomly generated strategies. The legend of Table 3 is equal to that of the first experiment. The legend of Table 4 is as follows. The first column of the table lists the label of the randomly generated opponent. The second column lists how often the trial was repeated. The third, fourth and fifth column list how often the goal was achieved in absolute terms in the mode where case-based adaptive game AI was disabled, where it operated in basic mode, and where it incorporated opponent modelling techniques, respectively. In the bottom row of the table, the average effectiveness over all trials is listed in terms of percentage.

The results displayed in Table 3 reveal that when pitted against the original AAI game AI on the map SmallDivide, initialised with randomly generated strategies, the effectiveness of case-based adaptive game AI increases significantly when opponent modelling techniques are incorporated (91%, compared to 64% without opponent modelling). On the maps TheRing and MetalHeckv2, the effectiveness of case-based adaptive game AI remains stable when opponent modelling techniques are incorporated (62%, compared to 62%), or increases (53%, compared to 40%), respectively. These results confirm our previous indication that applying opponent modelling techniques increases the effectiveness of case-based adaptive game AI.

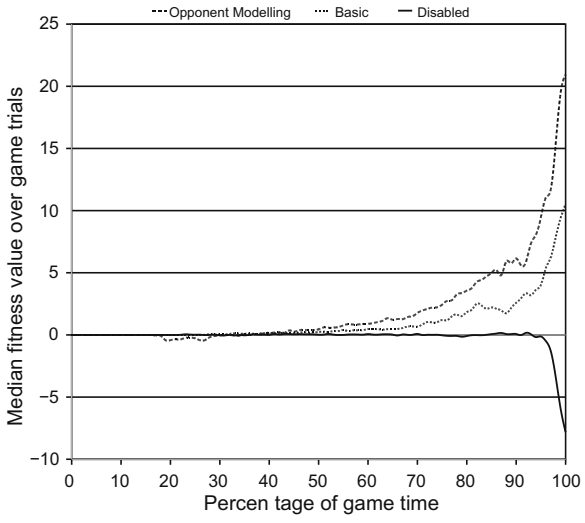

Fig. 4. Median fitness value over all game trials against the original AAI opponent on the map SmallDivide, as a function over the relative game time.

Table 3
Effectiveness of case-based adaptive game AI against random opponents.

Adaptation mode	Trials	Goal achv.	Goal achv. (%)
SMALLDIVIDE			
Disabled	150	71	47
Basic	150	96	64
OM	150	136	91
THERING			
Disabled	150	76	51
Basic	150	93	62
OM	150	93	62
METALHECKV2			
Disabled	150	54	36
Basic	150	60	40
OM	150	79	53

Table 4
Effectiveness of case-based adaptive game AI against sets of random opponents.

SMALLDIVIDE				
Opponent	Trials	Adaptation mode		
		Disabled	Basic	OM
1	10	4	9	9
2	10	6	8	9
3	10	5	5	9
4	10	4	3	5
5	10	7	9	9
6	10	7	6	9
7	10	6	7	9
8	10	7	7	9
9	10	3	6	10
10	10	5	7	9
11	10	6	8	5
12	10	6	8	7
13	10	5	7	9
14	10	5	8	7
15	10	6	6	6
Goal achv. avg. (%)		55%	69%	81%

The findings are analogous to what we observe when case-based adaptive game AI is pitted against sets of randomly generated opponents on the map SmallDivide (Table 4). Again, the effectiveness of case-based adaptive game AI increases when opponent modelling techniques are incorporated (81%, compared to 69% without opponent modelling).

6. Discussion

We observed that by utilising the automatically generated models of the opponent player, the case-based adaptive game AI was able to increase its effectiveness. 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 approach to 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 MetalHeckv2). We therefore theorise that to increase the effectiveness in these circumstances, the opponent models should incorporate additional features that model in more detail aspects of the opponent behaviour. In addition, improved results may be established by incorporating knowledge on how heavily the features of the models should be weighted (for instance by dividing features values in so-called bands [69]), and by investigating the relative importance of each feature (for instance by applying principal component analysis [70]).

