Rapid and Reliable Adaptation of Video Game AI

Sander Bakkes, Pieter Spronck, and Jaap van den Herik

***Abstract*—Current approaches to adaptive game AI typically require numerous trials to learn effective behavior (i.e., game adaptation is not rapid). In addition, game developers are concerned that applying adaptive game AI may result in uncontrollable and unpredictable behavior (i.e., game adaptation is not reliable). These characteristics hamper the incorporation of adaptive game AI in commercially available video games. In this paper, we discuss an alternative to these current approaches. Our alternative approach to adaptive game AI has as its goal adapting rapidly and reliably to game circumstances. Our approach can be classified in the area of 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 behavior in a controlled manner in online play. We performed experiments that test case-based adaptive game AI on three different maps in a commercial real-time strategy (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.**

***Index Terms*—Adaptive behavior, game AI, rapid adaptation, real-time strategy (RTS) games, reliable adaptation.**

I. Introduction

OVER the last decades, modern video games have become increasingly realistic in their visual and auditory presentation. However, game AI has not reached a high degree of realism yet. Game AI is typically based on nonadaptive techniques [1], [2]. A major disadvantage of nonadaptive 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 behavior, i.e., the ability to learn from mistakes. Adaptive game AI can be created 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 behavior (i.e., game adaptation is not rapid). In addition, game developers are concerned that applying adaptive game AI may result in uncontrollable and unpredictable behavior (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, we describe an approach to behavioral adaptation in video games that is inspired by the human capability to solve problems by generalizing over previous observations in a restricted problem domain. Our approach can be classified in the area of case-based adaptive game AI.

This paper extends our previous findings [3] by applying case-based adaptive game AI on three different game maps and incorporating a baseline comparison in the discussion of the experimental results. Thereby we strengthen the experimental conclusions, and demonstrate generalizability. In addition, we discuss in more detail the design considerations, we give a more extensive description of related work, and we discuss both the contributions and limitations of our approach.

The outline of this paper is as follows. We first discuss related work in the field of adaptive game AI (Section II). Subsequently, we describe our approach to create a case-based adaptive architecture for game AI (Section III). Next, we discuss an implementation of case-based adaptive game AI (Section IV). Then, we report on the experiments that test case-based adaptive game AI in an actual video game (Section V), which is followed by a discussion of the experimental results (Section VI). Finally, we provide conclusions and describe future work (Section VII).

II. Related Work

This section discusses related work about entertainment and game AI (Section II-A), adaptive game AI (Section II-B), and difficulty scaling (Section II-C). Finally, a summary of the section is provided (Section II-D).

A. Entertainment and Game AI

The purpose of a typical video game is to provide entertainment [1], [4]. Naturally, the criteria of what makes a game entertaining are dependent on who is playing the game. The literature suggests the concept of immersion as a general measure of entertainment [5], [6]. Immersion is the state of consciousness where an immersant's awareness of physical self is diminished or lost by being surrounded in an engrossing, often artificial environment [7]. Taylor argues that evoking an immersed feeling by a video game is essential for retaining a player's interest in the game [6]. As such, an entertaining game should at the very least not repel the feeling of immersion from the player [8]. Aesthetic elements of a video game such as graphical and auditory presentation are instrumental in establishing an immerse game environment. Once established, the game environment needs to uphold some form of *consistency* for the player to remain immersed [8].

To this end, the task for game AI is to control game characters in such a way that behavior exhibited by the characters is

Manuscript received March 24, 2009; revised May 09, 2009; accepted July 21, 2009. First published August 04, 2009; current version published August 19, 2009. This work was supported by The Netherlands Organization for Scientific Research (NWO) under Grant 612.066.406 and was performed in the framework of the ROLEC project.

The authors are with the Tilburg Centre for Creative Computing (TiCC), Faculty of Humanities, Tilburg University, Tilburg 5037 AB, The Netherlands (e-mail: s.bakkes@uvt.nl; p.spronck@uvt.nl; h.j.vdnherik@uvt.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCIAIG.2009.2029084

consistent within the game environment. In a realistic game environment, realistic character behavior is expected. As a result, game AI that is solely focused on exhibiting the most effective behavior [e.g., in a first-person shooter (FPS] game aiming with an accuracy of 100%) is not necessarily regarded as realistic.

Consistency of computer-controlled characters within a game environment is often established with tricks and cheats. For instance, in the game *Half-Life*, tricks were used to create the illusion of collaborative teamwork [8], causing human players to assume intelligence where in fact none existed [9]. While it is true that tricks and cheats may be required to uphold consistency of the game environment, they often are implemented only to compensate for the lack of sophistication in game AI [10]. In practice, game AI in most complex games is not consistent within the game environment, and exhibits what has been called "artificial stupidity" [9] rather than artificial intelligence. To increase game consistency, we primarily aim at creating an optimally playing game AI, as suggested by Buro and Furtak [10]. By increasing the game consistency, by implication, the entertainment value of a video game increases. In complex video games, such as real-time strategy (RTS) games, near-optimal game AI is seen as the basis for obtaining consistency of the game environment [8]. Naturally, we still consider that game AI needs to be tailored to be appropriate for individual players. This is discussed in Section II-C.

B. Adaptive Game AI

Modern video games present a complex and realistic environment in which characters controlled by game AI are expected to behave realistically (human-like). An important feature of human-like behavior 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 behavior while the game is in progress. Adaptive game AI has been successfully applied to uncomplicated video games [11]–[13], and to complex video games [14].

To deal with the complexities of video games, in recent years, researchers have adopted increasingly case-based reasoning (CBR) and case-based planning (CBP) approaches in their work. For instance, Sharma *et al.* developed an approach for achieving transfer learning in the *Madrts* game, by using a hybrid CBR and reinforcement learning algorithm [15]. Aha *et al.* developed a retrieval mechanism for tactical plans in the *Wargus* game [16] that builds upon domain knowledge generated by Ponsen and Spronck [17]. Ontañón *et al.* established a framework for CBP on the basis of annotated knowledge drawn from expert demonstrations in the *Wargus* game [18]. Auslander *et al.* used CBR to allow reinforcement learning to respond more quickly to changing circumstances in the *Unreal Tournament* domination game [19]. Louis and Miles applied case-injected genetic algorithms to learn resource allocation tasks in RTS games [20]. Baumgarten *et al.* established an approach for simulating human game play in strategy games using a variety of AI techniques, including, among others, CBR [21].

Generally, we observe that learning effective behavior while the game is in progress (i.e., online), typically requires an inefficiently large number of learning trials. In addition, it is not uncommon that a game has finished before any effective behavior 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 the game AI is learning. This renders the benefits of online learning in video games subjective and unclear [22]. In addition, we observe that even with advanced approaches to game AI, it is often difficult to establish effective behavior in a controlled and predictable manner. Therefore, the focus of this research is to create rapidly and reliably effective behavior of game AI.

