A review of computational intelligence in RTS games

Raúl Lara-Cabrera, Carlos Cotta and Antonio J. Fernández-Leiva

***Abstract*—Real-time strategy games offer a wide variety of fundamental AI research challenges. Most of these challenges have applications outside the game domain. This paper provides a review on computational intelligence in real-time strategy games (RTS). It starts with challenges in real-time strategy games, then it reviews different tasks to overcome this challenges. Later, it describes the techniques used to solve this challenges and it makes a relationship between techniques and tasks. Finally, it presents a set of different frameworks used as test-beds for the techniques employed. This paper is intended to be a starting point for future researchers on this topic.**

***Index Terms*—Computational intelligence, real-time strategy games, review.**

I. Introduction

Commercial video-games are a rising value in the entertainment industry. The total spent in the video-game industry in 2010 was 25.1 billion dollars [15]. Video-game budgets are high and their development teams are composed of many people. Traditionally, game developers have overlooked their non-player characters' artificial intelligence, focusing to other aspects of the game, such as graphic engines and 3-D modeling. This situation leads to a poor gaming experience, since human players are able to win the game without much effort. Computational intelligence is growing in importance and video-game players are demanding good artificial intelligence (AI) which makes these games more interesting and harder to beat.

RTS games are a genre of video-games which require managing different kind of units and resources in real-time. In a RTS game the participants position and maneuver units and structures under their control to secure areas of the map and/or destroy their opponents' assets. In a typical RTS, it is possible to create additional units and structures during the course of a game, but this is generally limited by the number of accumulated resources, which are gathered by controlling special points on the map and/or possessing certain types of units and structures devoted to this purpose. The typical game of the RTS genre features resource gathering, base building, in-game technological development and indirect control of units. They are usually played by two or more players (human or not) that have to deal with incomplete information during the game (the map is covered by fog of war, the technology developed by a player is unknown by every other player, ...). These features make RTS games a great tool for computational intelligence research, since a RTS game player needs to master many challenging problems such as resource allocation, spatial reasoning, strategy planning and opponent's strategy prediction. In addition, procedural content generation can be used to create maps, units and technologies for RTS games. Traditionally, academic game AI was mainly linked to non player character (NPC) behavior and pathfinding. However, there are new research areas that have recently provided innovative solutions for a number of game development challenges, like player experience modeling (PEM), procedural content generation (PCG) and large scale game data mining [86].

Real-time strategy and turn-based games (RTS and TBS respectively) are sub-genres of strategy games. They have a lot of aspects in common. Many of the proposed challenges and tasks could be applicable to RTS and TBS without distinction. However, aspects related to real-time (i.e. adversarial real-time planning) are only applicable to RTS.

Computational intelligence in RTS video games is quite a new field of research (although there are old papers on the use of evolutionary algorithms for real time adversarial gaming [6], [19]), as opposed to the computational intelligence in TBS and board games like *Chess* and *Go*. Since this is a recent topic, there are no previous reviews. This is the main motivation behind this paper.

II. Challenges in Real-time Strategy games

As we briefly noted before, RTS games offer a large variety of fundamental AI research problems [7]. In this section, we describe these research challenges and its relationships with RTS games.

A. Adversarial real-time planning

As its name suggests, RTS game's actions are made in real-time, so players have to make their decisions under severe time constraints and they need to be able to execute multiple orders simultaneously. In addition to this, games take place in dynamic and hostile environments that contains adversaries who modify the game state asynchronously. These characteristics denote the need to investigate adversarial real-time planning approaches. Turn-based games' (like Chess) agents have to deal with dynamic and hostile environments as well, but the fact of the game not happening in real time makes this challenge less difficult to overcome than the same challenge in a real-time environment.

B. Decision making under uncertainty

In most commercial RTS games, NPCs have all the information on the game state, including the location of the human player units and buildings. This situation should be considered as cheating, because players lack this information. To prevent this unbalanced situation, it is necessary to impose partial observability onto RTS games, ensuring that all players play the game on equal terms. This partial observability,

The three authors are with Department "Lenguajes y Ciencias de la Computación", ETSI Informática, University of Málaga, Spain (emails: {raul,ccottap,afdez}@lcc.uma.es)

usually named "fog of war", represents another challenge to the design of game agents, because players are not aware of the enemies base locations and intentions. This challenge is not present when designing an agent for a turn-based or board game like Chess and Go, whose player usually have all the information on the game state, including the location of other players units (there are, however, turn-based games whose state is partially known by the players).

C. Opponent modeling

Human players have an ability to analyze the enemy's actions and spot their weaknesses, exploiting them in future games. Artificial players need to be qualified to analyze their opponents' actions and predict their behaviors in basis of previous observations. This challenge is not exclusive of RTS games, because in any other game, modeling your opponent will be always useful.

D. Spatial and temporal reasoning

Maps in RTS games are elements with high influence during the course of the game (i.e. different types of terrain produce different types of resources, elevated positions give advantages to a unit's attack, ...), so it becomes necessary to make good terrain analysis. This way, agents can develop better offensive and defensive plans and more efficient resources gathering as well. Another advantage (which can be seen as another challenge) of human players against artificial ones is their ability to understand temporal relations of actions.

E. Resource management

RTS games usually include resources which are used to create or upgrade units and buildings, and develop new technologies as well. These resources can be distributed through the map and then gathered by units or they can be obtained from buildings placed on certain places on the map. A proper resource management strategy is therefore an essential part of any successful strategy.

F. Collaboration

During a RTS game, each player generate many units for her army. This army can be considered as a multi-agent system, so it is necessary to develop coordination methods that lead to good team tactics and strategies. There is not only collaboration at unit level, in team matches there are two or more teams of players (human or artificial) which fight each against other. This kind of collaboration between players has to be taken in account to be successful at team matches.

