A Competitive Combat Strategy and Tactics in RTS Games AI and StarCraft

Adil Khan[1(✉)], Kai Yang[2], Yunsheng Fu[3], Fang Lou[3], Worku Jifara[1], Feng Jiang[1], and Liu Shaohui[1]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, People's Republic of China
adil.adil25@yahoo.com, worku.jifara@gmail.com, {fjiang,shliu}@hit.edu.cn

[2] Chinese People's Liberation Army Aviation School, Beijing, People's Republic of China
yangkai4545@163.com

[3] Institute of Computer Applications, China Academy of Engineering Physics, Mianyang, People's Republic of China
Fyun@yahoo.com, Lfang@163.com

Abstract. This paper presents a competitive combat strategy and tactics in RTS Games AI. To put it simply, if a player is building up base, he is losing out on creating an army and If he is building up his army, he is losing out on having a strong base. The key to winning, in StarCraft or any other RTS game is to balance strategy, tactics, macro and micro. To improve the game, one has to be able to keep track of everything that's going on over the entire map. And one must be able to give orders quickly and efficiently so in this paper we propose a competitive battle strategy with the help of a plot and decision tree. We simulate the strategy in MicroRTS developed in java EE by conducting a game-play between human player and MicroRTS AI (Game AI), though our proposed strategy outperforms the Game AI rarely as we did not account game playing-speed that makes a huge difference in victory but at least we succeeded in introducing a strategy that could well compete the Game AI and may defeat it but rarely.

Keywords: Combat strategy · Game AI · MicroRTS · Macro management Real time strategy games · StarCraft

1 Introduction

Real time Strategy games are combat games in which a player try to dominate or destroy the opponent as quick as possible by managing and controlling resources, tactics and strategies [1]. Real time strategy is a sub-genre of strategy video games where the games do not progress or play incrementally [2]. The term 'Real Time Strategy' was first introduced in a magazine called BYTE in 1982 and the first game of this genre was Dune II developed by Westwood Studio and released by Virgin Interactive in 1992 followed by their seminal command & Conquer in 1995 [3]. On March 31, 1998 StarCraft was released for windows which was the first game in the StarCraft series based on science fiction real time strategy game and since then it grew in popularity among public in the

B. Zeng et al. (Eds.): PCM 2017, LNCS 10736, pp. 3–12, 2018.
https://doi.org/10.1007/978-3-319-77383-4_1

market [4]. Now this game is being played every day by millions of players or users and dominating and surpassing all other RTS games of this genre [5]. A 'strategy game' is a game in which the players intimidate and most often make independent strategic and tactical decisions that have a high importance in determining the outcome of the game. Almost all strategy games oblige internal decision tree style thinking, and typically very high situational consciousness or awareness [6].

Multiplayer online battle arena (MOBA) is a sub-genre of real time strategy video games in which the objective is to dominate and destroy the enemy's main structure like Bases, units, buildings and other resources [7]. Players typically have several abilities and benefits that improve over the progression of a game and that contribute in destroying the opponent team's overall strategy. Defense of the Ancients or "DotA" is one of the MOBA game. Players use commanding units known as heroes, and are supported by allied heroes and AI-controlled combatants called 'creeps'. Some of the best real time strategy games are StarCraft II: Wings of Liberty, StarCraft II: Legacy of the Void, Rome: Total War, Company of Heroes, Warcraft III: Reign of Chaos, Rise of Nations etc. We studied other related papers in the field that were either found outdated and specific to Some traditional strategic concepts or found with complicated and hybrid tactical and strategic techniques only, that is why we thought to write a paper with a unique and simple strategy or tactics pursuing updated information and literature, so this paper aims to introduce a simple but an efficient and competitive strategy for RTS games in the perspective of the current and recent past research and state of the arts in RTS games and StarCraft. It is organized as follows. Section 2 explains related work on RTS Games-AI and StarCraft. Section 3 presents a simple RTS game strategy with its decision tree. Section 4 includes experiments and implementation of the introduced strategy and Sect. 5 concludes the paper.