C. Difficulty Scaling

Difficulty scaling is the automatic adaptation of the *challenge* a game poses to the skills of a human player [23]. When applied to game AI, difficulty scaling aims usually at achieving an "even game," i.e., a game wherein the playing strength of the computer and the human player match.

Once near-optimal game AI is established, difficulty-scaling techniques can be applied to downgrade the playing strength of game AI [23] to ensure that a suitable challenge is created for the player. Many researchers and game developers consider game AI, in general, to be entertaining when it is difficult to defeat [24]. Although for strong players that may be true, novice players will not enjoy being overwhelmed by the computer. For novice players, a game is most entertaining when the game is challenging but beatable [25].

The only means of difficulty scaling implemented in most games is typically provided by a "difficulty setting," i.e., a discrete parameter that determines how difficult the game will be. The purpose of a difficulty setting is to allow both novice and experienced players to enjoy the appropriate challenge the game offers. Usually the parameter influences the psychical properties of the opponents, such as their strength. Very rarely the parameter influences the opponents' tactics. Consequently, even on a "hard" difficulty setting, opponents exhibit inferior behavior, despite their high physical strength and health. In addition, it is hard for the player to estimate reliably the difficulty level that is appropriate for himself. Finally, discrete difficulty settings cannot be fine-tuned to be appropriate for each player.

In recent years, researchers have developed advanced techniques for difficulty scaling of game AI. Hunicke and Chapman explore difficulty scaling by controlling the game environment (i.e., the number of weapons and power-ups available to a player) [26]. Demasi and Cruz use coevolutionary algorithms to gradually teach game characters how to behave [27]. Spronck *et al.* use weights assigned to possible game strategies, to determine dynamically whether predictably strong game strategies should be executed [23]. Yannakakis and Hallam provide a qualitative and quantitative means for measuring player entertainment in real time [28].

D. Summary of Related Work

In summary, two desired characteristics of game AI discussed in this section are consistency and challenge. They serve as a guideline of our approach to adaptive game AI. In our approach,

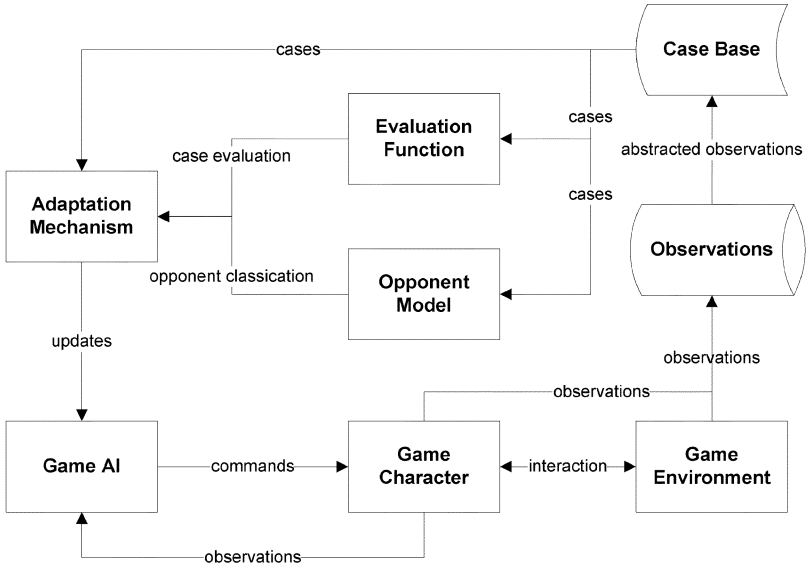

Fig. 1. Case-based adaptive game AI (see text for details).

consistency of the game environment is maintained by adapting rapidly and reliably to game circumstances. In addition, in our approach, we consider that the challenge that is provided by the game AI should be adaptable to fit individual players. Our approach is discussed next.

III. Adaptive Architecture

This section discusses our approach to create an effective adaptive architecture for game AI. First, we discuss the design considerations of the approach (Section III-A). Second, we discuss the approach, which we refer to as case-based adaptive game AI (Section III-B). Third, we discuss the contributions and limitations of the approach (Section III-C).

A. Design Considerations

Game AI should be challenging and consistent with the game environment in which it is situated. To this end, game AI requires the ability to adapt to changing circumstances. Typically, adaptive game AI is implemented for performing adaptation of the game AI in an online and computer-controlled fashion. Improved behavior is established by continuously making (small) adaptations in the game AI. To adapt to circumstances in the current game, the adaptation process normally is based only on observations of *current* game play. This is called incremental adaptive game AI. The incremental approach may be used to improve significantly the quality of game AI by endowing game AI with the capability of adapting its behavior while the game is in progress.

A recurring characteristic of incremental adaptive game AI is its difficulty with establishing *rapid* and *reliable* adaptation of game AI. The reason is that the incremental approach to adaptive game AI requires either 1) a high quality of the domain knowledge used (which generally is unavailable to the AI), or 2) a large number of trials to learn effective behavior online (which in an actual video game is highly undesirable).

Naturally, going through a large number of adaptation trials does not coincide with our goal of adapting rapidly game AI. As a result, one can only adapt rapidly to game circumstances, by improving the quality of the domain knowledge that is exploited. The same holds for our goal of reliably adapting game AI; one can only adapt reliably to game circumstances, when the domain knowledge incorporates accurate estimates on the effect of a game adaptation under consideration. Domain knowledge of such high quality is not available to the adaptation process of incremental adaptive game AI, which is based only on observations of the *current* game play.

Considering the previous discussion, it is clear that incremental adaptive game AI cannot be applied successfully to the extent that it can be used to adapt rapidly and reliably game AI in online game playing conditions. For online adaptation of game AI, capable of rapid and reliable adaptation of game AI, it is therefore necessary that an alternative is established for the incremental adaptive game AI approach. Our proposal for an alternative approach is discussed next.

B. Case-Based Adaptive Game AI

To achieve rapidly and reliably adaptive game AI, we propose an alternative, novel approach to adapting the AI of video games free of the hampering requirements of typical adaptive game AI. We refer to the approach as "case-based adaptive game AI." We define case-based adaptive game AI as 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 behavior.

This approach to adaptive game AI is expected to be particularly successful in games that have access to the internet to store and retrieve samples of game play experiences. For instance, in the popular 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 behavior 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 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 behavior 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 teammates). Inspired by the CBR paradigm, the approach collects character 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 behavior, 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.

The approach to case-based adaptive game AI is inspired by the human capability to reason reliably on a preferred course of action with only a few online observations on the problem domain. Following from the complexity of modern video games, we propose that for effective and rapid use, game observations should be 1) represented in such a way that the stored cases can be reused for previously unconsidered situations, and 2) be compactly stored in terms of the amount of gathered feature data. As far as we know, our approach to rapidly and reliably adapting game AI has not yet been implemented in an actual video game.