G. Pathfinding

Finding suitable paths on a quick manner between two locations on a map is of great importance in RTS games. Since the game environment is dynamic, it contains many moving objects that have to be taken in account when calculating paths. In addition to these moving objets, there are more aspects to deal with, like keeping unit formations, taking terrain properties or enemy influence, among other. All these aspects greatly complicates the problem of finding suitable paths.

H. Content Generation

Game content refers to all aspects of the game that affect game-play other than non-player character (NPC) behavior and the game engine itself. When it comes to RTS games, there is assorted content such as maps, units, buildings and weapons that can be generated in a procedural manner. If new content can be generated with enough variety then it may become possible to create endless games, with new maps, units and buildings on every new game. In addition to this, the generated content can adapt itself to specific criteria, such as the playing style of a particular player [76].

III. Tasks in Real-time Strategy games

Due to their characteristics, RTS games give us many challenges to deal with. The work done in computational intelligence in RTS games can be classified by the problem tackled. In the following section we will describe the tasks associated with these challenges and what work has been done in relation to each of these tasks.

A. Planning

Humans and adversaries can use any available action to form their game strategy, which is a plan. Planning is the process of determining action sequences that when executed accomplish a given goal. The presence of adversaries in addition to real-time and hidden information constraints greatly complicates the planning process.

Aha, Molineaux and Ponsen introduced in [1] a plan retrieval algorithm which uses three key sources of domain knowledge and removes the assumption of a static opponent. This algorithm was called *Case-based Tactician (CaT)*. This same algorithm was used in [51], where the authors focused on defeating a selected opponent while training up others, instead of defeating randomly selected opponents. The authors in [52] introduced an integrated *RL/CBR* algorithm that uses continuous models instead of discrete approximation of these models. The algorithm was called the *Continuous Action and State Space Learner (CASSL)*, and is an improvement of the results obtained in [1], including the ability to learn and reason with continuous action spaces. An improvement for plan retrieval can be found in [50]. This improve was made by introducing the concept of situation (high-level representation of the state of the world) into the algorithm. This technique represents a knowledge based approach for feature selection for improving the performance of case retrieval in case-based reasoning systems. In [67] an architecture for learning transfer was presented, so knowledge acquired previously is used to improve the performance of the artificial player in subsequent games. This architecture, called CAse-Based Reinforcement Learner (CARL), provides a useful task decomposition, allowing the agent to learn tactical policies that can be reused across different problem instances with similar characteristics. In [59], [60] the authors proposed to extract behavioral knowledge from expert demonstrations and reused

them as a case based behavior generator. They also presented a case-based planning framework for RTS games.

In [2] PDDL was also used to define a planning domain that can be used to implement an artificial player based on automated planning in a RTS game. In [9], the authors developed an online planner for resource production in the game *Wargus*. In [55], Muñoz-Avila and Aha used hierarchical task networks (HTN) as a planning mechanism for an artificial player of the game *Stratagus*. Another HTN as a planning mechanism was used in [42]. This planner was designed according to the balanced build policy which seeks a balance between acquiring resources and producing buildings and units. In [41], Kovarsky and Buro focused on build-order optimization, instead of strategy planning. Their aim was to optimize the gathering of resources and the creation of buildings and units in the initial stage of the game. The planning domain definition language (PDDL) was used. A wall-building (or other passive defensive buildings) algorithm for RTS games was described in [26]. In [83] a goal-directed approach was used to develop agents that reason about their goals in response to unanticipated game events. In [53], the authors extended online planning with a conceptual model of goal-driven autonomy, in which an agent reasons about its goals, identifies when they need to be updated, and changes or adds to them as needed for subsequent planning and execution. Agents using this technique can competently respond to unexpected events in complex environments, producing a significant increase in performance. The authors of [80] proposed a machine learning approach to establish effective game strategies based on the structure of the environments of the game.

A genetically evolved *Planet Wars* non-player character was developed in [16], [17], [54]. The genetic algorithm was used to tune a set of parameters for the decision engine of the NPC. In [36] the performance of an artificial player was improved by using a speciated evolutionary algorithm (based on NEAT) for an optimal strategy selection. In [43] gene expression programming was used to evolve a player for a gathering resources game. In [46], Miles, Louis and Cole define a system which learns general routing information from a human player and they used case-injected evolutionary algorithms to incorporate this acquired knowledge into subsequent planning. This case injection effectively biases the evolutionary algorithm toward producing plans that contain important strategic elements used by human players. The same approach was used in [47], where the improvement of the response time of a case-injected algorithm was shown. Another stochastic method, called *Stochastic Plan Optimization*, was presented in [79] and used for finding and improving plans. Another co-evolution approach was used in [49], where the use of evolutionary algorithms to co-evolve AI players for RTS games was investigated. This technique [62] was combined with an evolutionary algorithm which evolves the knowledge bases for the dynamic scripting. These evolved knowledge bases improve the performance of dynamic scripting against static opponents in the game Wargus.

In [11], Chung, Buro and Schaeffer presented *MCPlan*, a framework for Monte Carlo planning. They identified its performance parameters and showed the results of its implementation. This algorithm was applied to simple "capture the flag" scenarios and showed promising initial results. Another Monte Carlo planning algorithm, called *UCT*, was described in [4]. The algorithm was adapted from the context of board games to the context of multi-agent tactical planning and, across a set of 12 scenarios in the game of Wargus, UCT is a top performer compared to a variety of baselines bots and a human player. Moreover, MOCART-CGA [56] is another Monte Carlo method that deals with the path planning problem in RTS games. In [64], Sailer, Buro and Lanctot presented a planning framework that uses strategy simulation in conjunction with Nash-equilibrium strategy approximation. It was applied to an army-deployment problem.