2 Related Work

Real time strategy games are fast paced combat games that were first introduced in 1990's and since then grew in a great popularity. RTS games face many challenges in AI research such as RTS games are played on large maps on which large numbers of units move around under player control collecting resources, constructing buildings, scouting, and attacking opponents with the goal of destroying all enemy buildings but this renders traditional full width search infeasible [8]. Combat strategy or map exploration is a common discovery oriented activity that players perform in modern video games. Exploration is a game mechanism that players need to master in order to collect resources and advance in the game [9]. In every RTS game, time is the most valuable resource before thinking about other resources. A player needs to keep himself aware of the time that he loses. It suggests that time is an important factor in composing a combat strategy always because if the opponent is faster so then it does not matter how clever the player is or how carefully the player base defenses and army compositions are [10], this also does not mean that speed trumps intelligence in RTS games but the speed is the foundation and backbone of an effective strategy. Along with speed, an effective combat strategy must need to be well composed of balanced, smart planning and responsive tactics [11]. According to the research community in RTS games, speed is more worthwhile than accuracy because in RTS games a player too often cannot afford to spend enough time in deciding where to place or manage a building, base or any other resource [12]. Developing computer controlled clusters to engage in fight, manage the usage of restricted resources, and form units and buildings in real time strategy games is a unique presentation in game-AI. But, tightly controlled online commercial game carry challenges to researchers interested in observing player's behavior and activities, constructing player's strategy models, and developing practical AI technology in them [13]. In place of setting up new programming settings or building a huge extent of agent's decision instructions by player's experience for directing real time AI research, the authors used reruns or replays of the commercial RTS game StarCraft to assess human player activities and to develop an intelligent system to learn humanoid decisions and performances. A case based reasoning approach was applied for the objective of training the system to learn and guess player strategies [14]. The researcher's analysis demonstrated that the suggested system is capable of learning and expecting individual player strategies, and that players provide signs of their private features through their building production orders.

3 Proposed Strategy and Decision Tree

In this section we present a competitive combat game strategy followed by its Decision tree with details as follows.

3.1 A Combat Strategy

Though it is not easy to program a game while planning any strategy for the first time without those who know playing RTS games like StarCraft. We thought over different strategies starting from rushes to complex micromanagement strategies. After all a simple Macro strategy was decided which is to out resource the enemy (opponent) by collecting the resources quickly than them that may result in victory in a long run. During the design, obviously we had a focus on worker units because of their certain role in success. During designing the units are required to act according to the strategy in finding and taking the remote resources to the base, if there is not any base near then it would start building one. Attacking units normally find the target and attack but sometimes workers will defend if they needed to, besides the bases and barrack will also keep producing attackers and workers to combat. We thought of a strong defense supporting the rush strategy by introducing the defense team that will guard the bases and will wait targeting until someone comes close. A percentage of defenders are assigned to attacking units called 'Assaults' to attack enemy when they are required or when they exceed a specific size or number. During units' production the ratio of defenders and assault is always kept balance. Also with the passage of time new units are assigned to assault team as the force grows larger. At least one of the three attack units i.e. Light, heavy and ranged would be opted to build at a specific time. For the proposed strategy we decided to work with a decision Tree, being one of the simplest, easiest to implement, but most importantly quickly adaptable. With a decision Tree, we would have no problem adding new nodes, and thus decisions and actions, our AI could take advantage off. We present the process of mapping out actions in the form of a tree or prototype developed in Visio 2016, to record the prototype. It let us easily set up, move around nodes. Upon beginning work on the base code for our AI, we needed two things. A base line tree code to build nodes, and some sort of knowledge base to keep all of the information our AI knew about the game, along with all other game settings. Our AI thus updates the knowledge base every frame before going through the decision tree to get unit actions (Fig. 1).

