


Article

A Comparison of Evolutionary and Tree-Based Approaches for Game Feature Validation in Real-Time Strategy Games with a Novel Metric

Damijan Novak *, Domen Verber, Jani Dugonik and Iztok Fister, Jr. 

Institute of Informatics, University of Maribor, Koroška Cesta 46, 2000 Maribor, Slovenia;
domen.verber@um.si (D.V.); jani.dugonik@um.si (J.D.); iztok.fister1@um.si (I.F.J.)

* Correspondence: damijan.novak@um.si; Tel.: +386-2-220-7431

Received: 2 April 2020; Accepted: 29 April 2020; Published: 1 May 2020



Abstract: When it comes to game playing, evolutionary and tree-based approaches are the most popular approximate methods for decision making in the artificial intelligence field of game research. The evolutionary domain therefore draws its inspiration for the design of approximate methods from nature, while the tree-based domain builds an approximate representation of the world in a tree-like structure, and then a search is conducted to find the optimal path inside that tree. In this paper, we propose a novel metric for game feature validation in Real-Time Strategy (RTS) games. Firstly, the identification and grouping of Real-Time Strategy game features is carried out, and, secondly, groups are included into weighted classes with regard to their correlation and importance. A novel metric is based on the groups, weighted classes, and how many times the playtesting agent invalidated the game feature in a given game feature scenario. The metric is used in a series of experiments involving recent state-of-the-art evolutionary and tree-based playtesting agents. The experiments revealed that there was no major difference between evolutionary-based and tree-based playtesting agents.

Keywords: evolutionary computation; playtesting; game feature; game simulation; game trees; playtesting metric; validation

1. Introduction

Real-Time Strategy (RTS) games are designed as turn-based games where players, each following their own strategies, try to defeat one another through a series of turns. The term ‘strategy’ stands for the highest form of decision-making process, where the ultimate purpose is to defeat the opponent. Decisions are made between turns (a turn is a transition from the current game state to the next one), which are so short (i.e., in the range of milliseconds) that the game looks as though it is progressing in real time. After a decision is made, the actions are executed. The difference between RTS games and classical turn-based board games, of which probably the most well-known representative is the game of chess, is in the execution of the actions. Actions in RTS games are durative and simultaneous [1], as opposed to the instant moves, of which each player can make one per turn, in classical board games.

During the last decade, RTS games have become one of the best test beds for researching Artificial Intelligence (AI) for games [2,3]. The main reason for the growth in research is the fact that RTS games offer plenty of challenges for researchers. For example, RTS games are representatives of the highest class of computational complexity [4], which is due to their extremely large state-action spaces [5] (i.e., search space). Search space is often impossible to search exhaustively, because a specific game is a high-dimensional space of game variants (many different parameters are available), and it is also called game space [6].

Exploring the search space of games is often considered to be a difficult problem [7], and most of the complex optimization problems relating to games' search spaces cannot be solved using the exact methods that search for the optimal solution by enumerating all possible solutions. To solve these problems, various methods have emerged in the past decades that solve problems approximately. In recent times, researchers have been looking for inspiration for the design of these approximate algorithms/methods in nature, e.g., Darwin's evolutionary theory [8], the collective behavior of social living insects [9], the social behavior of some animal species [10,11], physical phenomena [12], and so on.

The bio-inspired computation field [13] is a field that covers all of the algorithms/methods that fall within the scope of these mentioned inspirations and is an extensively studied research area of AI. Nowadays, numerous algorithms exist that fall under the bio-inspired computation umbrella, such as the Artificial Bee Colony (ABC) Algorithm [14], Differential Evolution (DE) [15], Firefly Algorithm (FA), Genetic Algorithm (GA) [16], Monarch Butterfly Optimization (MBO) [17], etc. Due to the popularity of this subject, numerous unprecedented implications of these approaches exist among real-world applications [13]. Some of the application areas where bio-inspired computation approaches have been successfully applied include: antenna design [18], medicine [19], and dynamic data stream clustering [20].

In addition to the many different application areas, bio-inspired computation also plays an important role in the design and development of games. Bio-inspired computation approaches in games have been used for procedural content generation [21], the development of controllers that are able to play games [22], educational and serious games [23], intelligent gaming systems [24], evolutionary methods in board games [25], behavioral design of non-player characters [26], etc.

Gameplaying agents (algorithms) are made to play the game in question, with the game rules being hard-coded or self-obtained (general gameplaying), in a self-sustained way (i.e., no human input is needed during the (general) gameplay) [27]. The primary task of the gameplaying agent is to win games, and the secondary task is to win them with a higher score [28]. For the RTS gameplaying agent [29] to be able to cope with the high computational complexity of the game space, it has to be able to function inside different segments of the game, which are as follows: resource and production management (also categorized as economy) [30], strategical [31], tactical [32] and micromanagement [33] operations, scouting [34] and sometimes even diplomacy [35]. For one to be successful when playing an RTS game, a balanced combination of all those segments must be considered by the agent [36]. Since gameplaying agents are already built to operate and cover a variety of tasks in a given game space, they can be adapted to become playtesting agents.

Playtesting agents are meant to play through the game (or a slice of it) and try to explore the behavior that can generate data that would assist developers during the development phase of a game [37,38]. Game studios conduct countless tests on gameplaying with real players [39], but relying on humans for playtesting can result in higher costs and can also be inefficient [37]. The research on playtesting is, therefore, very important for the following two reasons: it has a huge economic potential and is of considerable interest to the game industry [40]. Further economic potential is also apparent in semi-related fields, like Gamification [41].

A Game Design Document (GDC) specifies core gameplay, game elements, necessary game features, etc. [42]. With this paper, we tackle the problem of the automatic validation of game features for the game space specified in GDC and also address research requirements from articles (for instance, [43]), where the authors point out the need that games with a higher complexity have of scaling.

In this article, we will try to find the answers to the following research questions:

- RQ1: How easy is it to adapt gameplaying agents as playtesting agents in RTS games?
- RQ2: Which RTS game definitions can be used to make a comparison between different playtesting agents?
- RQ3: How to evaluate playtesting agents based on RTS game definitions, and which are the most beneficial to them?

- RQ4: Is there a difference between evolutionary and the non-evolutionary approaches (like standard Monte Carlo tree searches [44]) with regard to playtesting abilities?
- RQ5: How does one define valid/invalid game features in the game space?

Altogether, the main contributions of this paper are as follows:

- A novel metric is proposed to make a comparison between different playtesting agents;
- A method is proposed for adapting gameplaying agents as playtesting agents in real-time strategy games; and
- The proposed metric is used in a series of experiments involving adapted evolutionary and tree-based state-of-the-art gameplaying agents.

The structure of the remainder of this paper is as follows. Section 2 outlines the game features of real-time strategy games and the microRTS simulation environment, while Section 3 presents the proposed novel metric that will allow for the comparison of different playtesting agents. Section 4 describes the experimental environment, adaptation of gameplaying agents as playtesting agents (including detailed descriptions of them) and the results of the experiments. A Discussion is provided in Section 5, and the conclusion is presented in Section 6.

2. Real-Time Strategy Games

This chapter briefly outlines the game features of RTS games, and a description of the microRTS environment is also provided.

2.1. Game Features of RTS Games

“Game feature” is a generic term used to refer to differences and similarities between games [45]. Game features are defined in GDC [46], and, after they are implemented, each game’s features rely on the use of game mechanics. Game mechanics are methods invoked by agents in interacting with the game world (e.g., to obtain the health value of the unit) [47]. In [48], 18 general definitions of game features (hereinafter referred to as groups) can be found.

In the RTS game domain, different kinds of game feature subset groupings are possible (Economic, Military, Map Coverage, Micro Skill and Macro Skill) [49], but to the best of our knowledge, the RTS game features have not yet been placed into groups. The placement of RTS game features into groups is, in our opinion, important, because it allows for the possibility of comparing RTS game features with the features of other game genres in the future.