Naturally, the case-based adaptive game AI may still be confronted with an opponent that it has not observed previously. In this situation, the inherent generalisation that is provided by the clustering of opponent models may already have led to the game AI being initialised with a strategy that is also effective against the unknown opponent. Should this behaviour still be ineffective, then it can be adapted during online play. To further increase the robustness of case-based adaptive game AI, it may be beneficial to track *dynamically* whether the projected effectiveness of applying a certain strategy is met, and when necessary adapt the strategy directly. In any case, the next time that offline processing is performed, since the case base is updated with the new game, the previously unobserved opponent will be covered in the opponent models, and accordingly predictably effective game AI will be generated with an increased reliability.

In our experiments we set the case-based adaptive game AI to win the game. Many human players, however, may not derive much entertainment from being beaten by a game AI, but would much more enjoy a game AI losing in a 'graceful manner.' In previous work we established a mechanism to scale the difficulty level to the human player, that is capable of maintaining a tie position [3,4]. Related research determined that this approach to difficulty scaling is generally considered entertaining by the human player [29].

7. Conclusions

In this paper we discussed how utilising automatically-generated opponent models may enhance the effectiveness of adaptive game AI. In our approach to adaptive game AI, 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. Opponent models were generated automatically on the basis of observations drawn from a multitude of games, and were utilised to initialise and adapt intelligently the game strategy. We performed experiments that test the approach in an actual RTS game, and observed a significantly increased effectiveness of case-based adaptive game AI when a means of opponent modelling was incorporated. From these results we may conclude that opponent modelling further improves the basis for implementation of case-based adaptive game AI in commercially available video games.

Acknowledgements

The authors wish to thank the anonymous referees for their constructive comments that helped to improve the article considerably. This research is funded by a grant from the Netherlands Organisation for Scientific Research (NWO), in the framework of the ROLEC project (grant number 612.066.406). In addition, the research is funded by the Dutch Ministry of Economic Affairs, in the framework of the Interactive Collaborative Information Systems (ICIS) (grant number BSIK03024).

Appendix A

In this appendix we describe the 27 parameters of strategic behaviour that were used in our experiments. The parameters affect the game AI's behaviour on a high, strategic level, and not so much on a low, tactical level. For example, the parameter AIRCRAFT_RATE determines on a high level how many aircraft units should be constructed. How exactly the constructed aircraft units should be employed is decided by lower-level game AI.

- AIRCRAFT_RATE. Determines how many air units AAI will build (a value of 7 means that every 7th unit will be an air unit; a value of 1 means that constructing air units is disabled).
- AIR_DEFENCE. How often air defence units will be built.
- FAST_UNITS_RATE. Determines the amount of units that will be selected taking their maximum speed into account ($4 \rightarrow 25\%$).
- HIGH_RANGE_UNITS_RATE. Determines the amount of units that will be selected taking weapons range into account ($4 \rightarrow 25\%$).
- MAX_AIR_GROUP_SIZE. Maximum air group size.
- MAX_ANTI_AIR_GROUP_SIZE. Maximum size of anti-air groups (ground, hover or sea).
- MAX_ASSISTANTS. Maximum number of builders assisting construction of other units/buildings.
- MAX_BASE_SIZE. Maximum base size in sectors.
- MAX_BUILDERS. Maximum builders used at the same time
- MAX_BUILDERS_PER_TYPE. How many builders of a certain type may be built.
- MAX_DEFENCES. Maximum number of defences AAI will build in a sector.
- MAX_FACTORIES_PER_TYPE. How many factories of a certain type may be built.
- MAX_GROUP_SIZE. Maximum group size; AAI will create additional groups if all groups of a certain type are full.
- MAX_METAL_COST. Maximum metal cost, units that cost more metal will not be built.
- MAX_METAL_MAKERS. Maximum number of metal makers, set to 0 if you want to disable usage of metal makers.
- MAX_MEX_DISTANCE. Tells AAI how many sectors away from its main base it is allowed to build metal extractors.
- MAX_MEX_DEFENCE_DISTANCE. Maximum distance to base where AAI defends metal extractors with cheap defence-buildings.
- MAX_SCOUTS. Maximum scouts used at the same time.
- MAX_STAT_ARTY. Maximum number of stationary artillery (e.g., big-bertha artillery).
- MAX_STORAGE. Maximum number of storage buildings.
- MIN_AIR_SUPPORT_EFFICIENCY. Minimum efficiency of an enemy unit to call for air support.
- MIN_ASSISTANCE_BUILDSPEED. Minimum workertime/buildspeed of a unit to be taken into account when.
- MIN_FACTORIES_FOR_DEFENCES. AAI will not start to build stationary defences before it has built at least that number of factories.
- MIN_FACTORIES_FOR_STORAGE. AAI will not start to build stationary defences before it has built at least that number of storage buildings.
- MIN_FACTORIES_FOR_RADAR_JAMMER. AAI will not start to build stationary defences before it has built at least that number of radars and jammers.
- MIN_SECTOR_THREAT. The higher the value the earlier AAI will stop to build further defences (if it has not already reached the maximum number of defences per sector).
- UNIT_SPEED_SUBGROUPS. AAI sorts units of the same category (e.g. ground assault units) into different groups according to their max speed (so that slow and fast units are in different groups to prevent the slower ones from arriving in combat much later). This parameter indicates how many different groups will be made.