Dynamic scripting [70] is a reinforcement learning technique designed for creating adaptive video game agents. It employs on-policy value iteration to optimize state-action values based solely on a scalar reward signal. This technique [62] was combined with an evolutionary algorithm which evolves the knowledge bases for the dynamic scripting. These evolved knowledge bases improve the performance of dynamic scripting against static opponents in the game Wargus. An extension to the dynamic scripting algorithm can be found in [12]. This extension is a goal-directed approach called *GoHDS*. Goals are used as domain knowledge for selecting rules, and a rule is seen as a strategy for achieving a goal. In [44] a tactical abstract game framework was described and used to evaluate an extended version of the dynamic scripting algorithm.

A method for evolving increasingly complex artificial neural networks in real time, called *rtNEAT*, was introduced in [71] by Stanley, Bryant and Miikkulainen and used later in [78].

B. Unit maneuvering (micro management)

RTS games management can be splitted into two levels: macro management (taking strategic decisions such as which building has to be created next or which map zone has to be scouted) and micro management. Unit formation planning and target of attack is the core of micro management in RTS games.

The work in [27] focused on micro-management of the units. They designed and implemented a CBR/RL hybrid system for learning which enemy units to target in given situations during a battle in an RTS game, as well as in [84] where a similar learning approach was used also in micro-management. In [3], the authors proposed a method which consists on each unit acting independently of the team and having its own influence map (IM). This way, they achieved team coordination while evolving all entities' IM parameters together. A similar evolutionary approach was described in [39] and [40].

Hagelbäck and Johansson postulated in [29], [30], [32], [33] the use of potential fields and multi-agent schemes for

the maneuvering of real-time strategy bots and dealing with the partial observability of this genre of game, named fog of war.

The intelligent moving and path-finding of units were investigated in [13]. The authors obtained smooth and natural movements of units combining flocking with IM pathfinding and improving the performance of the units in every game situation. Additionally, in [35] Jang and Cho proposed a strategy generation method to produce a neural artificial player with layered influence maps. In [48], a system with influence maps and trees was developed in the context of a tactical game, achieving more coordinated behaviors between the units. In [57], the authors described a method for unit formation planning. In this case, they applied potential field, fuzzy measure and integral to perform a solution on micro management. Their AI bot was able to divide the units into sub-groups and perform this unit formation planning. There are other papers [63], [81] that focused on units grouping and dynamic formations. In [73], the authors proposed controlling Starcraft units with a Bayesian model, outperforming the original AI as well as other bots (tied with the winner of AIIDE 2010 StarCraft competition). In [85], knowledge-rich player agents were developed. The authors connected the game engine with a SOAR cognitive architecture to improve its performance.

C. Plan recognition and predictions

Plan recognition refers to the act of an agent observing the actions of another agent whether it be human or computer-based with the intent of predicting its future actions, intentions, or goals. An introduction to case-based plan recognition for real-time strategy games was presented in [10]. Continuing on case-based plan recognition, collected data on building construction sequence can be used to analyze and categorize player strategies and playing styles [34] (collected data are replays of StarCraft games in this case). Ninety percent of these replays were used to train a CBR decision system, and the remaining ten percent were used to verify the predicting accuracy of the fully trained decision system. A similar approach was presented in [14], [25], where probabilistic models of opponent behavior and actions were learned from sets of saved games. Plan recognition can be separated into two levels, strategic and tactical [37]. Strategic plans dictate what kind of units the player will produce and if she will play aggressively or defensively, while tactical plans dictate how units are deployed and used. Another approach was described in [65] and [66]. Hierarchical structured models were used for opponent modeling. Two different classifiers were evaluated in order to test the effectiveness of this approach: fuzzy models and discounted rewards from game theory. [72], [74] presented a Bayesian model to predict the opening (rst strategy) of opponents. The model is general enough to be applied to any RTS game with the canonical gameplay of gathering resources to extend a technology tree and produce military units. The model can also predict the possible technology trees of the opponent. In [82] Weber and Mateas presented a data mining approach to opponent modeling. Machine learning techniques were applied to large collections of saved games. These techniques provide the ability to detect an opponent's strategy before it is executed and predict when an opponent is to perform strategic actions.

D. Procedural content generation

As defined in [75], procedural content generation (PCG) refers to the automatic or semi-automatic generation of game content. In this paper, the authors used a multi-objective evolutionary algorithm to evolve complete maps for Starcraft. This method is useful for automatic and machine-assisted map generation. Other methods for map generation were described in [68] and in [45]. Related to this, and from a more general perspective, Frade et al. introduced the idea of terrain programming, namely the use of genetic programming to evolve playing maps for video-games, using either subjective human-based feedback [20], [21] or automated quality measures such as accessibility [22] or edge-length [23]. A taxonomy of Procedural Content Generation (PGC) algorithms can be found in [76], [87].

E. Partial observability

Although dealing with partial observability is included in almost every planning paper reviewed before, there are some papers focused on this task. In [28], the authors presented a modified potential field bot that handles imperfect information about the game world (namely fog of war). The effect of imposing partial observability onto an RTS game with regard to making predictions was shown in [8]. The authors compared two different mechanisms that decide where best to direct the attention of the observers to maximize the benefit of predictions.