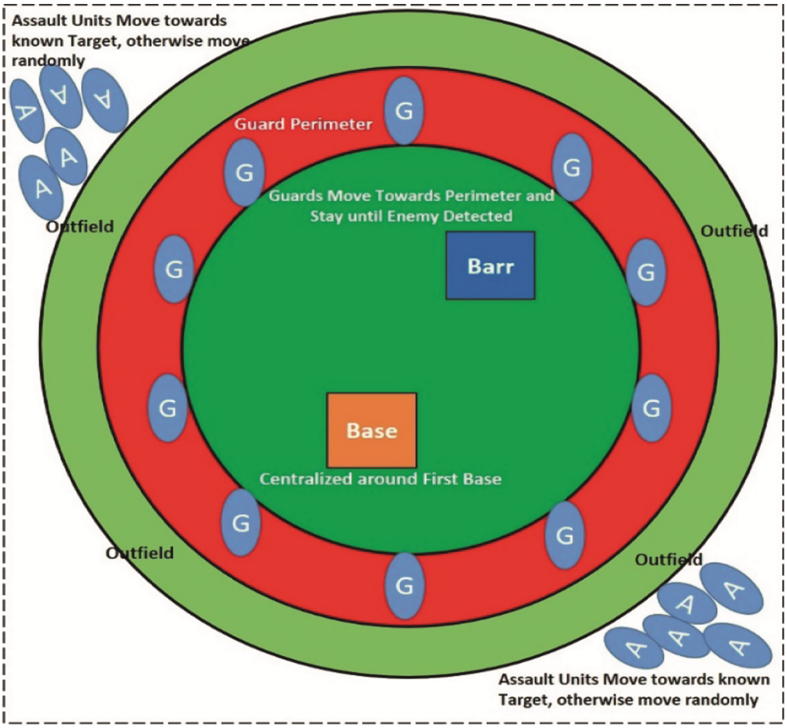

Fig. 1. An RTS combat game AI strategy

3.2 An RTS Combat Game Strategy Decision Tree

The tree consists of roughly 4 mini trees: Barracks, Base, Attack Units, and Workers. Each of these trees controls the specific type of unit. The 'Base' and 'Barracks' trees are roughly equivalent. The 'Base' simply checks if a worker cap has reached and to decide whether workers are to be build. The 'Barrack' is a little different, in so that the 'Barrack' first checks to see if the system will let build a particular unit, and if so, uses a random selection using weights. Each unit type has a weight associated with it in the 'Knowledge Base', and if the randomizer selects the range for that weight it will build that unit. For the attacking unit tree, it checks to see what team the unit is on, 'Assault' or 'Guard'. If it is an 'Assault Unit', it sees if it can attack an enemy, and if it can't it tries to move toward one if it has a target. If it has no target, it randomly moves. If it is a guard unit, it looks to see if there is a target and if there is, it moves and attack. If a guard unit does not have a target, they move towards the guard perimeter, and randomly move inside the perimeter. The 'Worker' has the most complex tree. It first checks to see if it needs to defend the base; if so, then it defends. Otherwise it checks if it's on the build or gather team. If the unit is on the build team, it checks to see if there aren't enough barracks, and if not, find space to build one (using a BFS) and builds one. Once the barracks cap has been reached it is reassigned to the gather team. The gather team first checks to see if the unit is holding a resource. If it is holding a resource, then it checks if a base is close enough to deposit it, and if it is, it will, otherwise it builds a new base next to resources. If it isn't holding resources, it tries to find the closest resources (using a BFS) and move towards them. Unfortunately, the breadth first search on this resource finding does not provide an optimal macro level solution. It does not factor in the amount of the resource (Fig. 2).