References

[1] P. Tozour, The perils of AI scripting, in: S. Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA., USA, 2002, pp. 541–547. ISBN 1-584-500-778.

[2] I. Millington, Artificial Intelligence for Games, Chap. Decision Making, Morgan Kaufmann Publishers Inc., San Francisco, California, 2006. pp. 301–471, ISBN 0-124-977820.

[3] S.C.J. Bakkes, P.H.M. Spronck, H.J. Van den Herik, Rapid and reliable adaptation of video game AI, IEEE Transactions on Computational Intelligence and AI in Games 1 (2) (2009) 93–104.

[4] S.C.J. Bakkes, P.H.M. Spronck, H.J. Van den Herik, Rapid adaptation of video game AI, in: P. Hingston, L. Barone (Eds.), Proceedings of the IEEE 2008 Symposium on Computational Intelligence and Games (CIG'08), iEEE XPlore, Catalog Number: CFP08CIG-CDR, 2008, pp. 79–86.

[5] D. Carmel, S. Markovitch, Learning models of opponent's strategy in game playing, in: Proceedings of AAAI Fall Symposium on Games: Planning and Learning, Raleigh, NC, 1993, pp. 140–147.

[6] H. Iida, J.W.H.M. Uiterwijk, H.J. Van den Herik, I.S. Herschberg, Potential applications of opponent-model search. Part 1: the domain of applicability, ICCA Journal 16 (4) (1993) 201–208.

[7] H.H.L.M. Donkers, J.W.H.M. Uiterwijk, H.J. Van den Herik, Probabilistic opponent-model search, Information Sciences 135 (3–4) (2001) 123–149.

[8] H.H.L.M. Donkers, Nosce Hostem – Searching with Opponent Models, Ph.D. thesis, SIKS Dissertation Series No. 2003-13, IKAT, Maastricht University, Maastricht, The Netherlands, 2003.

[9] J.Fürnkranz, Recent advances in machine learning and game playing, ÖGAI Journal 26 (2) (2007) 19–28.

[10] A. Samuel, Some studies in machine learning using the game of checkers, II – recent progress, IBM Journal 11 (6) (1967) 601–617.

[11] D. Knuth, R. Moore, An analysis of alpha-beta pruning, Artificial Intelligence 6 (4) (1975) 293–326.

[12] H. Berliner, Search and knowledge, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 77), 1977, pp. 975–979.

[13] R. Korf, Generalized game trees, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 89), Detroit, MI, 1989, pp. 328–333.

[14] B. Abramson, Expected outcome: a general model of static evaluation, IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (2) (1990) 182–193.

[15] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, D. Szafron, A world championship caliber Checkers program, Artificial Intelligence 53 (2–3) (1992) 273–290.

[16] H.J. Van den Herik, H.H.L.M. Donkers, P.H.M. Spronck, Opponent modelling and commercial games, in: G. Kendall, S. Lucas (Eds.), Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05), 2005, pp. 15–25.