F. Opponent matching

As in partial observability, opponent matching is often included in planning papers, but there are other that focuses on this task. In [77] the authors used a evolutionary algorithm to evolve a set of artificial neural networks which functions as a controller in deciding what type of unit should be created based on the enemy units. The experimentation results showed clearly a group of mixed randomized opponent can be defeated by the generated AI army. A simple and effective system of self-organizing maps for defending group selection was shown in [5]. The authors solved the problem of finding a suitable group of fighting units to combat incoming enemy groups.

G. Difficulty adjustment

In [31] the authors studied the feelings of players after playing some games against multiple kinds of opponents. The players found it more enjoyable to play an even game against an opponent that adapts to the performance of the player, than playing against an opponent with static difficulty. The neuro-evolution methodologies *NEAT* and *rtNEAT* were used in [58] to generate opponents who match the skill of players in real-time. In [24] the authors described a method for the automatic generation of virtual players that adapt to

the player skills. This was done by building initially a model of the player behavior in real time during the game, and further evolving the virtual player via this model in-between two games.

IV. Techniques used in RTS games

In the previous sections, we have presented the challenges and tasks that have been studied in relation to computational intelligence in RTS games. Next, we will describe what techniques have been used to overcome these challenges.

A. Evolutionary algorithms and stochastic optimization

Evolutionary algorithms and stochastic optimization are widely used in computational intelligence in RTS games. They are often combined with other techniques to improve the performance [62]. In [46] and [47], evolutionary algorithms were used to learn to play strategic games, combined with case-injection to improve the response time. In [81], the authors used stochastic optimization as a learning algorithm. Another use of evolution in machine learning was presented in [71], where complex artificial neural networks were evolved in real time, as the game was being played. The same algorithm, *rtNEAT*, was used in [58] to adjust the opponent difficulty level dynamically. Evolutionary algorithms were used in [79] for plan optimization. The search was initialized with expert plans to improve the performance of the optimization. An analysis of the fitness landscape of an abstract RTS game can be found in [38]. In [36] a speciated evolutionary algorithm was used to improve the performance of non player characters, while in [43] gene expression programming was used to evolve a player for a RTS game. In [39] the authors presented an analysis of evolved strategies. A set of ten strategies evolved in a single environment were compared to a second set of ten strategies evolved across a set of environments. In [16], [17], [54] the authors used evolutionary techniques to optimize the parameters of a bot, as well as in [61]. Evolutionary search was used in [75] to generate suitable *Starcraft* maps. There are other PCG related papers that used evolutionary algorithms for content generation: [20], [21], [22], [23], [45]. In [77], artificial neural networks were evolved using evolutionary algorithms, while in [24], virtual players were evolved using evolutionary algorithms.

In [35], [48], evolutive techniques were combined with influence maps. The co-evolution of these influence maps was introduced in this approach. Another co-evolution approach was used in [49], where the use of evolutionary algorithms to co-evolve AI players for RTS games was investigated. In [69] a co-evolutionary algorithm was used to generate spatially oriented tactics. Students can learn from non player characters who use these co-evolved tactics. In [40] evolutionary computation techniques were used to develop an automated player that uses a progressive refinement planning technique. This automated player was co-evolved and analyzed. Co-evolution was also used in [3] to generate coordinating team tactics for a RTS game.

B. Case-based reasoning/Reinforcement learning

Case-based techniques are mainly used for plan selection in RTS games, as shown in [1], [50], [51], [59], [60], [80]. In [34] the authors analysed saved games of *Starcraft* to evaluate human-player behaviors and construct and intelligent system. This system was trained using a case-based reasoning approach. Algorithms which combine case-based reasoning with reinforcement learning were presented in [52] and [27]. In the first case, the algorithm, called *Continuous Action and State Space Learner (CASSL)*, used continuous models instead of discrete models. In the second case, the algorithm focused on learning micro-management tasks. The same approach of combining case-based reasoning and reinforcement learning was shown in [67], where a multi-layered architecture, named *CAse-Based Reinforcement Learner (CARL)*, was presented. In the context of opponent modeling, a case-based plan recognition method was presented in [10].

C. Influence and potential maps

Influence maps were combined with evolutionary techniques in [3], [35], [48], [69]. This technique was used also in [13], where the authors dealt with intelligent moving of unit groups and intelligent team composition and maneuvering. In the context of opponent modeling, influence maps appeared in [37] and they were used to recognize opponent behavior. Potential fields were used in [28] to deal with the partial observability in RTS games, specifically with the fog of war in ORTS (Open Real-Time Strategy). The use of potential fields in real time strategy bots was discussed and promoted in [29], [30]. This approach was employed in [32] to create a non player character. In [57], potential fields were used for unit formation planning.

D. AI planners, PDDL and Hierarchical Tasks Networks

In [55] the authors described a representation for explanations in the context of hierarchical case-based planning and detailed four types of explanations. An online planner for resource production was developed in [9] (another online planner can be found in [53]). In [41] another planner was presented. This time, the *Planning Domain Definition Language (PDDL)* was used to define the domain. A hierarchical task network planner was developed in [42]. This planner focused on the strategic level (buildings, units and resource management). A planner for defensive buildings was presented in [26]. In [2] PDDL was also used to define a planning domain. Weber, Mateas and Jhala [83] presented a reactive planning implementation of the Goal-Driven Autonomy conceptual model.