2.2. microRTS

There are many different RTS game worlds in existence. Not all of them are openly available, but even some of the commercial ones have been opened up for research purposes (e.g., StarCraft™ was opened through the programming interface). microRTS is a simple non-commercial simulation environment, which was created to test any theoretical ideas a game researcher might have.

This simulation environment follows standard RTS game genre game rules:

1. Players gather resources and use them to create structures and new mobile units;
2. The game goal is to defeat the opposing player in a battle for supremacy; and
3. Resources, structures, and mobile units must be cleverly used.

The microRTS environment includes the following features (seen in Figure 1):

- Four mobile units: worker, light (battle unit), heavy (battle unit) and ranged (battle unit);
- Two structures: base and barracks;
- Resources; and
- A wall.

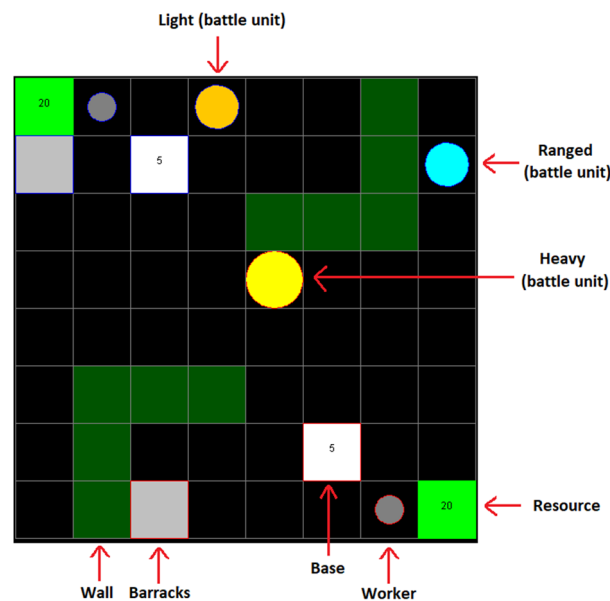


Figure 1. Micro real-time strategy (microRTS) environment, with all features visible.

Workers are used to gather resources and build structures, and they also possess the ability to attacks with limited firepower. Light, heavy and ranged are the main battle units used for attacks on opponent structures and mobile units. Battle units have different initial properties (i.e., a heavy battle unit can sustain more damage before being destroyed versus a light battle unit, and a ranged unit can shoot farther). Bases produce workers. Barracks are used to create battle units. The wall is used as a physical barrier in the map.

microRTS allows configurable scenarios to be placed in the environment. Figure 1 presents one such scenario: an 8×8 cell map, with fixed positions for resources (in the top left and bottom right corners) and walls. The mobile units are not fixed and can be moved freely inside this environment.

Scenarios can be configured for varying map sizes (4×4 , 8×8 , 12×12 , etc.) and with different starting positions for the unit types, structures, and resources (which can be placed anywhere on the map). The game can be played with visible features (graphical interface turned on for observations) or in the background (which allows for a faster execution of scenarios and quicker overall simulations, with less computer resources used).

microRTS also already includes many gameplaying agents that can be used in experiments.

3. Proposal of a Metric for Game Feature Validation

Our motivation to create a metric came from the need to be able to differentiate easily between different playtesting agents' performances, when multiple game features need to be validated. In order to propose a novel metric for comparing playtesting agents, the following steps were considered in our study:

- STEP 1: The RTS game features are identified;
- STEP 2: The game features are grouped in precise game feature groups; (STEP 2.1): Classification of game feature groups according to their correlation (groups that are similar in description tend to be correlated, and this also allows single game features to be placed into multiple groups) and importance (some groups are of a higher importance, because they reflect and are essential to RTS gameplay, while some could be left out without jeopardizing the game's position in the RTS game genre);
- STEP 3: For empty groups in STEP 2, a further identification of the RTS game features is conducted by including more search strings and other search engines (e.g., Google Scholar); and
- STEP 4: The novel metric is proposed.

All steps are described in detail in the following subsections and are presented graphically in Figure 2.

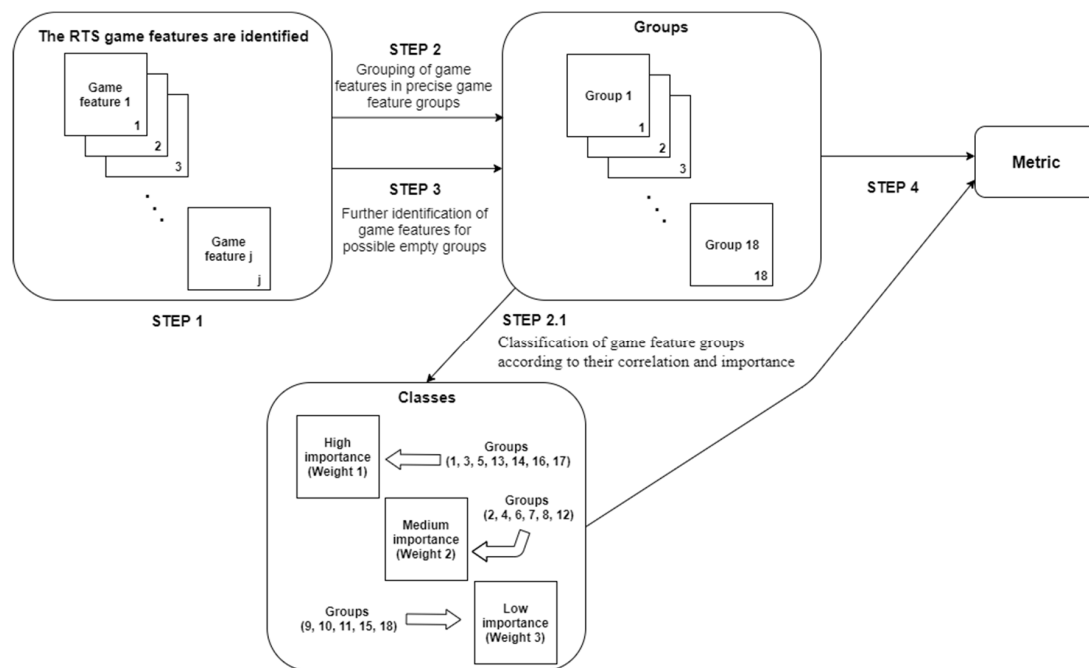


Figure 2. Pathway (steps taken) from the identification of the RTS game features to the novel metric.

3.1. Identification of RTS Game Features

Game features are mentioned in many RTS game research works, but they are scattered across different subdomains and research agendas. Our goal was to use the pool of research articles and dissertations and to identify the game features included in this research. The pool was reviewed with the help of a literature search. The ISI Web of Science and ProQuest research search engines were used. A search query with the following search string was made: “game features” and “real-time strategy games”, which returned 88 hits for the ISI Web of Science and 34 hits for ProQuest.

The results (articles and dissertations) were filtered to exclude non-RTS game research works. A manual search was conducted through the research work for mentions of the “feature” string (note: 14 works from the ISI Web of Science and 0 for ProQuest were located after a manual search). The located text was extracted and analyzed for surrounding context, then transformed into a compact format that could act as a short game feature description. The surrounding context was used to transform the text, because not all research work has game features that can be used as-is. A short description was then made of the list of game feature descriptions. If a description was already on the list, and it was not adding additional information, it was omitted (note: seven works from the ISI Web of Science were omitted). Additionally, one research work can include more than just one mention of the string “feature” with the surrounding context.

Note: Future work could broaden the scope and include other search strings (like “aspect” or “feature”) for a more in-depth survey of the general RTS features.

Table 1 includes a list of short game feature descriptions, which were produced after the completion of the first step. The short game feature descriptions are accompanied by a reference.

Table 1. Game feature descriptions derived from related work.