[17] S. Rabin, Preface, in: S. Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA., USA, 2008, pp. ix–xi. ISBN ISBN 1-584-505-230.

[18] D. Billings, L. Peña, J. Schaeffer, D. Szafron, The challenge of Poker, Artificial Intelligence 134 (1–2) (2002) 201–240. Special Issue on Games, Computers and Artificial Intelligence.

[19] J.R. Slagle, J.K. Dixon, Experiments with the M&N tree-searching program, Communications of the ACM 13 (3) (1970) 147–154.

[20] A.L. Reibman, B.W. Ballard, Nonminimax search strategies for use against fallible opponents, in: J.M. Tenenbaum (Ed.), Proceedings of the 3rd National Conference on Artificial Intelligence (AAAI-83), Morgan Kaufmann Publ., San Mateo, CA., USA, 1983, pp. 338–342. ISBN 0-262-510-529.

[21] P.J. Jansen, Using knowledge about the opponent in game-tree search, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1992.

[22] P.J. Jansen, KQKR: speculative thwarting a human opponent, ICCA Journal 16 (1) (1993) 3–18.

[23] J.W.H.M. Uiterwijk, H.J. Van den Herik, Speculative play in computer chess, in: H.J. Van den Herik, I.S. Herschberg, J.W.H.M. Uiterwijk (Eds.), Advances in Computer Chess 7, Maastricht University, Maastricht, The Netherlands, 1994, pp. 79–90. ISBN 9-062-161-014.

[24] D. Egnor, Iocaine power, ICGA Journal 23 (1) (2000) 33–35.

[25] G. Kendall, Iterated prisoner's dilemma competition, 2005 [online]. Available from: http://www.prisoners-dilemma.com/.

[26] D. Billings, Algorithms and Assessment in Computer Poker, Ph.D. thesis, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, 2006.

[27] B. Scott, The illusion of intelligence, in: S. Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA., USA, 2002, pp. 16–20. ISBN 1-584-500-778.

[28] D. Livingstone, D. Charles, Intelligent interfaces for digital games, in: D. Fu, S. Henke, J. Orkin (Eds.), Proceedings of the AAAI-04 Workshop on Challenges in Game Artificial Intelligence, AAAI Press, Menlo Park, CA., USA, 2004, pp. 6–10.

[29] G. Van Lankveld, P.H.M. Spronck, H.J. Van den Herik, Incongruity-based adaptive game balancing, in: Proceedings of the 12th Advances in Computer Games Conference (ACG12), 2009.

[30] R. Houlette, Player modeling for adaptive games, in: AI Game Programming Wisdom 2, Charles River Media, Inc., Hingham, MA., USA, 2004. pp. 557–566 (ISBN 1-584-502-894).

[31] D. Charles, M. Black, Dynamic player modeling: a framework for player-centered digital games, in: Q.H. Mehdi, N.E. Gough, D. Al-Dabass (Eds.), Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004), University of Wolverhampton, Wolverhampton, UK, 2004, pp. 29–35.

[32] D. Charles, M. McNeill, M. McAlister, M. Black, A. Moore, K. Stringer, J. Kucklich, A. Kerr, Player-centered design: player modeling and adaptive digital games, in: Proceedings of the DiGRA Conference: Changing ViewsWorld in Play, 2005, pp. 285–298.