E. Simulations

Monte-Carlo methods were employed in [4], [11] as planning algorithms. On the first work, the authors defined a modified Monte Carlo planning algorithm, called UCT, which extends recent algorithms for bandit problems to sequential decision problems while retaining the strong theoretical performance guarantees. On the second, they presented a framework for Monte Carlo planning called

Table I
TASKS AND TECHNIQUES

	Planning	Unit maneuvering	Opponent Matching	Partial observability	Plan recognition	PCG	Difficulty Adjustment
CBR/RL	[1], [50], [51], [52], [59], [60], [67], [80]	[27], [84]			[10], [25], [34]		
AI Planning	[2], [9], [26], [41], [42], [53], [55], [83]						
Influence and Potential Maps		[3], [13], [29], [30], [32], [33], [35], [48], [57]		[28]	[37]		
Evolutionary	[16], [17], [36], [43], [46], [47], [49], [54], [62], [79], [71]	[3], [35], [39], [40], [48], [61], [81]	[77]			[20], [21], [22], [23], [45], [75]	[24], [58]
Simulations	[4], [11], [56], [64]			[8]			
Dynamic scripting	[12], [44], [62], [70]						
ANN	[71], [78]	[63]	[5], [77]				[58]
Fuzzy/Bayessian models		[73]			[65], [66], [72], [74]		

MCPlan. Sailer, Buro and Lanctot [64] presented a planning framework that uses strategy simulation in conjunction with Nash-equilibrium strategy approximation. They applied this framework to an army deployment problem in a real-time strategy game setting and presented experimental results that indicate a performance gain over the scripted strategies that the system is built on.

F. Dynamic scripting

As defined on the previous section, dynamic scripting [70] is a reinforcement learning technique designed for creating adaptive video game agents. It employs on-policy value iteration to optimize state-action values based solely on a scalar reward signal. In [12] the authors suggested a goal-directed hierarchical dynamic scripting approach for incorporating learning into RTS games. The same approach was presented in [44] where dynamic scripting was extended to improve the performance. A combination of dynamic scripting and evolutionary algorithm was used in [62].

G. Fuzzy/Bayessian models

In [73] and [72] the authors presented Bayesian models for controlling the units and for opening prediction in *Starcraft*, respectively. In the context of opponent modeling, fuzzy models were used in [65], [66].

H. Other techniques

There are other techniques used to overcome challenges that have been previously presented. Self-organizing maps can be found in [63] and [5]. There are also papers that focused on artificial neural networks, like [58], [71], [77]; and hidden Markov models [14]. Data mining was used in [82] while a cognitive architecture (SOAR) was used in [85]. In [18], the authors presented a method to determine which strategy to use depending on what kind of map the controller is playing trough map characterization.

V. CONCLUSION

We have presented a review about the research in computational intelligence applied to real-time strategy games. This paper aims to be a starting point in this research topic, helping the reader to understand the application of computational intelligence to video-games, specifically real-time strategy games.

This review shows us that the main challenges tackled in this research topic are player and opponent modeling, while the most used techniques to overcome these challenges are evolutionary algorithms, stochastic optimization, case-based techniques, influence maps and probabilistic methods.

Many techniques and algorithms have been described in this paper. In the near future, we are going to combine these techniques into hybrid and interactive algorithms to improve the performance obtained from using these techniques separately. This is one of the goals of the *DNEMESIS* project.

ACKNOWLEDGEMENTS

This work is partially supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project P10-TIC-6083 (DNEMESIS).

REFERENCES

[1] D. W. Aha, M. Molineaux, and M. J. V. Ponsen, "Learning to win: Case-based plan selection in a real-time strategy game," in *International Conference on Case-Based Reasoning*, ser. Lecture Notes in Computer Science, H. Muñoz-Avila and F. Ricci, Eds., vol. 3620. Springer, 2005, pp. 5–20.

[2] V. Alcázar, D. Borrajo, and C. Linares López, "Modelling a RTS planning domain with cost conversion and rewards," in *Artificial Intelligence in Games. Workshop of the Eighteenth European Conference on Artificial Intelligence*, Patras, Greece, 2008, pp. 50–54.

[3] P. Avery and S. J. Louis, "Coevolving team tactics for a real-time strategy game," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[4] R.-K. Balla and A. Fern, "UCT for tactical assault planning in real-time strategy games," in *International Joint Conference on Artificial Intelligence*, C. Boutilier, Ed., 2009, pp. 40–45.

[5] N. Beume, T. Hein, B. Naujoks, N. Piatkowski, M. Preuss, and S. Wessing, "Intelligent anti-grouping in real-time strategy games," in *IEEE Symposium on Computational Intelligence and Games*, P. Hingston and L. Barone, Eds. IEEE, 2008, pp. 63–70.

[6] G. H. Burgin and L. J. Fogel, "Air-to-air combat tactics synthesis and analysis program based on an adaptive maneuvering logic," *Journal of Cybernetics*, vol. 2, no. 4, pp. 60–68, 1972.

[7] M. Buro, "RTS games and real-time AI research," in *Behavior Representation in Modeling and Simulation Conference*, vol. 1. Curran Associates, Inc., 2004.

[8] S. Butler and Y. Demiris, "Partial observability during predictions of the opponent's movements in an RTS game," in *IEEE Conference on Computational Intelligence and Games*, G. N. Yannakakis and J. Togelius, Eds. IEEE, 2010, pp. 46–53.

[9] H. Chan, A. Fern, S. Ray, N. Wilson, and C. Ventura, "Online planning for resource production in real-time strategy games," in *International Conference on Automated Planning and Scheduling*, M. S. Boddy *et al.*, Eds. The AAAI Press, 2007, pp. 65–72.

[10] D. Cheng and R. Thawonmas, "Case-based plan recognition for real-time strategy games," in *GameOn Conference*, A. El-Rhalibi and D. van Welden, Eds. EUROSIS, 2004, pp. 36–40.

[11] M. Chung, M. Buro, and J. Schaeffer, "Monte Carlo Planning in RTS Games," in *IEEE Symposium on Computational Intelligence and Games*. IEEE, 2005.