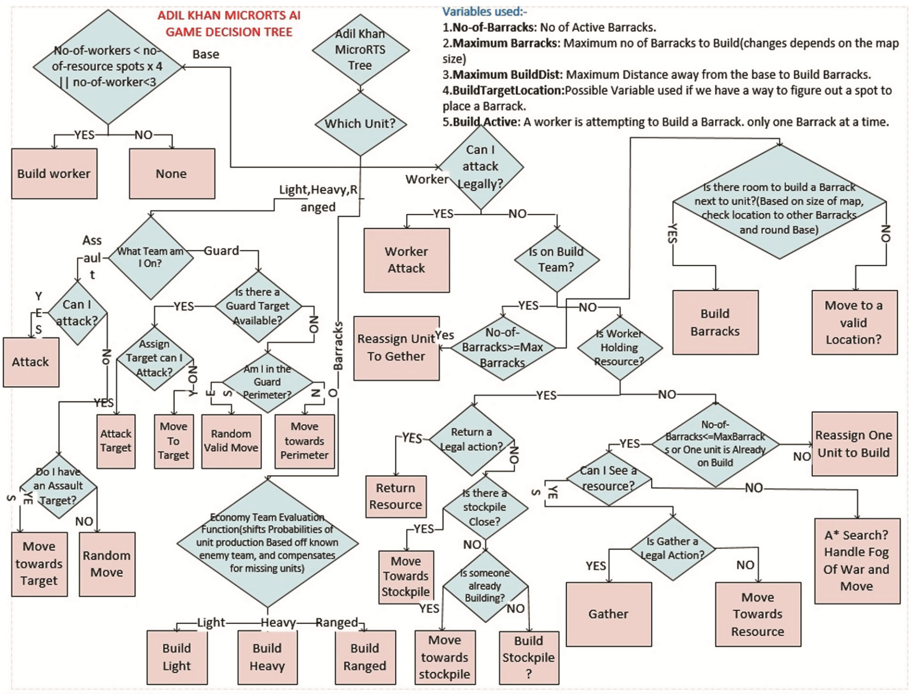

Fig. 2. An RTS combat game decision tree

4 Implementation/Experiment

We implemented our strategy in MicroRTS which is a simple RTS game designed for testing ‘AI’ procedures that provides the necessary features of an RTS game. MicroRTS supports six units and three building types each represent one tile. MicroRTS support configurable map sizes ranging from 8×8 to 32×32 in published publications. The game GUI and definitions of the unit types or resources types are notable in Fig. 3. Besides the unit and building types mentioned in Fig. 3, there are some other units with different specialties and buildings that we used in the MicroRTS like birds, ‘sky archers’, (air units), soldier offices and airports. We checked the performance of our introduced strategy against the built-in MicroRTS Game-AI by conducting a match between a human player (authors) and system AI where our proposed strategy shown a competitive performance against the Built-in Game-AI and even our strategy appeared successful in defeating the Game-AI but rarely because sometimes the system (Game AI) used to control and manage the resources, micro, macro and time efficiently as shown in the Fig. 4. Player ‘0’ is the GAME-AI player and Player ‘1’ is the human player (authors). We include the GUI of one of the match played between player ‘0’ and player ‘1’ where player ‘0’ scored (1074), resources (40,235), kills (3) with Deaths (1), worker (28), Heavy (10), bases (4) and soldier office (2), where in response player ‘1’ scored (415), resources (85, 111), kills only (1) with maximum deaths (3), light units (1), workers (6), heavy units (1), bases (1) and soldier office (1). Further during the match, the GUI symbols like ‘H’ represents heavy units, ‘W’ represents worker units and ‘B’ represents Bases. However, during the play according to the situations and tactics required, some units and buildings are not build or consumed shortly after building during the battle (Fig. 5).

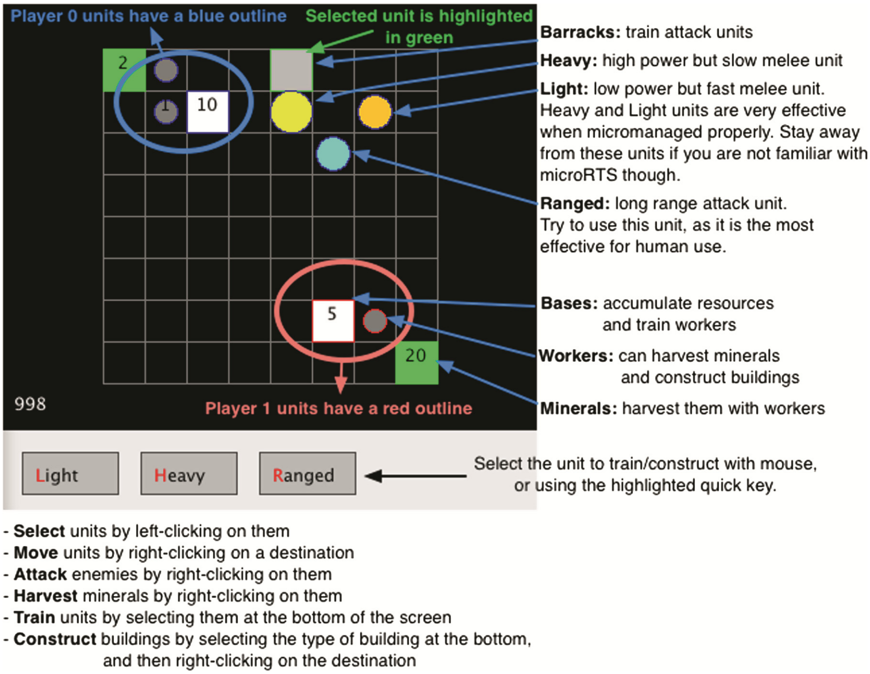

Fig. 3. A MicroRTS GUI explaining numerous units or resource types [15].