[33] C.J. Bohil, F.A. Biocca, Cognitive Modeling of Video Game Players, Tech. Rep., Media, Interface, and Network Design Labs (MINDLab), Dept. of Telecommunication, Information and Media, Michigan State University, East Lansing, Michigan, USA, 2007.
[34] I. Millington, Artificial Intelligence for Games , Morgan Kaufmann Publishers Inc., San Francisco, California, 2006. ISBN 0-124-977820.
[35] H.H.L.M. Donkers, P.H.M. Spronck, Preference-based player modeling, in: S. Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA., USA, 2006, pp. 647–659. ISBN 1-584-504-579.
[36] F.D. Laramée, Using *n*-gram statistical models to predict player behavior, in: S. Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA., USA, 2002, pp. 596–601. ISBN 1-584-500-778.
[37] J. Denzinger, J. Hamdan, Improving modeling of other agents using tentative stereotypes and compactification of observations, in: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2004), 2004, pp. 106–112.
[38] M. Rohs, Preference-based Player Modelling for Civilization IV, Bachelor's Thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands, 2007.
[39] G.N. Yannakakis, How to model and augment player satisfaction: a review, in: Proceedings of the 1st Workshop on Child, Computer and Interaction (ICMI'08), ACM Press, Montreal, Canada, 2008.
[40] F. Sailer, M. Lanctot, M. Buro, Simulation-based planning in RTS games, in: S. Rabin (Ed.), AI Game Programming Wisdom 4, Charles River Media, Inc., Hingham, MA., USA, 2008, pp. 405–418. ISBN 1-584-505-230.
[41] M.J.M. Van der Heijden, S.C.J. Bakkes, P.H.M. Spronck, Dynamic formations in real-time strategy games, in: P. Hingston, L. Barone (Eds.), Proceedings of the IEEE 2008 Symposium on Computational Intelligence and Games (CIG'08), iEEE XPlore, Catalog Number: CFP08CIG-CDR, 2008, pp. 47–54.
[42] M. Bergsma, Opponent Modelling in Machiavelli, Bachelor's Thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands, 2005.
[43] F. Aiolli, C.E. Palazzi, New Frontiers for Entertainment Computing, vol. 279 of IFIP International Federation for Information Processing, Chap. Enhancing Artificial Intelligence in Games by Learning the Opponent's Playing Style, Springer-Verlag, Heidelberg, Germany, 2008, pp. 1–10.
[44] F. Aiolli, C.E. Palazzi, Enhancing artificial intelligence on a real mobile game, International Journal of Computer Games Technology (2009). article ID 45616.
[45] A.J. Lockett, C.L. Chen, R. Miikkulainen, Evolving explicit opponent models in game playing, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO-2007), 2007, pp. 2106–2113, .
[46] F.C. Schadd, S.C.J. Bakkes, P.H.M. Spronck, Opponent modeling in real-time strategy games, in: M. Roccetti (Ed.), Proceedings of the 8th International Conference on Intelligent Games and Simulation (GAMEON'2007), EUROSIS-ETI, Ghent University, Ghent, Belgium, 2007, pp. 61–68.
[47] B.D. Davison, H. Hirsh, Predicting sequences of user actions, in: Predicting the Future: AI Approaches to Time-Series Problems, AAAI Press, Menlo Park, CA., USA, 1998, pp. 5–12.
[48] C.J. Darken, B.G. Anderegg, Particle filters and simulacra for more realistic opponent tracking, in: S. Rabin (Ed.), AI Game Programming Wisdom 4, Charles River Media, Inc., Hingham, MA., USA, 2008, pp. 419–428. ISBN 1-584-505-230.
[49] S. Hladky, V. Bulitko, An evaluation of models for predicting opponent locations in first-person shooter video games, in: P. Hingston, L. Barone (Eds.), Proceedings of the IEEE 2008 Symposium on Computational Intelligence and Games (CIG'08), iEEE XPlore, Catalog Number: CFP08CIG-CDR, 2008, pp. 39–46.
[50] A.J.J. Valkenberg, Opponent Modelling in World of Warcraft, Bachelor's Thesis, Department of Knowledge Engineering, Maastricht University, Maastricht, The Netherlands, 2007.
[51] C. Wong, J. Kim, E. Han, K. Jung, Human-centered modeling for style-based adaptive games, Journal of Zhejiang University – Science A 10 (4) (2009) 530–534.