[12] A. Dahlbom and L. Niklasson, "Goal-directed hierarchical dynamic scripting for RTS games," in *Artificial Intelligence and Interactive Digital Entertainment*, J. E. Laird and J. Schaeffer, Eds. The AAAI Press, 2006, pp. 21–28.

[13] H. Danielsiek, R. Stür, A. Thom, N. Beume, B. Naujoks, and M. Preuss, "Intelligent moving of groups in real-time strategy games," in *IEEE Symposium on Computational Intelligence and Games*, P. Hingston and L. Barone, Eds. IEEE, 2008, pp. 71–78.

[14] E. W. Dereszynski, J. Hostetler, A. Fern, T. G. Dietterich, T.-T. Hoang, and M. Udarbe, "Learning probabilistic behavior models in real-time strategy games," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, V. Bulitko and M. O. Riedl, Eds. The AAAI Press, 2011.

[15] Entertainment Software Association and Others, "Essential facts about the computer and video game industry," 2011. [Online]. Available: http://www.theesa.com/facts/pdfs/ESA_EF_2011.pdf

[16] A. Fernández-Ares, A. M. Mora, J. J. Merelo Guervós, P. García-Sánchez, and C. M. Fernandes, "Optimizing player behavior in a real-time strategy game using evolutionary algorithms," in *IEEE Congress on Evolutionary Computation*. IEEE, 2011, pp. 2017–2024.

[17] ——, "Optimizing strategy parameters in a game bot," in *International Work-Conference on Artificial Neural Networks*, ser. Lecture Notes in Computer Science, J. Cabestany *et al.*, Eds., vol. 6692. Springer, 2011, pp. 325–332.

[18] A. Fernndez-Ares, P. Garca-Snchez, A. M. Mora, and J. J. Merelo, "Adaptive bots for real-time strategy games via map characterization," in *Computational Intelligence and Games*. IEEE, 2012, pp. 417–423.

[19] L. Fogel and G. Burgin, "Competitive goal-seeking through evolutionary programming." DTIC Document, Tech. Rep., 1969.

[20] M. Frade, F. F. de Vega, and C. Cotta, "Modelling video games' landscapes by means of genetic terrain programming - a new approach for improving users' experience," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science, M. Giacobini *et al.*, Eds., vol. 4974. Springer, 2008, pp. 485–490.

[21] ——, "Breeding terrains with genetic terrain programming: The evolution of terrain generators," *International Journal of Computer Games Technology*, vol. 2009, 2009.

[22] ——, "Evolution of artificial terrains for video games based on accessibility," in *Applications of Evolutionary Computation 2010*, ser. Lecture Notes in Computer Science, C. D. Chio *et al.*, Eds., vol. 6024. Springer-Verlag, 2010, pp. 90–99.

[23] ——, "Evolution of artificial terrains for video games based on obstacles edge length," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–8.

[24] J. A. García Gutiérrez, C. Cotta, and A. J. Fernández Leiva, "Design of emergent and adaptive virtual players in a war RTS game," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, ser. Lecture Notes in Computer Science, J. M. Ferrández *et al.*, Eds., vol. 6686. Springer, 2011, pp. 372–382.

[25] W. Gong, E. Lim, P. Achananuparp, F. Zhu, D. Lo, and F. Chong Tat Chua, "In-game action list segmentation and labeling in real-time strategy games," in *Computational Intelligence and Games*. IEEE, 2012, pp. 147–154.

[26] M. Grimani, "Wall Building for RTS Games," in *AI Game Programming Wisdom 2*. Hingham, Massachusetts: Charles River Media, Inc., 2004, pp. 425–437.

[27] M. Gunnerud, "A CBR/RL system for learning micromanagement in real-time strategy games," Master's thesis, Norwegian University of Science and Technology, 2009.

[28] J. Hagelbäck and S. J. Johansson, "Dealing with fog of war in a real time strategy game environment," in *IEEE Symposium on Computational Intelligence and Games*, P. Hingston and L. Barone, Eds. IEEE, 2008, pp. 55–62.

[29] ——, "The rise of potential fields in real time strategy bots," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, C. Darken and M. Mateas, Eds. The AAAI Press, 2008.

[30] ——, "Using multi-agent potential fields in real-time strategy games," in *Autonomous Agents and Multiagent Systems*, L. Padgham *et al.*, Eds. IFAAMAS, 2008, pp. 631–638.

[31] ——, "Measuring player experience on runtime dynamic difficulty scaling in an RTS game," in *IEEE Symposium on Computational Intelligence and Games*, P. L. Lanzi, Ed. IEEE, 2009, pp. 46–52.

[32] ——, "A multi-agent potential field-based bot for a full RTS game scenario," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, C. Darken and G. M. Youngblood, Eds. The AAAI Press, 2009.

[33] J. Hagelbck, "Potential-field based navigation in StarCraft," in *Computational Intelligence and Games*. IEEE, 2012, pp. 388–393.

[34] J.-L. Hsieh and C.-T. Sun, "Building a player strategy model by analyzing replays of real-time strategy games," in *International Joint Conference on Neural Networks*. IEEE, 2008, pp. 3106–3111.

[35] S.-H. Jang and S.-B. Cho, "Evolving neural NPCs with layered influence map in the real-time simulation game 'Conqueror'," in *IEEE Symposium on Computational Intelligence and Games*, P. Hingston and L. Barone, Eds. IEEE, 2008, pp. 385–388.

[36] S.-H. Jang, J. Yoon, and S.-B. Cho, "Optimal strategy selection of non-player character on real time strategy game using a speciated evolutionary algorithm," in *IEEE Symposium on Computational Intelligence and Games*, P. L. Lanzi, Ed. IEEE, 2009, pp. 75–79.