C. Contributions and Limitations of the Approach

Game developers may consider using an alternative approach to game AI when they are convinced of its qualitative effectiveness. The main contribution of our research therefore is demonstrating that case-based adaptive game AI can be applied generically in a complex video game. In this paper, we will demonstrate in an RTS game that the approach can be applied both to generate effective game strategies and to scale automatically the difficulty level to the player.

To this end, the approach is adjusting a preexisting game AI, rather than "being the AI" (as most previous CBR approaches to game AI), based on a case base drawn from observations of a multitude of games. Particularly in the popular multiplayer games, the case base is expected to grow rapidly. With a sufficiently large and diverse case base, the adaptation process no longer needs to go through a large number of adaptation trials, but instead can adapt instantly to game circumstances. Furthermore, the game AI will become robust in dealing with nondeterminism, since the case base can be used as a model to predict the results of game strategies. An advantage of the approach tying in with a preexisting game AI is that it enables game developers to control and predict with relative accuracy the behavior that is exhibited by the game AI. In addition, the case base can be utilized for providing feedback on distinct strengths and weaknesses of a player, and can provide inexpensive insight into the balance of a game. Thus, it can help in testing and debugging the game.

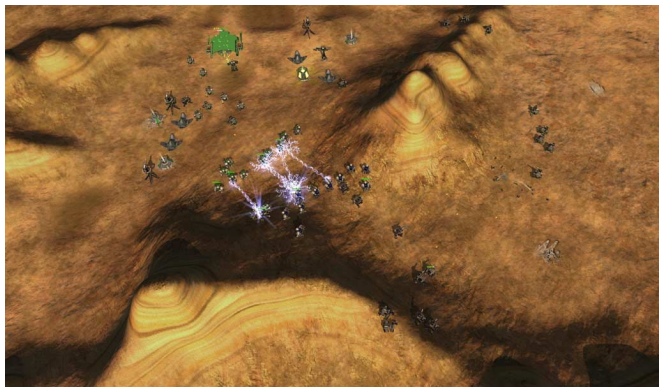

Fig. 2. Screenshot of the *Spring* game environment. The base on the top left is being attacked via a narrow cliff passage.

Naturally, the approach has certain limitations. A first limitation is that it is not fully knowledge free, but requires some domain knowledge to steer the adaptation process. For instance, the features to compare the similarity of game observations need to be established. We assume that such domain knowledge is available to the developers of the game. A second limitation is that for effective adaptation, the case base needs to contain cases relevant to the current circumstances. This limitation is partially addressed by generalizing over stored cases. A third limitation is that the ability to adapt to changing circumstances is restricted by the behavioral expressiveness provided by the game AI that the approach ties in with.

IV. Implementation

This section discusses our implementation of case-based adaptive game AI. We subsequently discuss the game environment in which we implement case-based adaptive game AI (Section IV-A), the established evaluation function (Section IV-B), and an adaptation mechanism inspired by the CBR paradigm (Section IV-C). Previously created opponent models [29] will be incorporated in future research.

A. The Spring Game Environment

The game environment in which we implement case-based adaptive game AI, is the video game *Spring* [30]. *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 a player can take within the game. Traversing parts of the tech tree is (almost) always 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.

B. Evaluation Function

To exhibit behavior consistent within the game environment presented by modern video games, the 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 [31]. The approach to generate an evaluation function incorporated temporal difference (TD) learning [32] to learn unit-type weights, which reflect the actual playing strength of each unit type. Our evaluation function for the game's state is denoted by

$$v(p) = w_p v_1 + (1 - w_p) v_2 \tag{1}$$

where $w_p \in [0 \dots 1]$ is a free parameter to determine the weight of each term v_n of the evaluation function, and $p \in \mathbb{N}$ is a parameter that represents the current *phase of the game*. Our evaluation function incorporates two evaluative terms, the term v_1 that represents the material strength and the term v_2 that represents the commander safety. The selection of the two terms follows our expert knowledge with the game. The terms can be distinctly weighted for each phase of the game, to reflect their varying importance during play of the game (e.g., material balance may be more import early in the game, where commander safety may be more important later in the game).

Previous research performed in the *Spring* environment has shown that the accuracy of situation assessments is closely related to the phase of the game in which they are made [33]. To distinguish phases of the *Spring* game, we map tech levels to game phases and distinguish between when tech levels are "new," and when they are "mature," as indicated by the presence of units with a long construction time. This leads us to define the following five game phases.

- **Phase 1:** level 1 structures observed.
- **Phase 2:** level 1 units observed that have a build time $\geq$ 2500.
- **Phase 3:** level 2 structures observed.
- **Phase 4:** level 2 units observed that have a build time $\geq$ 15 000.
- **Phase 5:** level 3 units or level 3 structures observed.

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%. In addition, experimental results showed that the evaluation function predicts ultimate wins and losses accurately before half of the game is played.[1] From these results, we then 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.

C. Adaptation Mechanism

In our approach, domain knowledge collected in a case base is exploited for adapting game AI. To generalize 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 initialized with a previously observed, successful game strategy. For online strategy selection, a similarity matching is performed that considers six experimentally determined features.

We define the game strategy as the configuration of parameters that determine strategic behavior. 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 used in our experiments, we found 27 parameters that determine the game strategy of the game AI. The concerning parameters affect the game AI's behavior on a high, strategic level, and not 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. All 27 parameters are described in the Appendix.

The adaptation mechanism is algorithmically described below, and is subsequently discussed in detail.

// Offline processing
- A1) Game indexing; calculate indexes for all stored games.
- A2) Clustering of observations; group together similar observations.

// Initialization of game AI
- B1) Establish the (most likely) strategy of the opponent player.
- B2) Determine to which parameter-band values this opponent strategy can be abstracted.
- B3) Initialize game AI with an effective strategy observed against the opponent with the most similar parameter-band values.

[1]We should point out here that a human player would probably score 100% on correctly predicting the outcome of a game in its final stage. The fact that the score function does not achieve human performance is not an indication that it is badly designed for the following two reasons. First, the evaluation function is tuned to make predictions that are good during a large part of the game, not only at the end, and therefore, it will trade prediction accuracy at the end of the game for higher prediction accuracy earlier in the game. Second, if the goal of the game was to destroy all the opponent's units, a correct prediction would be easy to make at the end. However, the goal is to destroy the opponent's cCommander, and we found that it sometimes happens that a player who is behind in material strength can still win (e.g., when the opponent's commander makes a high-risk move, such as attacking strong enemy units on its own).

// Online strategy selection
- C1) Use game indexes to select the N most similar games.
- C2) Of the selected N games, select the M games that best satisfy the goal criterion.
- C3) Of the selected M games, select the most similar observation.
- C4) Perform the game strategy stored for the selected observation.