Game Feature Label	Short Game Feature Description	Short Game Feature Label	Reference (Used as a Basis for Extraction)
Resource gathering	Game unit (worker) collects at least x units of type A resources and at least y units of type B resources in num trips.	GF1_RG	[50]
Game engine features and objects	Game unit (battle unit) always hits with x points of damage.	GF2_EOBJ	[51]
Game difficulty (aiding)	The opponent is aided with x more units, resulting in a player losing every game. Note: such a feature can be part of an advanced mode, where non-advanced users must not/cannot win.	GF3_DIFA	[52]
Game objective (construction)	If the player tries to, it must be able to create x game structure(s) (e.g., barracks).	GF4_CONS	[53]
Game assessment	Game score is calculated based on raw features (e.g., no. of workers) and must represent the game state status correctly when presented to the player.	GF5_AST	[54]
Stumbling block	The player cannot destroy the enemy in a specific part of the map due to stumbling blocks (e.g., a wall).	GF6_SB	[55]
Game exploration (unlocking new technologies)	If the player tries discovery, it can create x game units (e.g., battle unit–light) through the usage of game structure(s) (e.g., barracks).	GF7_EXPL	[56]
Special unit	The player is confronted with a special game unit (e.g., Super-Heavy with special features), which cannot be destroyed with the given resources.	GF8_FANT	[57]
Partial information (fog-of-war)	The player cannot operate in a partially observable environment, so it therefore cannot destroy the opponent in such an environment.	GF9_PARI	[58]
Game difficulty (challenge)	The player cannot destroy x structures (e.g., barracks) guarded with y rushing game units (e.g., battle unit–heavy) with access to z units of A type resources.	GF10_DIFC	[52]
Game control (take over the map)	The player can destroy all the structures on the map before the time runs out.	GF11_GCMP	[59]
Interaction on a complex map	If the player controlling x battle units (e.g., a heavy battle unit) finds a static unit (e.g., barracks) in a maze (or complex map), the static unit is always destroyed.	GF12_INTE	[60]
Resource gathering under attack	A gatherer (e.g., a worker) is always destroyed when trying to gather resources.	GF13_RG2	[61]

3.2. Grouping the Game Features into Specific Groups

The grouping of game features into specific groups has two benefits: a group consists of game features with similar modus operandi (i.e., correlated and in the same context), and groups can serve as a basis for sharing research with other game genres.

We already mentioned (Section 2.1) that the literature search revealed 18 groups, which are formed independently of the specific game genre, and which we will use for grouping. These groups are: adaptation, assessment/rewards/scores, challenge, conflict, control, exploration, fantasy/location, interaction/interactivity (equipment), interaction (interpersonal/social), language/communication, motivation, mystery, pieces or players, progress and surprise, representation, rules/goals, safety and sensory stimuli. A detailed description about the meaning of each of the groups can be found in the tabular presentation in [62].

Table 2 presents the results after the completion of both steps, with references to the source of the compact description. Our goal was to have at least one game feature representative for each of the groups. If there was no game feature available in Table 1 for an empty group, we tried to locate the research work for that group by searching via Google Scholar (STEP 3) using different search strings (e.g., “impassable terrain” for a conflict group) in regard to the context of the group. The research

works found went through the procedure described in STEP 1, and a short game feature description was included in Table 1 and in the accordingly empty group in Table 2. (Observation: we noticed that many research works on game feature descriptions originated from the domain of player/opponent modeling, RTS replay analysis, game balancing and strategy selection/prediction.) One game feature can belong to more than one group. For some groups, we could not find or create any viable game feature description that could be measured by the game mechanics. Such groups remained empty but were still included in the Table. The reason: future RTS research could produce game features for currently empty groups.

Table 2. Game definition groups and their game feature representatives.

ID	Group	Short Game Feature Label
G ₁	Adaptation	GF3_DIFA ¹ , GF10_DIFC
G ₂	Assessment/Rewards/Scores	GF5_AST ¹
G ₃	Challenge	GF3_DIFA, GF8_FANT, GF10_DIFC ¹ , GF12_INTE
G ₄	Conflict	GF6_SB ¹ , GF9_PARI, GF10_DIFC, GF12_INTE, GF13_RG2
G ₅	Control	GF2_EOBJ, GF9_PARI, GF10_DIFC, GF11_GCMP ¹ , GF12_INTE
G ₆	Exploration	GF7_EXPL ¹ , GF9_PARI, GF12_INTE
G ₇	Fantasy/Location	GF8_FANT ¹
G ₈	Interaction/Interactivity (Equipment)	GF2_EOBJ, GF4_CONS, GF7_EXPL, GF12_INTE ¹
G ₉	Interaction (Interpersonal/Social)	(empty—beyond the scope of this article ²)
G ₁₀	Language/Communication	(empty)
G ₁₁	Motivation	(empty)
G ₁₂	Mystery	GF9_PARI ¹
G ₁₃	Pieces or Players	GF1_RG, GF2_EOBJ ¹ , GF3_DIFA, GF4_CONS, GF5_AST, GF6_SB, GF7_EXPL, GF8_FANT, GF9_PARI, GF10_DIFC, GF11_GCMP, GF12_INTE, GF13_RG2
G ₁₄	Progress and Surprise	GF1_RG, GF4_CONS ¹ , GF6_SB, GF7_EXPL, GF8_FANT, GF9_PARI, GF10_DIFC, GF11_GCMP, GF12_INTE, GF13_RG2
G ₁₅	Representation	(empty)
G ₁₆	Rules/goals	GF1_RG ¹ , GF2_EOBJ, GF4_CONS, GF7_EXPL, GF13_RG2
G ₁₇	Safety	GF1_RG, GF13_RG2 ¹
G ₁₈	Sensory stimuli	(empty)

¹ Representative of the group used for the experiment. ² The interaction (Interpersonal/Social) group was left empty, because it would require the interaction of multiple players (a single gameplaying agent modified for a playtesting agent supports only single player operations).

3.3. Classification of Feature Groups According to Their Correlation and Importance

As game features tend to be correlated, so do groups. One group can be, context wise, closely related to some groups but only loosely related to others. Additionally, some contexts are more important than others with regard to RTS gameplay.

Table 3 presents the classification of feature groups into three importance classes:

- The high-importance class contains groups that represent the essence of RTS gameplay (based on our understanding of the RTS game worlds and their aspects [63]);
- Groups that operate on a game mechanics level (e.g., Interaction/Interactivity (Equipment) group) or are not essential to the game (they could potentially be left out, e.g., Mystery group) are in the medium-importance class; and
- Groups that, in Table 2, did not have a feature representative (empty of features) were included in the low-importance class.

Table 3. Classification of feature groups based on their correlation and importance.

Class	Groups	Weight	Set
High importance	Adaptation, Challenge, Control, Pieces or Players, Progress and Surprise, Rules/goals, Safety	W1	$C_H = \{G_1, G_3, G_5, G_{13}, G_{14}, G_{16}, G_{17}\}$
Medium importance	Assessment/Rewards/Scores, Conflict, Exploration, Fantasy/Location, Interaction/Interactivity (Equipment), Motivation, Mystery	W2	$C_M = \{G_2, G_4, G_6, G_7, G_8, G_{12}\}$
Low importance	Interaction (Interpersonal/Social), Language/Communication, Representation, Sensory stimuli	W3	$C_L = \{G_9, G_{10}, G_{11}, G_{15}, G_{18}\}$

The importance level of each of the groups is represented by a class. Regarding the game worlds, we allow for the possibility of different reconfigurations of the groups inside the classes. We also included the weight and mathematical description of the set. Weight is a numerical value that is set by the user of the metric. It represents how much the groups belonging to the specific class will count towards the metric score.

3.4. Proposal of the Metric

In this subchapter, we explain our metric for summarizing agents’ performance while they validate game features in an RTS game space. The metric calculates its score based on how many times the playtesting agent invalidated the game feature of a fixed number of repeats for a given scenario (the sum of validations and invalidations equals the number of scenario repeats).

If the playtesting agent during the execution of the scenario could not test the game feature, because it does not come into a situation, or it is not programmed to deal with the situation where validation can take place, then such a game feature is valid from this point of view. The number of successful validations is, therefore, omitted from the game score, since it is biased.