[37] F. Kabanza, P. Bellefeuille, F. Bisson, A. R. Benaskeur, and H. Irandoust, "Opponent behaviour recognition for real-time strategy games," in *Plan, Activity, and Intent Recognition*, ser. AAAI Workshops, vol. WS-10-05. The AAAI Press, 2010.

[38] D. Keaveney and C. O'Riordan, "Analysing the fitness landscape of an abstract real-time strategy game," in *GameOn Conference*, V. J. Botti *et al.*, Eds. EUROSIS, 2008, pp. 51–55.

[39] ——, "Evolving robust strategies for an abstract real-time strategy game," in *IEEE Symposium on Computational Intelligence and Games*, P. L. Lanzi, Ed. IEEE, 2009, pp. 371–378.

[40] ——, "Evolving coordination for real-time strategy games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 155–167, 2011.

[41] A. Kovarsky and M. Buro, "A First Look at Build-Order Optimization in Real-Time Strategy Games," in *GameOn Conference*, L. Wolf and M. Magnor, Eds. EUROSIS, 2006, pp. 18–22.

[42] J. Laagland, "A HTN planner for a real-time strategy game." [Online]. Available: http://hmi.ewi.utwente.nl/verslagen/capita-selecta/CS-Laagland-Jasper.pdf

[43] P. Lichocki, K. Krawiec, and W. Jaskowski, "Evolving teams of cooperating agents for real-time strategy game," in *EvoWorkshops*, ser. Lecture Notes in Computer Science, M. Giacobini *et al.*, Eds., vol. 5484. Springer, 2009, pp. 333–342.

[44] J. Ludwig and A. Farley, "Examining extended dynamic scripting in a tactical game framework," in *Artificial Intelligence and Interactive Digital Entertainment*, C. Darken and G. M. Youngblood, Eds. The AAAI Press, 2009.

[45] T. Mahlmann, J. Togelius, and G. N. Yannakakis, "Spicing up map generation," in *EvoApplications*, ser. Lecture Notes in Computer Science, C. D. Chio *et al.*, Eds., vol. 7248. Springer, 2012, pp. 224–233.

[46] C. Miles, S. Louis, N. Cole, and J. McDonnell, "Learning to play like a human: case injected genetic algorithms for strategic computer

gaming," in *Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1441–1448.

[47] C. Miles and S. J. Louis, "Case-injection improves response time for a real-time strategy game," in *IEEE Symposium on Computational Intelligence and Games*. IEEE, 2005.

[48] ——, "Towards the co-evolution of influence map tree based strategy game players," in *IEEE Symposium on Computational Intelligence and Games*, S. J. Louis and G. Kendall, Eds. IEEE, 2006, pp. 75–82.

[49] C. E. Miles, "Co-evolving real-time strategy game players," Ph.D. dissertation, University of Nevada, Reno, NV, USA, 2007.

[50] K. Mishra, S. Ontañón, and A. Ram, "Situation assessment for plan retrieval in real-time strategy games," in *European Conference on Advances in Case-Based Reasoning*, ser. Lecture Notes in Computer Science, K.-D. Althoff *et al.*, Eds., vol. 5239. Springer, 2008, pp. 355–369.

[51] M. Molineaux and D. Aha, "Defeating novel opponents in a real-time strategy game," in *International Joint Conference on Artificial Intelligence Workshop on Reasoning, Representation, and Learning in Computer Games*, D. W. Aha *et al.*, Eds. The AAAI Press, 2005, pp. 72–77.

[52] M. Molineaux, D. W. Aha, and P. Moore, "Learning continuous action models in a real-time strategy environment," in *Florida Artificial Intelligence Research Society Conference*, D. Wilson and H. C. Lane, Eds. The AAAI Press, 2008, pp. 257–262.

[53] M. Molineaux, M. Klenk, and D. W. Aha, "Goal-driven autonomy in a navy strategy simulation," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, M. Fox and D. Poole, Eds., 2010.

[54] A. M. Mora, A. Fernández-Ares, J. J. Merelo Guervós, and P. García-Sánchez, "Dealing with noisy fitness in the design of a rts game bot," in *EvoApplications*, ser. Lecture Notes in Computer Science, C. D. Chio *et al.*, Eds., vol. 7248. Springer, 2012, pp. 234–244.

[55] H. Muñoz Avila and D. Aha, "On the role of explanation for hierarchical case-based planning in real-time strategy games," in *European Conference on Case-Based Reasoning, Workshop on Explanations in CBR*, 2004.

[56] M. Naveed, D. Kitchin, A. Crampton, L. Chrpa, and P. Gregory, "A monte-carlo path planner for dynamic and partially observable environments," in *Computational Intelligence and Games*. IEEE, 2012, pp. 211–218.

[57] P. H. F. Ng, Y. J. Li, and S. C. K. Shiu, "Unit formation planning in RTS game by using potential field and fuzzy integral," in *Fuzzy Systems*. IEEE, 2011, pp. 178–184.

[58] J. K. Olesen, G. N. Yannakakis, and J. Hallam, "Real-time challenge balance in an RTS game using rtNEAT," in *IEEE Symposium on Computational Intelligence and Games*, P. Hingston and L. Barone, Eds. IEEE, 2008, pp. 87–94.

[59] S. Ontañón, K. Mishra, N. Sugandh, and A. Ram, "Case-based planning and execution for real-time strategy games," in *International Conference on Case-Based Reasoning*, ser. Lecture Notes in Computer Science, R. Weber and M. M. Richter, Eds., vol. 4626. Springer, 2007, pp. 164–178.