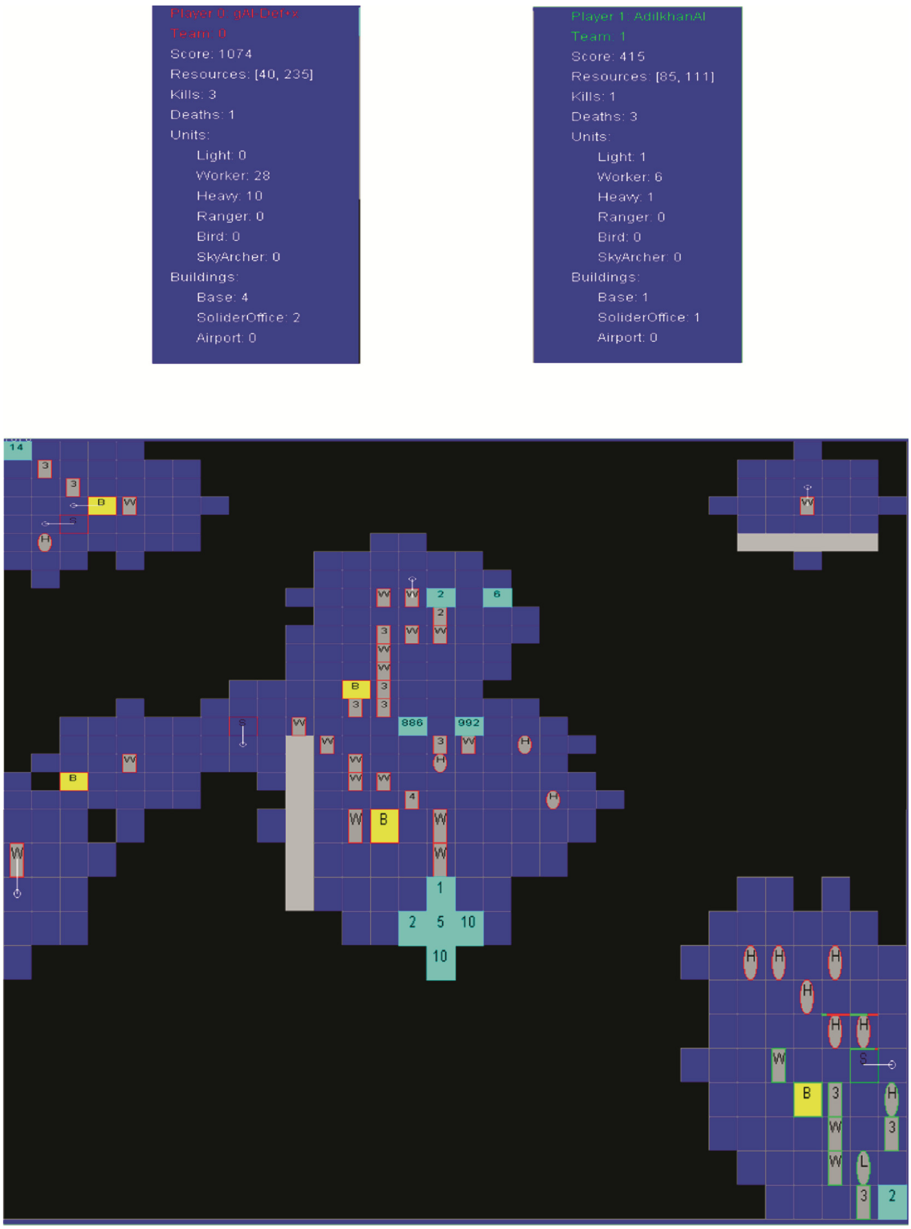

Fig. 4. Match played between human player and game AI (clear and visible GUI but in parts)