For a set of groups $G_i, 1 \leq i \leq 18$, where each member of group G_i holds a set of Game features (GFs) ($GF_j \in G_i, 0 \leq j$), and each GF_i holds a set of executable scenarios S ($S_k \in GF_i, 1 \leq k$), the number of unsuccessful validations per scenario is defined by $numInvalid_{ijk}$, and the number of times the scenario is repeated is defined by $numOfScenRep = n, 1 \leq n$, the following formulas apply:

$$invalidPercPerScen (ijk, numOfScenReps) = numInvalid_{ijk}/numOfScenRep \tag{1}$$

$$calcSetScore (set, numOfScenReps) = \sum_{i \in set} \sum_{j \geq 0} \sum_{k \geq 1} invalidPercPerScen (ijk, numOfScenRep) \tag{2}$$

$$\begin{aligned}
 agentPlaytestingScore = \\
 & W1 * calcSetScore (\{1, 3, 5, 13, 14, 16, 17\}, numOfScenRep) \\
 & + W2 * calcSetScore (\{2, 4, 6, 7, 8, 12\}, numOfScenRep) \\
 & + W3 * calcSetScore (\{9, 10, 11, 15, 18\}, numOfScenRep)
 \end{aligned} \tag{3}$$

In Equation (1), the number of invalidations of a given group (index i), game feature (index j) and scenario (index k) is divided by the total number of scenario repeats. In Equation (2), the score is calculated for all the game features and scenarios that the set of groups holds. In Equation (3), the scores of the set of groups are multiplied by their respective weights.

4. Experiments and Results

In this chapter, we present the specifications of hardware and software used for the experimental environment, as well as the results of the experiments.

4.1. Experimental Environment

Hardware: The experiment was carried out on an i7-3770k CPU computer @ 3.50 (turbo: 3.9) GHz, 4 cores (note: during the experimentation, only one core was used, since agents do not implement the multi-core support) and 16 GB RAM.

Software: OS Windows 10 Pro and Java Development Kit 13.0.2. The experiment was set in the latest version of the microRTS environment, acquired from an online source at the time of preparing this article [64]. The microRTS environment comes pre-loaded with the following gameplaying agents: RandomAI, RandomAIBiased, MonteCarlo, IDRTMinimax, IDRTMinimaxRandomized, IDABCD, UCT, PuppetSearchMCTS, and NaiveMCTS. TiamatBot was acquired from the online source [65]. MixedBot (which includes TiamatBot source files but an improved version) was acquired from the online source [66] and was included in the microRTS environment. Every gameplaying agent is used in the experiment as it was acquired from the online source of original authors (i.e., no code or internal parameter was changed for experimental purposes).

Table 4 shows the hyper-parameters used for the validation of every game feature presented in Table 1.

Table 4. Hyper-parameters used in the experimentation.

Hyper-Parameter	Value
continuing	true
max_actions	100
max_playouts	-1
playout_time	100
max_depth	10
randomized_ab_repeats	10
max_cycles	3000
max_inactive_cycles	300

These hyper-parameters are pre-set within the microRTS environment. The only parameter that we changed was iterations, which we set to 50 (before it was set to 10). The standard UnitTypeTable was used where necessary. Note: to validate the GF9_PARI, we changed the environment from fully observable to partially observable.

The game feature descriptions presented in Table 1 were derived from related works and written independently of a specific game environment, i.e., they can be implemented in any RTS game engine. In Table 5, we present the same game features as those presented in Table 1, although the former are adapted to the microRTS environment and a specific scenario. All game features in Table 5 are written with the assumption that they are valid for the microRTS environment. If the playtesting agent actually manages to invalidate a game feature from the list, it will add to its metric score.

4.2. Adaptation of Gameplaying Agents as Playtesting Agents

To adapt a gameplaying agent to the playtesting task, we created a non-intrusive component. The component contains information about the scenario (map, position of units and the opponent) and controls the validation procedure by following the playtesting agents' progress (i.e., actions that it executes) and by accessing game environment information (e.g., current game state status). All the information is accessible through well-defined interfaces of the microRTS source code. One of the interface methods is the method that returns the best action for the given game state, and every gameplaying agent operating in the microRTS environment implements it.

Table 5. microRTS game feature scenario.

Short Game Feature Label	Experimental microRTS Game Feature Description	Map
GF1_RG	Worker collects at least 2 units of a resource in 2 trips.	basesWorkers8x8.xml (standard map, which comes with microRTS)
GF2_EOBJ	A light battle unit always hits with 2 points of damage.	melee4x4light2.xml (standard map)
GF3_DIFA	The opponent is aided by 5 more heavy battle units, resulting in the player losing every game.	basesWorkers8x8.xml (standard map with 5 heavy units added for the opponent)
GF4_CONS	If the player tries to, they must be able to create 1 barracks.	basesWorkers8x8.xml (standard map)
GF5_AST	The game score is calculated on the basis of raw features of the game state (no. of workers and no. of light, heavy and ranged units multiplied by their cost factors) and must represent the game state status correctly when presented to the player.	melee14x12Mixed18.xml (standard map)
GF6_SB	The player cannot destroy the enemy in a specific part of the map due to a wall.	basesWorkers12x12.xml (standard map with a wall placed in the middle of the map).
GF7_EXPL	If the player tries discovery, it must be able to create 1 light battle unit through the usage of game barracks.	basesWorkers8x8.xml (standard map)
GF8_FANT	The player is confronted with a special game unit (Super-Heavy battle unit with ten-times the armor of a normal-Heavy one), which cannot be destroyed with the given resources.	basesWorkers8x8 (standard map with Super-Heavy battle units added to help the opponent)
GF9_PARI	The player cannot operate in a partially observable environment, so it therefore cannot destroy the opponent in such an environment.	basesWorkers12x12.xml (standard map with a partially observable environment enabled)
GF10_DIFC	The player cannot destroy 2 barracks guarded with 3 heavy rushing units with access to 60 units of resources.	8x8_2barracks3rushingHeavy60res.xml (custom map)
GF11_GCMP	The player can destroy three barracks before the time runs out.	8x8_3barracks.xml (custom map)
GF12_INTE	If the player controlling four heavy battle units finds an enemy barracks in a large map (with obstacles and walls), the enemy barracks are always destroyed.	chambers32x32.xml (standard map with four heavy battle units and barracks added)
GF13_RG2	The worker is always destroyed when trying to gather resources.	8x8_workerDestroyed.xml (custom map with the base and resources on different parts of the map and four light battle units in the middle)

When the actions are executed in a game state, it cycles to the next one (i.e., actions change the inner state). During such cycles, our component tests if the Game Feature is valid or invalid. A Game Feature is invalid if the condition that is written in the validation procedure of the game feature in question is not fulfilled. The condition is tested against the information provided from the agent's executed action and the environment's current game state. The validation procedure checks the validity of the game feature, until either the maximum number of cycles is reached, or the game is over (i.e., one of the players has no more units left on the field).

For example, the game feature, GF8_FANT, is validated by checking if the resulting game state still holds this special unit after the agent has given an order to fire on it. If in any cycle the unit is destroyed, the game feature is invalid.