---

1) Game Indexing (A1): We define a game's index as a vector of fitness values, containing one entry for each time step. These fitness values represent the desirability of all observed game states. To calculate the fitness value of an observed game state, we use the previously established evaluation function [shown in (1)]. Game indexing is supportive for later strategy selection. As it is a computationally expensive procedure, it is performed offline.

2) Clustering of Observations (A2): As an initial means to cluster similar observations, we apply the standard k-means clustering algorithm [34]. Even though this algorithm is effective for our current setup, alternatives such as tree-indexing structures (e.g., kd-trees [35] or cover trees [36]) may be considered when working with increasingly large collections of cases.

The metric that expresses an observation's position in the cluster space comprises a weighted sum of the six observational features that also are applied for similarity matching. Clustering of observations is supportive for later strategy selection. As it is a computationally expensive procedure, it is performed offline.

3) Similarity Matching (A2 and C3): To compare a given observation with another observation, we define six observational features, namely: 1) phase of the game, 2) material strength, 3) commander safety, 4) positions captured, 5) economical strength, and 6) unit count. Similarity is defined by a weighted sum of the absolute *difference* in features values. The selection of the features and the weights assigned to each feature are determined by the experimenter based on experience with the game environment. The weighted sum for both clustering of observations and similarity matching is calculated as follows:

$$\begin{aligned}&\text{similarity}\\&= ((1+\text{diff}(\text{phase\_of\_the\_game}))*(0.5*\text{diff}(\text{unit\_count})))\\&\quad+\text{diff}(\text{material\_strength})+\text{diff}(\text{commander\_safety})\\&\quad+\text{diff}(\text{positions\_captured})+\text{diff}(\text{economical\_strength}).\end{aligned}$$

As observations are clustered, calculating the similarity between observations is computationally relatively inexpensive. This is important, as similarity matching must be performed online.

4) Initialization of Game AI (B1–B3): To select intelligently the strategy initially followed by the game AI, one needs to determine which strategy the opponent is likely to employ. To this end, we model opponent players based on actual game observations. In the current experiment, we construct the opponent models on the basis of observations of the parameter values of the opponent strategies, which indicate the strategic preferences of particular opponents. In future work, we will assume that parameters of the underlying opponent behavior cannot be observed directly, and thus opponent models will have to be established via alternative techniques, such as statistical learning.

Our procedure to initialize the game AI is as follows. Once the opponent strategy has been identified, we determine in which parameter bands [37] the opponent strategy can be abstracted. We define three bands for each parameter: "low," "medium," and "high." We subsequently initialize the game AI with an effective strategy observed against the most similar opponent. We consider a strategy effective when in previous play it achieved a set goal criterion (thus, the game AI will never be initialized with a predictably ineffective strategy), and consider opponents strictly similar when the abstracted values of the parameter bands are identical.

5) Online Strategy Selection (C1–C4): This step selects online which strategy to employ. The procedure is as follows. Using the game indexes, we first preselect the N games with the smallest accumulated fitness difference with the current game, up until the current observation. Subsequently, of the selected N games, we perform the game strategy of the most similar observation of the M games that satisfy a particular goal criterion. The goal criterion can be any metric to represent preferred behavior. 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 game play. 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 concerning game strategy will result ultimately in a fitness value of 0.

V. Experiments

This section discusses experiments that test our implementation of case-based adaptive game AI. We first describe the experimental setup (Section V-A). Subsequently, we discuss the performance evaluation (Section V-B). Next, we discuss the results obtained with game adaptation (Section V-C). Finally, we discuss the results obtained with difficulty scaling (Section V-D).

A. Experimental Setup

To test our implementation we start collecting observations of games where two game AIs are pitted against each other. As multiple *Spring* game AIs are available, we first have to select a game AI that is suitable for our experiments. We found one open-source game AI, which the author of the AI named "AAI" [38]. AAI is under active development, and is regarded stable and effective in general play. The game AI integrates robust resource management, and follows a moderately defensive

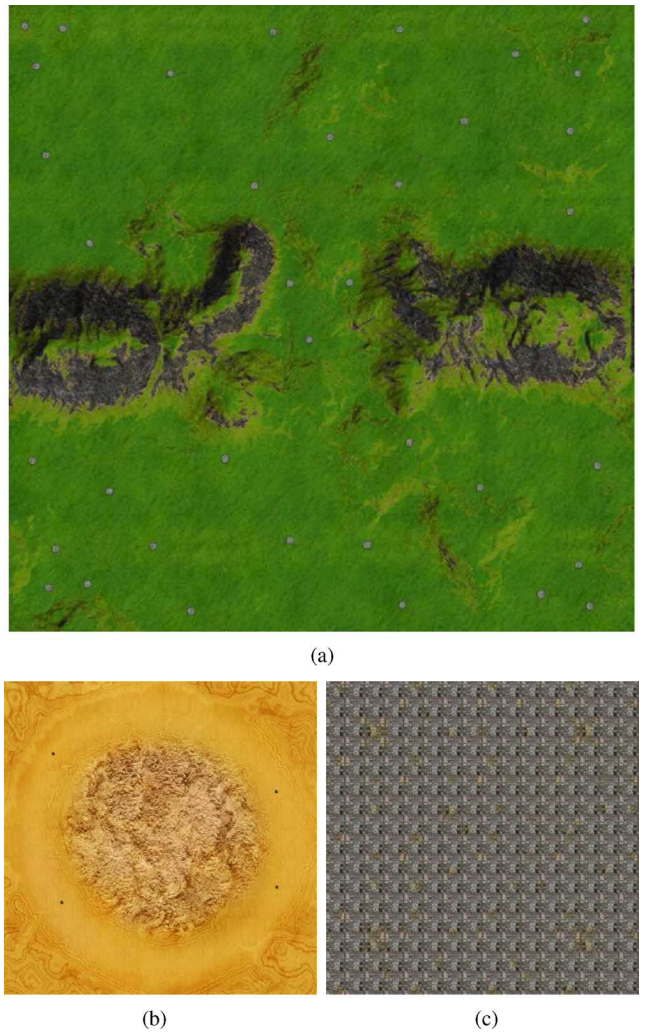

Fig. 3. The three maps that were used in our experiments. (a) SmallDivide. (b) TheRing. (c) MetalHeckv2.

playing style. 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. As opponent player, we used the original AAI game AI.

To determine to what extent case-based adaptive game AI can be applied generically, we test it while operating in three different RTS maps. To this end, for each map, we collect observations from numerous games played on the particular map, and exploit these observations in adaptation trials. The three concerning maps are: 1) SmallDivide, 2) TheRing, and 3) MetalHeckv2. All maps are 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 center of the map. The map TheRing, illustrated in Fig. 3(b), is a map with an impassable mountain in the center of the map. The map MetalHeckv2, illustrated in Fig. 3(c), is a map without significant obstacles, which in addition is abundant with metal resources.