[60] ——, "Learning from demonstration and case-based planning for real-time strategy games," in *Soft Computing Applications in Industry*, ser. Studies in Fuzziness and Soft Computing, B. Prasad, Ed., vol. 226. Springer, 2008, pp. 293–310.

[61] N. Othman, J. Decraene, W. Cai, N. Hu, M. Y. H. Low, and A. Gouaillard, "Simulation-based optimization of StarCraft tactical AI through evolutionary computation," in *Computational Intelligence and Games*. IEEE, 2012, pp. 394–401.

[62] M. J. V. Ponsen, H. Muñoz-Avila, P. Spronck, and D. W. Aha, "Automatically generating game tactics through evolutionary learning," *AI Magazine*, vol. 27, no. 3, pp. 75–84, 2006.

[63] M. Preuss, N. Beume, H. Danielsiek, T. Hein, B. Naujoks, N. Piatkowski, R. Stür, A. Thom, and S. Wessing, "Towards intelligent team composition and maneuvering in real-time strategy games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 2, pp. 82–98, 2010.

[64] F. Sailer, M. Buro, and M. Lanctot, "Adversarial planning through strategy simulation," in *IEEE Symposium on Computational Intelligence and Games*. IEEE, 2007, pp. 80–87.

[65] F. Schadd, "Hierarchical opponent models for real-time strategy games," Universiteit Maastricht, Tech. Rep., 2007.

[66] F. Schadd, S. Bakkes, and P. Spronck, "Opponent modeling in real-time strategy games," in *GameOn Conference*, M. Roccetti, Ed. EUROSIS, 2007, pp. 61–68.

[67] M. Sharma, M. P. Holmes, J. C. Santamaría, A. Irani, C. L. Isbell Jr., and A. Ram, "Transfer learning in real-time strategy games using hybrid CBR/RL," in *International Joint Conference on Artificial Intelligence*, M. M. Veloso, Ed., 2007, pp. 1041–1046.

[68] S. Shoemaker, "Random Map Generation for Strategy Games," in *AI Game Programming Wisdom 2*. Hingham, Massachusetts: Charles River Media, Inc., 2004, pp. 405–412.

[69] G. Smith, P. Avery, R. Houmanfar, and S. J. Louis, "Using co-evolved RTS opponents to teach spatial tactics," in *IEEE Conference on Computational Intelligence and Games*, G. N. Yannakakis and J. Togelius, Eds. IEEE, 2010, pp. 146–153.

[70] P. Spronck, I. G. Sprinkhuizen-Kuyper, and E. O. Postma, "On-line adaptation of game opponent ai with dynamic scripting," *Int. J. Intell. Games & Simulation*, vol. 3, no. 1, 2004.

[71] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the NERO video game," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, 2005.

[72] G. Synnaeve and P. Bessiere, "A bayesian model for opening prediction in RTS games with application to StarCraft," in *Computational Intelligence and Games*. IEEE, 2011, pp. 281 –288.

[73] ——, "A bayesian model for RTS units control applied to StarCraft," in *IEEE Conference on Computational Intelligence and Games*. IEEE, 2011, pp. 190 –196.

[74] ——, "Special tactics: a bayesian approach to tactical decision-making," in *Computational Intelligence and Games*. IEEE, 2012, pp. 409–416.

[75] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G. N. Yannakakis, "Multiobjective exploration of the Starcraft map space," in *IEEE Conference on Computational Intelligence and Games*, G. N. Yannakakis and J. Togelius, Eds. IEEE, 2010, pp. 265–272.

[76] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[77] C. K. Tong, C. K. On, J. Teo, and A. M. J. Kiring, "Evolving neural controllers using GA for Warcraft 3 real time strategy game," in *Bio-Inspired Computing: Theories and Applications*. IEEE, 2011, pp. 15–20.

[78] J. M. Traish and J. R. Tulip, "Towards adaptive online RTS AI with NEAT," in *Computational Intelligence and Games*. IEEE, 2012, pp. 430–437.

[79] A. Trusty, S. Ontañón, and A. Ram, "Stochastic plan optimization in real-time strategy games," in *Artificial Intelligence and Interactive Digital Entertainment*, C. Darken and M. Mateas, Eds. The AAAI Press, 2008.

[80] L. van der Blom, S. Bakkes, and P. Spronck, "Map-adaptive artificial intelligence for video games," in *GameOn Conference*, M. Roccetti, Ed. EUROSIS, 2007, pp. 53–60.

[81] M. van der Heijden, S. Bakkes, and P. Spronck, "Dynamic formations in real-time strategy games," in *IEEE Symposium on Computational Intelligence and Games*, P. Hingston and L. Barone, Eds. IEEE, 2008, pp. 47–54.

[82] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *IEEE Symposium on Computational Intelligence and Games*, P. L. Lanzi, Ed. IEEE, 2009, pp. 140–147.

[83] B. G. Weber, M. Mateas, and A. Jhala, "Applying goal-driven autonomy to StarCraft," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, G. M. Youngblood and V. Bulitko, Eds. The AAAI Press, 2010.

[84] S. Wender and I. Watson, "Applying reinforcement learning to small scale combat in the real-time strategy game StarCraft:Broodwar," in *Computational Intelligence and Games*. IEEE, 2012, pp. 402–408.

[85] S. Wintermute, J. Z. Xu, and J. E. Laird, "SORTS: A human-level approach to real-time strategy ai," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, J. Schaeffer and M. Mateas, Eds. The AAAI Press, 2007, pp. 55–60.

[86] G. N. Yannakakis, "Game ai revisited," in *Proceedings of the 9th conference on Computing Frontiers*, ser. CF '12. New York, NY, USA: ACM, 2012, pp. 285–292.

[87] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.