4.3. Playtesting Agents

The following gameplaying agents have been adapted as playtesting agents for the purposes of experimentation:

1. Basic (part of the microRTS package):
 - RandomAI: The choice of actions is completely random;
 - RandomBiasedAI: Based on RandomAI, but with a five times higher probability of choosing fighting or harvesting action over other actions; and
 - MonteCarlo: A standard Monte Carlo search algorithm.
2. Evolutionary Algorithm (online source):
 - TiamatBot (original): Uses an evolutionary procedure to derive action abstractions (conducted as a preprocessing step [67]). The generation of action abstractions can be cast as a problem of selecting a subset of pure strategies from a pool of options. It uses Stratified Strategy Selection (SSS) to plan in real time in the space defined by the action abstraction thus generated [68]. It outperformed the best performing methods in the 2017 microRTS competition [69] and is therefore considered as one of the current state-of-the-art gameplaying agents.
3. Tree-Based (part of the microRTS package):
 - IDRTMinimax: An iterative-deepening version of RTMinimax (minimax is defined here by time, not by agent moves) that uses available time to search in a tree as deeply as possible;
 - IDRTMinimaxRandomized: An agent that uses randomized alpha-beta (a better assessment for situations where players execute moves simultaneously);
 - IDABCD: Alpha-beta considering duration. It is a modified RTMinimax [70];
 - UCT: Standard UCT (with a UCB1 sampling policy);
 - PuppetSearchMCTS: An adversarial search framework based on scripts that can expose choice points to a look-ahead procedure. A Monte Carlo adversarial search tree was used to search over sequences of puppet moves. The input script into an agent's constructor was a basic configurable script that used a Unit Type Table [71].
 - NaiveMCTS: A Standard Monte Carlo search, but which uses naïve sampling [72]. Two variations of the same algorithm were used (which differ in their initial parameter settings): NaiveMCTS#A (max_Depth = 1, $\epsilon_1 = 0.33$, $\epsilon_0 = 0.75$) and NaiveMCTS#B (max_depth = 10, $\epsilon_1 = 1.00$, $\epsilon_0 = 0.25$).
4. Evolutionary and Tree-Based (online source):
 - MixedBot: This bot integrates three bots into a single agent. The TiamatBot (improved original) was used for strategy decisions, Capivara was used for tactical decisions [73], and MicroRTSbot [74] included a mechanism that could change the time allocated for two decision parts dynamically based on the number of close armies. MixedBot placed second in the 2019 microRTS (standard track) competition (first place went to the game bot that also uses offline/out-game learning [75]).

4.4. Results of the Playtesting Agents

For each of the playtesting agents, Table 6 shows how many times the group's game feature representative was validated or invalidated. Table 6 also shows what metric score they acquired. The weights for calculating the metric score were set as follows: $W_1 = 1$, $W_2 = 0.5$ and $W_3 = 0$. W_3 was set to 0, because the C_L class groups are devoid of features. Additionally, empty groups were omitted from the Table.

To allow for clearer results, we abbreviated the game feature representatives' labels, e.g., G_1 and its GF3_DIFA Game Feature representative, if validated 50 times and invalidated 0 times, was shortened to G1GF3(50, 0).

Table 6. Playtesting agent results for feature validations and their metric score.

Playtesting Agent	Groups and Game Features (Valid num./Invalid num.)	Metric Score
RandomAI	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(50, 0), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0
RandomBiasedAI	G1GF3(49, 1), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(28, 22), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.24
MonteCarlo	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.5
TiamatBot	G1GF3(21, 29), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.08
IDRTMinimax	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.5
IDRTMinimaxRandomized	G1GF3(50, 0), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.5
IDABCD	G1GF3(49, 1), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.52
UCT	G1GF3(24, 26), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.02
PuppetSearchMCTS	G1GF3(46, 4), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.58
NaiveMCTS#A	G1GF3(11, 39), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.28
NaiveMCTS#B	G1GF3(12, 38), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	1.26
MixedBot	G1GF3(34, 16), G2GF5(50, 0), G3GF10(50, 0), G4GF6(50, 0), G5GF11(50, 0), G6GF7(50, 0), G7GF8(0, 50), G8GF12(50, 0), G12GF9(50, 0), G13GF2(50, 0), G14GF4(50, 0), G16GF1(50, 0), G17GF13(50, 0)	0.82

Table A1, which, due to its size, can be found in Appendix A, shows how the metric score changes for each of the playtesting agents and all combinations of the W1 to be decreased from 1 to 0.55 (with steps of 0.05) and those of W2 to be decreased from 0.50 to 0.05 (also with steps of 0.05). Note: the data

used for calculating the metric scores is the same as those presented in the second column of Table 6. RandomAI was omitted from Table A1, because its metric score is zero for all the combinations (it did not invalidate any of the features).

5. Discussion

During the experimentation phase, the microRTS game engine environment performed as expected (i.e., without visible or known bugs). Our presumption from the start of the experiment was that all of the Game Features were valid, yet the experiments showed that two of the Game Features were actually invalid (GF3 and GF8). A closer inspection of the GF3 results, specifically its invalidation number, revealed that not all of the playtesting agents caught the invalid game feature, and that some of them only invalidated it in a fraction of tries. Additionally, if the number of scenario repeats would be set to lower than fifty, it is possible that only the playtesting agents with a better performance would be successful in finding GF3 to be invalid.

GF3 was invalidated by eight playtesting agents, while GF8 was invalidated by all of them, with the only exception being the basic RandomAI. The difference in the number of playtesting agents that invalidated the game features, GF3 and GF8, successfully shows us that some game features are more sophisticated and require agents that intelligently explore and exploit the search space in question.

We discovered two important guidelines for validation testing:

1. Good agents' gameplaying performance is important, because it also reflects playtesting performance; and
2. With a greater number of scenario repeats comes a higher probability of game features being valid.

Our purpose was not to judge the existing gameplaying agents created by the research community based on the score they achieved. We did, however, use the invalid number part that they attained to calculate the metric score for metric testing purposes. The results were encouraging. The state-of-the-art evolutionary and tree-based agents were good performers, not just for gameplaying, but also for playtesting. The line between basic agents (e.g., G1GF3(50, 0)) and advanced ones (e.g., G1GF3(21, 29)) can also be clearly seen. We did not measure the average time for an agent to complete a scenario, but during playtesting, we noticed that agents that were either basic (e.g., RandomAI) or very good performers (e.g., NaiveMCTS) completed the validations in the fastest amount of time. We believe that this resulted from decisions being made quickly (either bad or good).

At this point, we can also provide answers to the research questions presented in the Introduction. RQ1: the adaptation of a gameplaying agent as a playtesting agent is straightforward, provided that the game engine follows good software design techniques (components, interfaces, etc.). In our estimation, this is very important, because it allows for research discoveries in the gameplaying domain to be transferred to the playtesting domain and probably also for higher adaptation rates of such discoveries for commercial use. RQ2: In comparing different playtesting agents, our metric relies on the groups presented in Table 2. The groups belong to different classes (Table 3), each with their own weights. Additional information for comparisons can also be found based on the calibration of these weights. For that purpose, Table A1 was created in Appendix A, which shows how the metric score changes in relation to the changes of the weights. In this way, we can give importance to a specific set of groups and achieve a greater differentiation between the playtesting agents covering them. RQ3: playtesting agents are evaluated through game feature definitions using the created metric. The most beneficial Game Feature definitions are the ones that belong to the groups that are in the high-importance class, shown in Table 3. RQ4: evolutionary and non-evolutionary approaches in the state-of-the-art segment both performed well, and their playtesting abilities are high. No major differences were detected for the game features and scenarios tested. RQ5: the validity of the game feature was defined, with the condition of the validation procedure inside the component used for the adaptation of the gameplaying agents.

Table A1. Cont.