For collecting observations, we simulate competition between different players by pseudorandomizing the strategic parameters of both players for each game. This results in randomly generated strategic variations of predictably reasonable behavior (and not fully random strategic behavior). The collection process was as follows. During each game, game observations were collected every 127 game cycles, which corresponds to the update 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.

TABLE I
CONTENTS OF THE CASE BASE

Map	Games in Case Base	Obs. in Case Base	Data Size
SmallDivide	325	213.005	650 MB
TheRing	325	128.481	341 MB
MetalHeckv2	325	107.081	201 MB

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,[2] resulting in a case base consisting of 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 [39], and automatic condensation of the case base [40]. However, incorporating these approaches lies outside the scope of this research.

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 I. We observe that the amount of gathered observations depends on the structure of the map. For instance, due to the choke point in the center of the SmallDivide map, games on this map generally take a relatively long time to finish.

For offline clustering of observations, k is set to 10% 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. When the case-based adaptive game AI is set to uphold a tie, strategy selection is performed when the difference between the desired goal fitness and the fitness value of the currently observed game state is larger than a predefined threshold value.

As a general setting for online strategy selection, the parameter N for strategy selection is set to 50, and the parameter M is set to 5. The threshold value for adaptation in the upholding tie scenario is set to 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.

[2]Naturally, the effectiveness of behavior established via a case-based approach depends on the quality of the cases that are gathered in the case base. For our setup, where game strategies are established from pseudorandomized self-play, our estimation is that for each map several hundred games must be observed before effective behavior can be established.

Online strategy selection takes about 0.1 s. Experiments are performed on a PC built around an Intel Core 2 Quad CPU @ 2.40 GHz, with 2-GB RAM.

B. Performance Evaluation

To evaluate the performance of the case-based adaptive game AI, we determined to what extent it is capable of effectively adapting to game circumstances. We performed three 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, initialized with randomly generated strategies. Third, as a form of difficulty scaling, we tested to what extent the case-based adaptive game AI is capable of upholding a tie when pitted against the original AAI game AI, also set to play at a medium playing strength.

For each of the first two experiments, we performed a trial where the case-based adaptive game AI was set to win the game (i.e., obtain a positive fitness value). For the third experiment, we set the adaptive game AI to uphold a tie (i.e., maintain a fitness value of 0, while never obtaining a fitness value smaller than -10, or greater than 10). To measure how well the case-based adaptive game AI is able to maintain a fitness value of 0, the variance in fitness value is calculated. A low variance implies that the case-based adaptive game AI has the ability to maintain consistently a predefined fitness value. All experimental trials were repeated 150 times.

To establish a baseline for comparing the experimental results, all experiments on the map SmallDivide are repeated in a configuration where the case-based adaptation mechanism is disabled. In this configuration, 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.

C. Results of Game Adaptation

Table II gives an overview of the baseline results of the first and second experiments performed in the *Spring* game, obtained with disabled case-based adaptation. The first column of each table lists against which opponent the game AI was pitted. The second column lists how often the trial was repeated. The third and fourth columns list how often the goal was achieved in absolute terms, and in terms of percentage, respectively.

The baseline results reveal that by default, the original AAI game AI is initialized at a competitive level on the map SmallDivide, as the game AI with disabled case-based adaptation mechanism is only able to win 39% of the time. In contrast, on the map TheRing, the opponent is more easy to defeat. In addition, the results reveal that in randomized play on two of the maps, the effectiveness of the game AI approximates 50%, as may be expected.

Table III gives an overview of the results of the first and second experiments performed in the *Spring* game, obtained with the case-based adaptation mechanism. The experiments concerned the adaptation ability of the case-based adaptation mechanism. The legend of Table III is equal to that of the table with the baseline results.

TABLE II
BASELINE EFFECTIVENESS WITH DISABLED ADAPTATION MECHANISM

SMALLDIVIDE			
Opponent	Trials	Goal achv.	Goal achv. (%)
Original AAI	150	59	39%
Random	150	71	47%
THERING			
Opponent	Trials	Goal achv.	Goal achv. (%)
Original AAI	150	90	60%
Random	150	76	51%
METALHECKV2			
Opponent	Trials	Goal achv.	Goal achv. (%)
Original AAI	150	70	47%
Random	150	54	36%

TABLE III
EFFECTIVENESS WITH ENABLED ADAPTATION MECHANISM

SMALLDIVIDE			
Opponent	Trials	Goal achv.	Goal achv. (%)
Original AAI	150	115	77%
Random	150	96	64%
THERING			
Opponent	Trials	Goal achv.	Goal achv. (%)
Original AAI	150	122	81%
Random	150	93	62%
METALHECKV2			
Opponent	Trials	Goal achv.	Goal achv. (%)
Original AAI	150	124	83%
Random	150	60	40%

The results reveal that when pitted against the original AAI game AI, set to play in a medium playing strength, the case-based adaptation mechanism improves significantly on the established baseline effectiveness on the map SmallDivide (77%, compared to the baseline 39%). In addition, the results reveal that when pitted against the original AAI game AI, on each of the three maps, the case-based adaptation mechanism effectively obtains a victory (77%, 81%, and 83% of the experimental runs, respectively). These results indicate that the case-based adaptation mechanism is generically effective in play against the original AAI game AI. Fig. 4 displays the obtained fitness value as a function over time of two typical experimental runs on the map SmallDivide. Fig. 5 displays the obtained median

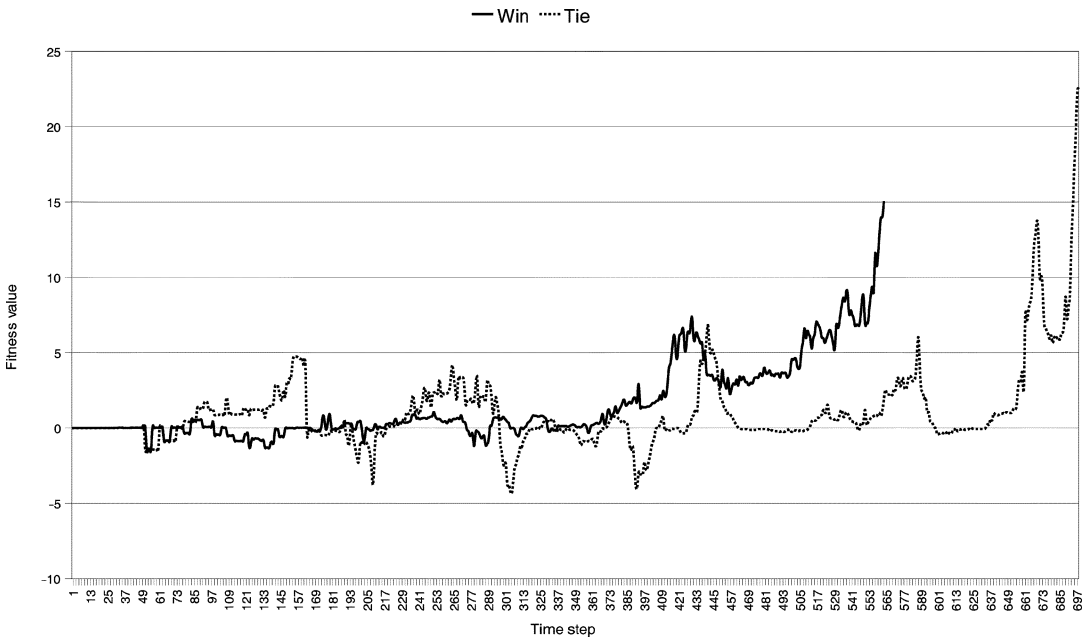

Fig. 4. Obtained fitness values as a function over time, when pitted against the original AAI game AI on the map SmallDivide. The figure displays a typical experimental result of 1) adaptation mechanism set to win the game, and 2) the adaptation mechanism set to uphold a tie.