[52] D.J. Thue, V. Bulitko, M. Spetch, Player modeling for interactive storytelling: a practical approach, in: S. Rabin (Ed.), AI Game Programming Wisdom 4, Charles River Media, Inc., Hingham, MA., USA, 2008, pp. 633–646. ISBN 1-584-505-230.
[53] P. Demasi, A.J. de, O. Cruz, Online coevolution for action games, International Journal of Intelligent Games and Simulation 2 (3) (2002) 80–88.
[54] T. Graepel, R. Herbrich, J. Gold, Learning to fight, in: Q.H. Mehdi, N.E. Gough, D. Al-Dabass (Eds.), Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004), University of Wolverhampton, Wolverhampton, UK, 2004, pp. 193–200.
[55] S. Johnson, Adaptive AI: a practical example, in: S. Rabin (Ed.), AI Game Programming Wisdom 2, Charles River Media, Inc., Hingham, MA., USA, 2004, pp. 639–647. ISBN 1-584-502-894.
[56] P.H.M. Spronck, M.J.V. Ponsen, I.G. Sprinkhuizen-Kuyper, E.O. Postma, Adaptive game AI with dynamic scripting, Machine Learning 63 (3) (2006) 217–248.
[57] M. Sharma, M. Holmes, J. Santamaria, A. Irani, C. Isbell, A. Ram, Transfer learning in real-time strategy games using hybrid CBR/RL, in: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-2007, 2007, pp. 1041–1046.
[58] D.W. Aha, M. Molineaux, M.J.V. Ponsen, Learning to win case-based plan selection in a real-time strategy game, in: Proceedings of 6th International Conference on Case-Based Reasoning (ICCBR-05), DePaul University, Chicago, Illinois, USA, 2005, pp. 5–20.
[59] M.J.V. Ponsen, P.H.M. Spronck, Improving adaptive game AI with evolutionary learning, in: Q.H. Mehdi, N.E. Gough, D. Al-Dabass (Eds.), Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004), University of Wolverhampton, Wolverhampton, UK, 2004, pp. 389–396.
[60] S. Ontañón, K. Mishra, N. Sugandh, A. Ram, Case-based planning and execution for real-time strategy games, Proceedings of the 7th International Conference on Case-Based Reasoning (ICCBR '07), Springer-Verlag, Heidelberg, Germany, 2007, pp. 164–178, doi:10.1007/978-3-540-74141-1_12. ISBN 978-3-540-74138-1.
[61] B. Auslander, S. Lee-Urban, C. Hogg, H. Muñoz-Avila, Recognizing the enemy: combining reinforcement learning with strategy selection using case-based reasoning, in: Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR 2008), University of Trier, Trier, Germany, 2008, pp. 59–73.
[62] S. Johansson, The Spring Project, 2009 [online]. Available from: http://spring.clan-sy.com/.
[63] S.C.J. Bakkes, P.H.M. Spronck, Automatically generating a score function for strategy games, in: S. Rabin (Ed.), AI Game Programming Wisdom 4, Charles River Media, Inc., Hingham, MA., USA, 2008, pp. 647–658. ISBN 1-584-505-230.
[64] R.S. Sutton, Learning to predict by the methods of temporal differences, in: Machine Learning, Kluwer Academic Publishers, Boston, Manufactured in The Netherlands, 1988, 9–44 [online]. Available from: http://www.cs.ualberta.ca/~sutton/papers/sutton-88.pdf.
[65] J.A. Hartigan, M.A. Wong, A *K*-means clustering algorithm, Applied Statistics 28 (1) (1979) 100–108.
[66] A. Seizinger, AI:AAI, creator of the game AI 'AAI', 2006 [online]. Available: from: http://spring.clan-sy.com/wiki/AI:AAI.
[67] S. Abou-Samra, C. Comair, R. Champagne, S.T. Fam, P. Ghali, S. Lee, J. Pan, X. Li, Data compression/decompression based on pattern and symbol run length encoding for use in a portable handheld video game system, USA Patent #6,416,410, 2002.
[68] F. Angiulli, G. Folino, Distributed nearest neighbor-based condensation of very large data sets, IEEE Transactions on Knowledge and Data Engineering 19 (12) (2007) 1593–1606.
[69] R. Evans, Varieties of learning, in: S. Rabin (Ed.), AI Game Programming Wisdom, Charles River Media, Inc., Hingham, MA., USA, 2002, pp. 571–575. ISBN 1-584-500-778.
[70] K. Pearson, On lines and planes of closest fit to systems of points in space, Philosophical Magazine 2 (6) (1901) 559–572.