Playtesting Agent	Metric Scores										
	1	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	
TiamatBot	0.5	1.08	1.051	1.022	0.993	0.964	0.935	0.906	0.877	0.848	0.819
	0.45	1.03	1.001	0.972	0.943	0.914	0.885	0.856	0.827	0.798	0.769
	0.4	0.98	0.951	0.922	0.893	0.864	0.835	0.806	0.777	0.748	0.719
	0.35	0.93	0.901	0.872	0.843	0.814	0.785	0.756	0.727	0.698	0.669
	0.3	0.88	0.851	0.822	0.793	0.764	0.735	0.706	0.677	0.648	0.619
	0.25	0.83	0.801	0.772	0.743	0.714	0.685	0.656	0.627	0.598	0.569
	0.2	0.78	0.751	0.722	0.693	0.664	0.635	0.606	0.577	0.548	0.519
	0.15	0.73	0.701	0.672	0.643	0.614	0.585	0.556	0.527	0.498	0.469
	0.1	0.68	0.651	0.622	0.593	0.564	0.535	0.506	0.477	0.448	0.419
	0.05	0.63	0.601	0.572	0.543	0.514	0.485	0.456	0.427	0.398	0.369
	IDRTMinimax	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.45		0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
0.4		0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
0.35		0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
0.3		0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
0.25		0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.2		0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
0.15		0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
0.1		0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
0.05		0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
IDRTMinimaxRandomized		0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45
	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.4
	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15
	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	IDABCD	0.5	0.52	0.519	0.518	0.517	0.516	0.515	0.514	0.513	0.512
0.45		0.47	0.469	0.468	0.467	0.466	0.465	0.464	0.463	0.462	0.461
0.4		0.42	0.419	0.418	0.417	0.416	0.415	0.414	0.413	0.412	0.411
0.35		0.37	0.369	0.368	0.367	0.366	0.365	0.364	0.363	0.362	0.361
0.3		0.32	0.319	0.318	0.317	0.316	0.315	0.314	0.313	0.312	0.311
0.25		0.27	0.269	0.268	0.267	0.266	0.265	0.264	0.263	0.262	0.261
0.2		0.22	0.219	0.218	0.217	0.216	0.215	0.214	0.213	0.212	0.211
0.15		0.17	0.169	0.168	0.167	0.166	0.165	0.164	0.163	0.162	0.161
0.1		0.12	0.119	0.118	0.117	0.116	0.115	0.114	0.113	0.112	0.111
0.05		0.07	0.069	0.068	0.067	0.066	0.065	0.064	0.063	0.062	0.061
UCT		0.5	1.02	0.994	0.968	0.942	0.916	0.89	0.864	0.838	0.812
	0.45	0.97	0.944	0.918	0.892	0.866	0.84	0.814	0.788	0.762	0.736
	0.4	0.92	0.894	0.868	0.842	0.816	0.79	0.764	0.738	0.712	0.686
	0.35	0.87	0.844	0.818	0.792	0.766	0.74	0.714	0.688	0.662	0.636
	0.3	0.82	0.794	0.768	0.742	0.716	0.69	0.664	0.638	0.612	0.586
	0.25	0.77	0.744	0.718	0.692	0.666	0.64	0.614	0.588	0.562	0.536
	0.2	0.72	0.694	0.668	0.642	0.616	0.59	0.564	0.538	0.512	0.486
	0.15	0.67	0.644	0.618	0.592	0.566	0.54	0.514	0.488	0.462	0.436
	0.1	0.62	0.594	0.568	0.542	0.516	0.49	0.464	0.438	0.412	0.386
	0.05	0.57	0.544	0.518	0.492	0.466	0.44	0.414	0.388	0.362	0.336

Table A1. Cont.

Playtesting Agent	Metric Scores										
	1	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	
PuppetSearchMCTS	0.5	0.58	0.576	0.572	0.568	0.564	0.56	0.556	0.552	0.548	0.544
	0.45	0.53	0.526	0.522	0.518	0.514	0.51	0.506	0.502	0.498	0.494
	0.4	0.48	0.476	0.472	0.468	0.464	0.46	0.456	0.452	0.448	0.444
	0.35	0.43	0.426	0.422	0.418	0.414	0.41	0.406	0.402	0.398	0.394
	0.3	0.38	0.376	0.372	0.368	0.364	0.36	0.356	0.352	0.348	0.344
	0.25	0.33	0.326	0.322	0.318	0.314	0.31	0.306	0.302	0.298	0.294
	0.2	0.28	0.276	0.272	0.268	0.264	0.26	0.256	0.252	0.248	0.244
	0.15	0.23	0.226	0.222	0.218	0.214	0.21	0.206	0.202	0.198	0.194
	0.1	0.18	0.176	0.172	0.168	0.164	0.16	0.156	0.152	0.148	0.144
	0.05	0.13	0.126	0.122	0.118	0.114	0.11	0.106	0.102	0.098	0.094
NaiveMCTS#A	1	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	
	0.5	1.28	1.241	1.202	1.163	1.124	1.085	1.046	1.007	0.968	0.929
	0.45	1.23	1.191	1.152	1.113	1.074	1.035	0.996	0.957	0.918	0.879
	0.4	1.18	1.141	1.102	1.063	1.024	0.985	0.946	0.907	0.868	0.829
	0.35	1.13	1.091	1.052	1.013	0.974	0.935	0.896	0.857	0.818	0.779
	0.3	1.08	1.041	1.002	0.963	0.924	0.885	0.846	0.807	0.768	0.729
	0.25	1.03	0.991	0.952	0.913	0.874	0.835	0.796	0.757	0.718	0.679
	0.2	0.98	0.941	0.902	0.863	0.824	0.785	0.746	0.707	0.668	0.629
	0.15	0.93	0.891	0.852	0.813	0.774	0.735	0.696	0.657	0.618	0.579
	0.1	0.88	0.841	0.802	0.763	0.724	0.685	0.646	0.607	0.568	0.529
0.05	0.83	0.791	0.752	0.713	0.674	0.635	0.596	0.557	0.518	0.479	
NaiveMCTS#B	1	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	
	0.5	1.26	1.222	1.184	1.146	1.108	1.07	1.032	0.994	0.956	0.918
	0.45	1.21	1.172	1.134	1.096	1.058	1.02	0.982	0.944	0.906	0.868
	0.4	1.16	1.122	1.084	1.046	1.008	0.97	0.932	0.894	0.856	0.818
	0.35	1.11	1.072	1.034	0.996	0.958	0.92	0.882	0.844	0.806	0.768
	0.3	1.06	1.022	0.984	0.946	0.908	0.87	0.832	0.794	0.756	0.718
	0.25	1.01	0.972	0.934	0.896	0.858	0.82	0.782	0.744	0.706	0.668
	0.2	0.96	0.922	0.884	0.846	0.808	0.77	0.732	0.694	0.656	0.618
	0.15	0.91	0.872	0.834	0.796	0.758	0.72	0.682	0.644	0.606	0.568
	0.1	0.86	0.822	0.784	0.746	0.708	0.67	0.632	0.594	0.556	0.518
0.05	0.81	0.772	0.734	0.696	0.658	0.62	0.582	0.544	0.506	0.468	
MixedBot	1	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	
	0.5	0.82	0.84	0.788	0.772	0.756	0.74	0.724	0.708	0.692	0.676
	0.45	0.77	0.754	0.738	0.722	0.706	0.69	0.674	0.658	0.642	0.626
	0.4	0.72	0.704	0.688	0.672	0.656	0.64	0.624	0.608	0.592	0.576
	0.35	0.67	0.654	0.638	0.622	0.606	0.59	0.574	0.558	0.542	0.526
	0.3	0.62	0.604	0.588	0.572	0.556	0.54	0.524	0.508	0.492	0.476
	0.25	0.57	0.554	0.538	0.522	0.506	0.49	0.474	0.458	0.442	0.426
	0.2	0.52	0.504	0.488	0.472	0.456	0.44	0.424	0.408	0.392	0.376
	0.15	0.47	0.454	0.438	0.422	0.406	0.39	0.374	0.358	0.342	0.326
	0.1	0.42	0.404	0.388	0.372	0.356	0.34	0.324	0.308	0.292	0.276
0.05	0.37	0.354	0.338	0.322	0.306	0.29	0.274	0.258	0.242	0.226	

References

1. Balla, R.K.; Fern, A. UCT for Tactical Assault Planning in Real-Time Strategy Games. In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, Pasadena, CA, USA, 14–17 July 2009; AAAI Press: Menlo Park, CA, USA, 2009; pp. 40–45.
2. Buro, M. Real-Time Strategy Games: A New AI Research Challenge. In Proceedings of the IJCAI, Acapulco, Mexico, 9–15 August 2003; Morgan Kaufmann: Burlington, MA, USA, 2003; pp. 1534–1535.
3. Shafi, K.; Abbass, H.A. A Survey of Learning Classifier Systems in Games. *IEEE Comput. Intell. Mag.* **2017**, *12*, 42–55. [[CrossRef](#)]
4. Synnaeve, G.; Bessiere, P. Multi-scale Bayesian modeling for RTS games: An application to StarCraft AI. *IEEE Trans. Comput. Intell. AI Games* **2015**, *8*, 338–350. [[CrossRef](#)]