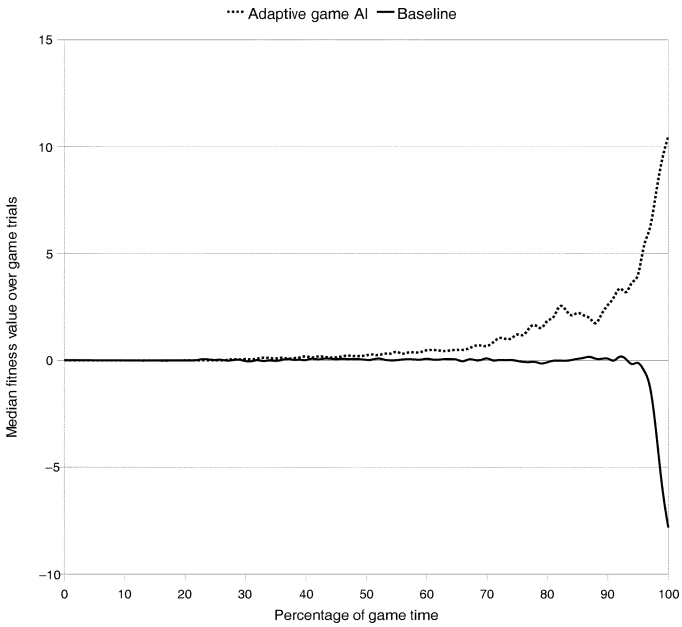

Fig. 5. Median fitness value over all game trials against the original AAI opponent on the map SmallDivide, as a function over the relative game time.

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

In addition, the results reveal that when pitted against the original AAI game AI, initialized with randomly generated strategies, the case-based adaptation mechanism improves by 17% on the established baseline effectiveness on the map SmallDivide. This improvement in effectiveness is consistent with our findings on the map TheRing, where the case-based adaptation mechanism improves by 11% on the baseline effectiveness, compared to the baseline effectiveness. In randomized play on the map MetalHeckv2, the effectiveness of the game AI improves by 4%. A precise explanation for the latter, relatively small improvement is difficult to pin down. We surmise that considerable improvement is hampered by certain low-level AI effects that, in randomized play on the map, are not influenced by adapting the high-level AI parameters. That is, while the starting positions might seem equal from the perspective of a human player, the low-level AI might be biased in being more effective at starting, for instance, at the top of the map rather than at the bottom.

TABLE IV
BASELINE EFFECTIVENESS UPHOLDING A TIE

Opponent	Trials	Time to uphold tie	Fitness variance
Original AAI	150	26.98 min (5.98 min)	1.70 (0.53)
Random	150	27.56 min (7.78 min)	1.80 (0.62)

D. Results of Difficulty Scaling

Table IV gives an overview of the baseline results of the third experiment, obtained with disabled case-based adaptation. Table V gives an overview of the results of the third experiment, obtained with case-based adaptive game AI. The first column of each table lists against which opponent the game AI was pitted. The second column lists how often the trial was repeated. The third column lists the average time to uphold a tie position, and, between brackets, the accompanying standard deviation of the obtained result. The fourth column lists the average variance in fitness value, and, between brackets, the accompanying standard deviation of the obtained result.

The experiment concerned the difficulty scaling ability of the case-based adaptive game AI. The results reveal that when

TABLE V
UPHOLDING A TIE WITH CASE-BASED ADAPTIVE GAME AI

SMALLDIVIDE			
Opponent	Trials	Time to uphold tie	Fitness variance
Original AAI	150	37.37 min (16.72 min)	1.93 (0.55)
Random	150	36.23 min (17.64 min)	1.90 (0.66)
THERING			
Opponent	Trials	Time to uphold tie	Fitness variance
Original AAI	150	18.42 min (4.16 min)	1.55 (0.40)
Random	150	21.95 min (7.27 min)	1.56 (0.44)
METALHECKV2			
Opponent	Trials	Time to uphold tie	Fitness variance
Original AAI	150	18.94 min (6.16 min)	1.80 (0.43)
Random	150	18.49 min (5.49 min)	1.89 (0.50)

pitted against the original AAI opponent, the case-based adaptive game AI improves significantly on the time in which a tie is upheld on the map SmallDivide (37.37 min, compared to the baseline 26.98 min). In addition, the results reveal that on each of the three maps the case-based adaptive game AI is capable of upholding a tie for a sustained period of time (37.37, 18.42, and 18.94 min on average, respectively). In its attempt to get close to the target fitness value, the case-based adaptive game AI obtains a relatively low variance (1.93, 1.55, and 1.80 on average, respectively). The typical result given in Fig. 4 reveals that a tie can be upheld for a sustained period of time. However, at certain point in time, inevitably, the game AI will no longer be able to compensate for the play of the opponent, and the game will either be won or lost by the player.

Comparable difficulty scaling results are obtained when the case-based adaptive game AI was pitted against opponents with randomly generated strategies. The results reveal that when pitted against opponents with randomly generated strategies, the case-based adaptive game AI improves significantly on the time in which a tie is upheld on the map SmallDivide (36.23 min, compared to the baseline 27.56 min). In addition, the results reveal that when pitted against opponents with randomly generated strategies, on each of the three maps, the case-based adaptive game AI is able to uphold a tie for a sustained period of time (36.23, 21.95, and 18.49 min on average, respectively). In its attempt to get close to the target fitness value, the case-based adaptive game AI obtains a relatively low variance (1.90, 1.56, and 1.89 on average, respectively).

VI. DISCUSSION

In the experiments that test our implementation of case-based adaptive game AI, we observed that the game AI was well able to achieve a victory when pitted against the original AAI game AI, set to play in a medium playing strength. We noticed that the case-based adaptive game AI was able to find in the case base a strategy that could effectively defeat the original AAI game AI. As the original AAI game AI is not able to adapt its behavior, the case-based adaptive game AI could exploit its discovery indefinitely. Note that in some cases, the case-based adaptive game AI did not win the game, despite it exhibiting strong behavior. Such outliers cannot be avoided due to the inherent randomness that is typical to video games. For instance, in the *Spring* game, the most powerful unit is able to destroy a commander unit with a single shot. Should the commander be destroyed in such a way, the question would arise if this was due to bad luck, or due to an effective strategy by the opponent. For game AI to be accepted as effective players, one could argue, recalling the previously mentioned need for consistent AI behavior, that game AI should not force a situation that may be regarded as the result of lucky circumstances.

In addition, we observed that even in play with randomized strategic parameter values, the case-based adaptation mechanism is generally able to find effective strategies in the case base, and was thereby able to improve on the randomized performance. This is a satisfactory result. As randomized play may be considered a simulated way to test the game AI against previously unobserved opponents, naturally, the question remains how the performance in randomized play can be further enhanced. We discuss two approaches to enhance the performance in play with randomized strategic parameter values.

First, note that for each map our case base currently consists of observations collected over 325 games. For randomized play, determined by 27 pseudorandomized behavioral parameters, it would be beneficial to collect more games in the case base in order to increase the probability of it containing effective game strategies. As case-based adaptive game AI can be expected to be applied in the play-testing phase of game development, and predictably in multiplayer games, the case base in practical applications is expected to grow rapidly to contain a multitude of effective strategies.

Second, 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 initializing the game AI with effective behavior. In order to do so, a player needs to determine accurately the opponent against whom it will be pitted. In video-game practice, (human) game opponents do not exhibit behavior as random as in our experimental setup, but will typically exhibit behavior that can be abstracted into a limited set of opponent models. Our previous research has shown that even in complex RTS games such as *Spring*, accurate models of the opponent player can be established [29]. Therefore, we will follow the expert opinion that game AI should not be focused so much on directly exploiting current game observations, but should rather focus on effectively applying models of the opponent in actual game circumstances [22].

In addition, we found the case-based adaptive game AI to be able to uphold a tie for a sustained period of time, while maintaining a relatively low variance in the targeted fitness value. This ability may be regarded as a straightforward form of difficulty scaling. If a metric can be established that represents the preferred level of challenge for the human player, then in theory the case-based adaptive game AI would be capable of scaling the difficulty level to the human player. Such a capability provides an interesting challenge for future research.

VII. Conclusion and Future Work

In this paper, we discussed an approach to adaptive game AI capable of adapting rapidly and reliably to game circumstances. Our approach can be classified in the area of 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 behavior in a controlled manner in online play. Results of experiments that test the approach on three different maps in the *Spring* game show that case-based adaptive game AI can successfully obtain effective performance, and is capable of upholding a tie for a sustained period of time. From these results, we may conclude that the proposed case-based adaptive game AI provides a strong basis for effectively adapting game AI in actual video games.

For future work, we will extend the established case-based adaptive game AI with a means to scale the difficulty level to the human player. Subsequently, we will investigate how our approach to rapidly and reliably adapting game AI can be improved by incorporating opponent models.

Appendix

In this appendix, we describe the 27 parameters of strategic behavior that were used in our experiments.

- AIRCRAFT_RATE: determines how many air units AAI will build (a value of 7 means that every seventh unit will be an air unit; a value of 1 means that constructing air units is disabled).
- AIR_DEFENSE: how often air defense 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_DEFENSES: maximum number of defenses 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_DEFENSE_DISTANCE: maximum distance to base where AAI defends metal extractors with cheap defense 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 worker-time/buildspeed of a unit to be taken into account when.
- MIN_FACTORIES_FOR_DEFENSES: AAI will not start to build stationary defenses before it has built at least that number of factories.
- MIN_FACTORIES_FOR_STORAGE: AAI will not start to build stationary defenses before it has built at least that number of storage buildings.
- MIN_FACTORIES_FOR_RADAR_JAMMER: AAI will not start to build stationary defenses 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 defenses (if it has not already reached the maximum number of defenses 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.

Acknowledgment

The authors would like to thank the anonymous referees for their constructive comments that helped to improve this paper considerably.

References

[1] P. Tozour, "The perils of AI scripting," in *AI Game Programming Wisdom*, S. Rabin, Ed. Hingham, MA: Charles River, 2002, pp. 541–547.

[2] I. Millington, "Decision making," in *Artificial Intelligence for Games*. San Francisco, CA: Morgan Kaufmann, 2006, pp. 301–471.

[3] S. C. J. Bakkes, P. H. M. Spronck, and H. J. van den Herik, "Rapid adaptation of video game AI," in *Proc. IEEE Symp. Comput. Intell. Games*, P. Hingston and L. Barone, Eds., 2008, pp. 79–86, IEEE XPlore Catalog Number: CFP08CIG-CDR.

[4] A. Nareyek, "Artificial intelligence in computer games: State of the art and future directions," *ACM Queue*, vol. 1, no. 10, pp. 58–65, Feb. 2004.

[5] L. Manovich, *The Language of New Media*. Cambridge, MA: MIT Press, 2002.

[6] L. N. Taylor, "Video games: Perspective, point-of-view, and immersion," M.S. thesis, Grad. Art School, Univ. Florida, Gainesville, FL, 2002.

[7] J. Nechvatal, "Immersive ideals/critical distances. A study of the affinity between artistic ideologies based in virtual reality and previous immersive idioms," Ph.D. dissertation, Centre Adv. Inquiry Interactive Arts (CAiiA), Univ. Wales College, Newport, Wales, U.K., 1999.

[8] R. Laursen and D. Nielsen, "Investigating small scale combat situations in real-time-strategy computer games," M.S. thesis, Dept. Comput. Sci., Univ. Aarhus, Aarhus, Denmark, 2005.