5. Usunier, N.; Synnaeve, G.; Lin, Z.; Chintala, S. Episodic Exploration for Deep Deterministic Policies: An Application to StarCraft Micromangement Tasks. *arXiv* **2016**, arXiv:1609.02993.
6. Isaksen, A.; Gopstein, D.; Nealen, A. Exploring Game Space Using Survival Analysis. In Proceedings of the FDG, Pacific Grove, CA, USA, 22–25 June 2015.
7. Gottlob, G.; Greco, G.; Scarcello, F. Pure Nash Equilibria: Hard and Easy Games. *JAIR* **2005**, *24*, 357–406. [[CrossRef](#)]
8. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Eiben, A.E., Ed.; Springer: Berlin, Germany, 2003; Volume 53, p. 18.
9. Fister, I., Jr.; Yang, X.S.; Fister, I.; Brest, J.; Fister, D. A brief review of nature-inspired algorithms for optimization. *arXiv* **2013**, arXiv:1307.4186.
10. Yang, X.S. *Nature-Inspired Metaheuristic Algorithms*; Luniver Press: Beckington, UK, 2010.
11. Yang, X.S. *Nature-Inspired Optimization Algorithms*; Elsevier: London/Waltham, UK, 2014.
12. Biswas, A.; Mishra, K.; Tiwari, S.; Misra, A. Physics-Inspired Optimization Algorithms: A Survey. *J. Optim.* **2013**. [[CrossRef](#)]
13. Del Ser, J.; Osaba, E.; Molina, D.; Yang, X.S.; Salcedo-Sanz, S.; Camacho, D.; Das, S.; Suganthan, P.; Coello, C.; Herrera, F. Bio-inspired computation: Where we stand and what's next. *SWEVO* **2019**, *48*. [[CrossRef](#)]
14. Karaboga, D.; Basturk, B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *J. Glob. Optim.* **2007**, *39*, 459–471. [[CrossRef](#)]
15. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [[CrossRef](#)]
16. Goldberg, D.E. Genetic algorithms in search. In *Optimization, and Machine Learning*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1989.
17. Wang, G.G.; Deb, S.; Cui, Z. Monarch Butterfly Optimization. *Neural Comput. Appl.* **2015**, *31*, 1995–2014. [[CrossRef](#)]
18. Jin, N.; Rahmat-Samii, Y. Advances in Particle Swarm Optimization for Antenna Designs: Real-Number, Binary, Single-Objective and Multiobjective Implementations. *IEEE Trans. Antennas Propag.* **2007**, *55*, 556–567. [[CrossRef](#)]
19. Santucci, V.; Milani, A.; Caraffini, F. An Optimisation-Driven Prediction Method for Automated Diagnosis and Prognosis. *Mathematics* **2019**, *7*, 1051. [[CrossRef](#)]
20. Yeoh, J.M.; Caraffini, F.; Homapour, E.; Santucci, V.; Milani, A. A Clustering System for Dynamic Data Streams Based on Metaheuristic Optimisation. *Mathematics* **2019**, *7*, 1229. [[CrossRef](#)]
21. Hendriks, M.; Meijer, S.; Van Der Velden, J.; Iosup, A. Procedural content generation for games: A survey. *ACM Trans. Multimed. Comput. Commun. Appl.* **2013**, *9*, 1–22. [[CrossRef](#)]
22. Wilson, D.G.; Cussat-Blanc, S.; Luga, H.; Miller, J.F. Evolving simple programs for playing Atari games. *Proc. Genet. Evol. Comput. Conf.* **2018**, 229–236. [[CrossRef](#)]
23. Ponticorvo, M.; Rega, A.; Di Ferdinando, A.; Marocco, D.; Miglino, O. Approaches to Embed Bio-inspired Computational Algorithms in Educational and Serious Games. In Proceedings of the CAID@ IJCAI, Melbourne, Australia, May 2017.
24. Woźniak, M.; Połap, D.; Napoli, C.; Tramontana, E. Application of Bio-Inspired Methods in Distributed Gaming Systems. *ITC* **2017**, *46*. [[CrossRef](#)]
25. Boskovic, B.; Greiner, S.; Brest, J.; Zumer, V. A differential evolution for the tuning of a chess evaluation function. In Proceedings of the 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 1851–1856.
26. Diaz, G.; Iglesias, A. Evolutionary Behavioral Design of Non-Player Characters in a FPS Video Game through Particle Swarm Optimization. In Proceedings of the 13th International Conference on SKIMA, Island of Ulkulhas, Ulkulhas, Maldives, 26–28 August 2019; pp. 1–8. [[CrossRef](#)]
27. Kuhlmann, G.; Stone, P. Automatic Heuristic Construction in a Complete General Game Player. *AAAI Conf.* **2006**, *6*, 1456–1462.
28. Joppen, T.; Strubig, T.; Furnkranz, J. Ordinal Bucketing for Game Trees using Dynamic Quantile Approximation. In Proceedings of the IEEE CoG, London, UK, 20–23 August 2019; pp. 1–8. [[CrossRef](#)]
29. Borovikov, I.; Zhao, Y.; Beirami, A.; Harder, J.; Kolen, J.; Pestrak, J.; Pinto, J.; Pourabolghasem, R.; Chaput, H.; Sardari, M.; et al. Winning isn't everything: Training agents to playtest modern games. In Proceedings of the AAAI Workshop on Reinforcement Learning in Games, Honolulu, HI, USA, 27 January–1 February 2019.

30. Naves, T.; Lopes, C. One Approach to Determine Goals in RTS Games Using Maximization of Resource Production with Local Search and Scheduling. In Proceedings of the ICTAI, Vietri sul Mare, Italy, 9–11 November 2015; pp. 469–477. [\[CrossRef\]](#)
31. Bosc, G.; Tan, P.; Boulicaut, J.F.; Raissi, C.; Kaytoue, M. A Pattern Mining Approach to Study Strategy Balance in RTS Games. *IEEE T-CIAIG* **2015**, *9*, 123–132. [\[CrossRef\]](#)
32. Uriarte, A.; Ontañón, S. Combat Models for RTS Games. *IEEE TOG* **2018**, *10*, 29–41. [\[CrossRef\]](#)
33. Rogers, K.; Skabar, A. A Micromanagement Task Allocation System for Real-Time Strategy Games. *IEEE TCIAIG* **2014**, *6*, 67–77. [\[CrossRef\]](#)
34. Kawase, K.; Thawonmas, R. Scout of the route of entry into the enemy camp in StarCraft with potential field. In Proceedings of the GCCE, Tokyo, Japan, 1–4 October 2013; pp. 318–319. [\[CrossRef\]](#)
35. Cunha, R.; Chaimowicz, L. An Artificial Intelligence System to Help the Player of Real-Time Strategy Games. In Proceedings of the SBGames, Florianopolis, Brazil, 8–10 November 2010; pp. 71–81. [\[CrossRef\]](#)
36. Ontañón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; Preuss, M. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE T-CIAIG* **2013**, *5*, 293–311. [\[CrossRef\]](#)
37. Zhao, Y.; Borovikov, I.; Beirami, A.; Rupert, J.; Somers, C.; Harder, J.; De Mesentier Silva, F.; Kolen, J.; Pinto, J.; Pourabolghasem, R.; et al. Winning Isn't Everything: Enhancing Game Development with Intelligent Agents. In Proceedings of the AAAI Workshop on Reinforcement Learning in Games, Honolulu, HI, USA, 27 January–1 February 2019.
38. Guerrero-Romero, C.; Lucas, S.; Perez Liebana, D. Using a Team of General AI Algorithms to Assist Game Design and Testing. In Proceedings of the IEEE Conference on CIG, Maastricht, The Netherlands, 14–17 August 2018; pp. 1–8. [\[CrossRef\]](#)
39. Jaffe, A.B. Understanding Game Balance with Quantitative Methods. Ph.D. Thesis, University of Washington, Washington, DC, USA, 2013.
40. Risi, S.; Preuss, M. From Chess and Atari to StarCraft and Beyond: How Game AI is Driving the World of AI. *KI-Künstliche Intell.* **2020**, *34*, 1–11. [\[CrossRef\]](#)
41. Perrotta, C.; Bailey, C.; Ryder, J.; Haggis-Burridge, M.; Persico, D. Games as (Not) Culture: A Critical Policy Analysis of the Economic Agenda of Horizon 2020. *Games Cult.* **2019**. [\[CrossRef\]](#)
42. Salazar, M.G.; Mitre, H.A.; Olalde, C.L.; Sánchez, J.L.G. Proposal of Game Design Document from software engineering requirements perspective. In Proceedings of the Conference on CGAMES, Louisville, KY, USA, 30 July–1 August 2012; pp. 81–85.
43. Holmgård, C.; Green, M.C.; Liapis, A.; Togelius, J. Automated Playtesting with Procedural Personas through MCTS with Evolved Heuristics. *IEEE Trans. Games* **2018**, *11*, 352–362. [\[CrossRef\]](#)
44. Chaslot, G.; Bakkes, S.; Szita, I.; Spronck, P. Monte-Carlo Tree Search: A New Framework for Game AI. In Proceedings of the AAAI Conference on AIIDE, Palo Alto, CA, USA, 22–24 October 2008.
45. Heintz, S.; Law, E. Digital Educational Games: Methodologies for Evaluating the Impact of Game Type. *ACM Trans. Comput. Hum. Interact.* **2018**, *25*, 1–47. [\[CrossRef\]](#)
46. Walfisz, M.; Zackariasson, P.; Wilson, T. Real-Time Strategy: Evolutionary Game Development. *Bus. Horiz.* **2006**, *49*, 487–498. [\[CrossRef\]](#)
47. Sicart, M. Defining Game Mechanics. *Int. J. Comput. Game Res.* **2008**, *8*.
48. Wilson, K.; Bedwell, W.; Lazzara, E.; Salas, E.; Burke, S.; Estock, J.; Orvis, K.; Conkey, C. Relationships Between Game Attributes and Learning Outcomes: Review and Research Proposals. *Simul. Gaming* **2008**, *40*, 217–266. [\[CrossRef\]](#)
49. Erickson, G.; Buro, M. Global state evaluation in StarCraft. In Proceedings of the AAAI Conference on AIIDE, Raleigh, NC, USA, 3–7 October 2014; pp. 112–118.
50. Ludwig, J.; Farley, A. Examining Extended Dynamic Scripting in a Tactical Game Framework. In Proceedings of the Conference on AIIDE, Palo Alto, CA, USA, 14–16 October 2009.
51. Aly, M.; Aref, M.; Hassan, M. Dimensions-based classifier for strategy classification of opponent models in real-time strategy games. In Proceedings of the IEEE Seventh ICICIS, Cairo, Egypt, 12–14 December 2015; pp. 442–446.
52. Bangay, S.; Makin, O. Generating an attribute space for analyzing balance in single unit RTS game combat. In Proceedings of the IEEE Conference on CIG, Dortmund, Germany, 26–29 August 2014; pp. 1–8. [\[CrossRef\]](#)
53. Cho, H.; Park, H.; Kim, C.Y.; Kim, K.J. Investigation of the Effect of “Fog of War” in the Prediction of StarCraft Strategy Using Machine Learning. *Comput. Entertain.* **2017**, *14*, 1–16. [\[CrossRef\]](#)

54. Mishra, K.; Ontañón, S.; Ram, A. Situation Assessment for Plan Retrieval in Real-Time Strategy Games. In *Advances in Case-Based Reasoning, ECCBR 2008*; Althoff, K.D., Bergmann, R., Minor, M., Hanft, A., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Germany, 2008; Volume 5239, pp. 355–369.
55. Togelius, J.; Preuss, M.; Hochstrate, N.; Wessing, S.; Hagelbäck, J.; Yannakakis, G. Multiobjective exploration of the StarCraft map space. *IEEE Conf. CIG* **2010**, *1*, 265–272. [[CrossRef](#)]
56. Lin, M.; Wang, T.; Li, X.; Liu, J.; Wang, Y.; Zhu, Y.; Wang, W. An Uncertainty-Incorporated Approach to Predict the Winner in StarCraft II Using Neural Processes. *IEEE Access* **2019**, *7*, 101609–101619. [[CrossRef](#)]
57. Tong, C.; On, C.; Teo, J.; Chua, B.L. Automatic generation of real time strategy tournament units using differential evolution. In Proceedings of the IEEE CSUDET, Semenyih, Malaysia, 20–21 October 2011; pp. 101–106. [[CrossRef](#)]
58. Long, M. Radio General: A Real-Time Strategy Game Where You Cannot See Your Units. In Proceedings of the Annual Symposium on CHI PLAY, Melbourne, Australia, 28–31 October 2018; pp. 345–351. [[CrossRef](#)]
59. Li, Y.; Li, Y.; Zhai, J.; Shiu, S. RTS game strategy evaluation using extreme learning machine. *Soft Comput.* **2012**. [[CrossRef](#)]
60. Si, C.; Pisan, Y.; Tan, C.T. A Scouting Strategy for Real-Time Strategy Games. *Conf. Interact. Entertain.* **2014**, 1–8. [[CrossRef](#)]
61. McCoy, J.; Mateas, M. An Integrated Agent for Playing Real-Time Strategy Games. *AAAI Conf. AI* **2008**, *8*, 1313–1318.
62. DeRouin-Jessen, R. Game on: The Impact of Game Features in Computer-Based Training. Ph.D. Thesis, University of Central Florida, Orlando, FL, USA, 2008.
63. Novak, D.; Čep, A.; Verber, D. Classification of modern real-time strategy game worlds. *GSTF J. Comput.* **2018**, *6*. [[CrossRef](#)]
64. Microrts. Available online: <https://github.com/santiontanon/microrts> (accessed on 20 March 2020).
65. TiamatBot. Available online: <https://github.com/jr9Hernandez/TiamatBot> (accessed on 20 March 2020).
66. MixedBotmRTS. Available online: <https://github.com/AmoyZhp/MixedBotmRTS> (accessed on 20 March 2020).
67. Evolutionary Action-Abstractions. Available online: <https://github.com/julianmarino/evolutionary-action-abstractions> (accessed on 15 April 2020).
68. Mariño, J.; De Oliveira Moraes Filho, R.; Toledo, C.; Lelis, L. Evolving Action Abstractions for Real-Time Planning in Extensive-Form Games. *AAAI Conf. AI* **2019**, *33*, 2330–2337. [[CrossRef](#)]
69. Ontanon, S.; Barriga, N.A.; Silva, C.; De Oliveira Moraes Filho, R.; Lelis, L. The First microRTS Artificial Intelligence Competition. *AI Mag.* **2018**, *39*, 75. [[CrossRef](#)]
70. Churchill, D.; Saffidine, A.; Buro, M. Fast Heuristic Search for RTS Game Combat Scenarios. In Proceedings of the AAAI Conference on AIIDE, Stanford, CA, USA, 8–12 October 2012.
71. Barriga, N.A.; Stanescu, M.; Buro, M. Game Tree Search Based on Non-Deterministic Action Scripts in Real-Time Strategy Games. *IEEE TCIAIG* **2017**. [[CrossRef](#)]
72. Ontanon, S. The combinatorial Multi-armed Bandit problem and its application to real-time strategy games. In Proceedings of the Conference on AIIDE, Boston, MA, USA, 14–18 October 2013; pp. 58–64.
73. De Oliveira Moraes Filho, R.; Mariño, J.; Lelis, L.; Nascimento, M. Action Abstractions for Combinatorial Multi-Armed Bandit Tree Search. In Proceedings of the Conference on AIIDE, Edmonton, AB, Canada, 13–17 November 2018.
74. Barriga, N.A.; Stanescu, M.; Buro, M. Combining Strategic Learning and Tactical Search in Real-Time Strategy Games. In Proceedings of the AAAI Conference on AIIDE, Snowbird, UT, USA, 5–9 October 2017.
75. Stanley, K.; Bryant, B.; Miikkulainen, R. Real-Time Neuroevolution in the NERO Video Game. *IEEE TEVC* **2005**, *9*, 653–668. [[CrossRef](#)]