[9] L. Lidén, "Artificial stupidity: The art of making intentional mistakes," in *AI Game Programming Wisdom 2*, S. Rabin, Ed. Hingham, MA: Charles River, 2004, pp. 41–48.
[10] M. Buro and T. M. Furtak, "RTS games and real-time AI research," in *Proc. Behav. Represent. Model. Simul. Conf. Simulation Interoperability Standards Organization (SISO)*, Arlington, VA, 2004, pp. 34–41.
[11] P. Demasi and A. J. d. O. Cruz, "Online coevolution for action games," *Int. J. Intell. Games Simul.*, vol. 2, no. 3, pp. 80–88, 2002.
[12] T. Graepel, R. Herbrich, and J. Gold, "Learning to fight," in *Proc. Comput. Games, Artif. Intell. Design Edu.*, Q. H. Mehdi, N. E. Gough, and D. Al-Dabass, Eds., Wolverhampton, U.K., 2004, pp. 193–200.
[13] S. Johnson, "Adaptive AI: A practical example," in *AI Game Programming Wisdom 2*, S. Rabin, Ed. Hingham, MA: Charles River, 2004, pp. 639–647.
[14] P. H. M. Spronck, M. J. V. Ponsen, I. G. Sprinkhuizen-Kuyper, and E. O. Postma, "Adaptive game AI with dynamic scripting," *Mach. Learn.*, vol. 63, no. 3, pp. 217–248, 2006.
[15] 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. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 1041–1046.
[16] D. Aha, M. Molineaux, and M. J. V. Ponsen, "Learning to win: Case-based plan selection in a real-time strategy game," in *Proc. 6th Int. Conf. Case-Based Reason.*, Chicago, IL, 2005, pp. 5–20.
[17] M. J. V. Ponsen and P. H. M. Spronck, "Improving adaptive game AI with evolutionary learning," in *Proc. Comput. Games, Artif. Intell. Design Edu.*, Wolverhampton, U.K., 2004, pp. 389–396.
[18] S. Ontañón, K. Mishra, N. Sugandh, and A. Ram, "Case-based planning and execution for real-time strategy games," in *Proc. 7th Int. Conf. Case-Based Reason.*, Heidelberg, Germany, 2007, pp. 164–178, ISBN 978-3-540-74138-1.
[19] 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. 9th Eur. Conf. Case-Based Reason.*, Trier, Germany, 2008, pp. 59–73.
[20] S. J. Louis and C. Miles, "Playing to learn: Case-injected genetic algorithms for learning to play computer games," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 669–681, Dec. 2005.
[21] R. Baumgarten, S. Colton, and M. Morris, "Combining AI methods for learning BOTS in a real time strategy game," *Int. J. Comput. Game Technol.*, vol. 2009, 2009, Article ID 129075.
[22] S. Rabin, "Preface," in *AI Game Programming Wisdom 4*, S. Rabin, Ed. Hingham, MA: Charles River, 2008.
[23] P. H. M. Spronck, I. G. Sprinkhuizen-Kuyper, and E. O. Postma, "Difficulty scaling of game AI," in *Proc. 5th Int. Conf. Intell. Games Simul.*, A. E. Rhalibi and D. V. Welden, Eds., Ghent, Belgium, 2004, pp. 33–37.
[24] M. Buro and T. Furtak, "RTS games as test-bed for real-time AI research," in *Proc. 7th Joint Conf. Inf. Sci.*, 2003, pp. 481–484.
[25] B. Scott, "The illusion of intelligence," in *AI Game Programming Wisdom*, K. Chen, S.-H. Chen, H.-D. Cheng, D. K. Y. Chiu, S. Das, R. Duro, Z. Jiang, N. Kasabov, E. Kerre, H. V. Leong, Q. Li, M. Lu, M. G. Romay, D. Ventura, P. P. Wang, and J. Wu, Eds., S. Rabin, Ed. Hingham, MA: Charles River, 2002, pp. 16–20.
[26] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games," in *Proc. AAAI Workshop Challenges Game Artif. Intell.*, Menlo Park, CA, 2004, pp. 91–96.
[27] P. Demasi and A. J. d. O. Cruz, "Anticipating opponent behaviour using sequential prediction and real-time fuzzy rule learning," in *Proc. 4th Int. Conf. Intell. Games Simul.*, A. E. Rhalibi and D. V. Welden, Eds., Ghent, Belgium, 2004, pp. 101–105.
[28] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Appl. Artif. Intell.*, vol. 21, no. 10, pp. 933–971, 2007.
[29] F. C. Schadd, S. C. J. Bakkes, and P. H. M. Spronck, M. Roccetti, Ed., "Opponent modeling in real-time strategy games," in *Proc. 8th Int. Conf. Intell. Games Simul.*, Ghent, Belgium, 2007, pp. 61–68.
[30] S. Johansson, "The Spring Project," 2009 [Online]. Available: http://spring.clan-sy.com/
[31] 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. Hingham, MA: Charles River, 2008, pp. 647–658.
[32] R. S. Sutton, "Learning to predict by the methods of temporal differences," in *Machine Learning*. Boston, MA: Kluwer, 1988, pp. 9–44.
[33] S. C. J. Bakkes, P. H. M. Spronck, and H. J. van den Herik, "Phase-dependent evaluation in RTS games," in *Proc. 19th Belgian-Dutch Conf. Artif. Intell.*, M. M. Dastani and E. de Jong, Eds., Utrecht, The Netherlands, 2007, pp. 3–10.
[34] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 100–108, 1979.
[35] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[36] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proc. 23rd Int. Conf. Mach. Learn.*, New York, 2006, pp. 97–104.
[37] R. Evans, "Varieties of learning," in *AI Game Programming Wisdom*, S. Rabin, Ed. Hingham, MA: Charles River, 2002, pp. 571–575.
[38] A. Seizinger, AI:AAI "Creator of the Game AI 'AAI',", 2006 [Online]. Available: http://spring.clan-sy.com/wiki/AI:AAI
[39] S. Abou-Samra, C. Comair, R. Champagne, S. T. Fam, P. Ghali, S. Lee, J. Pan, and X. Li, "Data compression/decompression based on pattern and symbol run length encoding for use in a portable handheld video game system," U.S. Patent #6 416 410, 2002.
[40] F. Angiulli and G. Folino, "Distributed nearest neighbor-based condensation of very large data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 12, pp. 1593–1606, Dec. 2007.

Sander Bakkes studied knowledge engineering at Maastricht University, The Netherlands, and graduated in 2003. He is currently working towards the Ph.D. degree at the Faculty of Humanities, Tilburg University, Tilburg, The Netherlands.

He is affiliated with the Tilburg Centre for Creative Computing (TiCC). His research interests include AI (in particular in computer games), human–computer interaction, player immersion, and player satisfaction.

Pieter Spronck studied computer science at the Delft University of Technology, Delft, The Netherlands, and graduated *cum laude* in 1996. He received the Ph.D. degree in 2005 with a dissertation titled "Adaptive game AI."

Currently, he is an Associate Professor at the Faculty of Humanities, Tilburg University, Tilburg, The Netherlands. He is affiliated with the Tilburg Centre for Creative Computing (TiCC). His research interests include AI (in particular in computer games), machine learning, intelligent agents, and knowledge technology.

Jaap van den Herik studied mathematics at the VU University Amsterdam, The Netherlands, and graduated *cum laude* in 1974. He received the Ph.D. degree in 1983 with a dissertation titled "Computerschaak, Schaakwereld en Kunstmatige Intelligentie."

Currently, he is a Full Professor of Computer Science at the Faculty of Humanities, Tilburg University, Tilburg, The Netherlands. He is the Director of the Tilburg Centre for Creative Computing (TiCC). In addition, he is a Full Professor of Law and Computer Science at the Faculty of Law, Universiteit Leiden, Leiden, The Netherlands. His research interests include agents, AI, computer networks, computer games, game theory, and knowledge management.