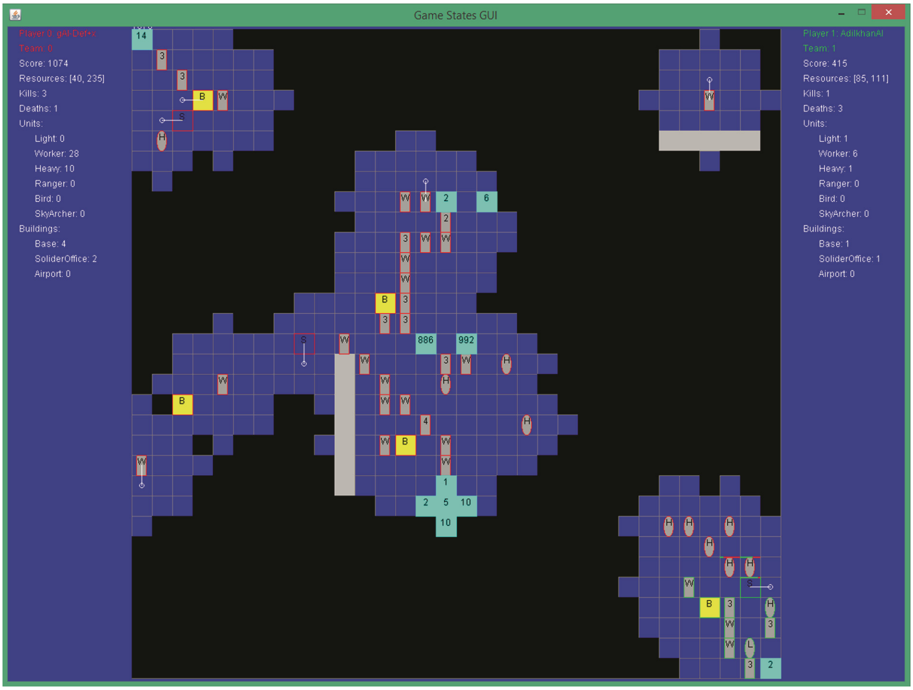

Fig. 5. Match played between human player and game AI (original GUI but adjusted)

5 Conclusion and Future Work

In this paper we introduced a competitive combat strategy for real time strategy games and StarCraft by implementing it in MicroRTS and played between human player and computer AI (Game-AI) that showed to be an efficient and a competitive strategy as far as real time performance is concerned during Game-play and observations but regardless of game-playing speed. In future, we are thinking to implement this strategy using a full-fledged game like Wargus or StarCraft using BWAPI, as MicroRTS is a small implementation of an RTS game, designed to perform AI research that initially helps to quickly test theoretical ideas, before moving on to full-fledged RTS games.

Author profile. Adil Khan received C.T. from AIOU Islamabad Pakistan, MCSE from Microsoft, CCNA from Cisco, B.Ed from the University of Peshawar, BS Honors in Computer Science from Edwards college Peshawar, M.S in Computer Science from City University of Science and Information Technology Peshawar, and PhD from HIT, Harbin, China. In 2014-2016, he was a senior Lecturer in Higher Education Department KPK, Pakistan. He has published many publications in top-tier academic journals and conferences including IEEE and Springer. Currently, he is working as a Research Scientist in Game AI at the School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001 PR China. He is interested in Artificial Intelligence, Game

AI, Neural networks, RTS Games, FPS Games, Machine Learning. He can be reached at personal E-mails: adil.adil25@yahoo.com, Dradil@hit.edu.cn

Acknowledgement. Thanks for the support provided by CSC and Department of Computer Science, HIT Harbin, China.

References

1. Robertson, G., Watson, I.: A review of real-time strategy game AI. AI Mag. **35**(4), 75–104 (2014)
2. Si, C., Pisan, Y., Tan, C.T.: A scouting strategy for real-time strategy games, pp. 1–8 (2014)
3. Ontanón, S., et al.: A survey of real-time strategy game AI research and competition in StarCraft. IEEE Trans. Comput. Intell. AI Games **5**(4), 293–311 (2013)
4. Preuss, M., et al.: Reactive strategy choice in StarCraft by means of fuzzy control. In: 2013 IEEE Conference on Computational Intelligence in Games (CIG). IEEE (2013)
5. Farooq, S.S., et al.: StarCraft AI competition: a step toward human-level AI for real-time strategy games. AI Mag. **37**(2), 102–107 (2016)
6. Stanescu, M., et al.: Predicting army combat outcomes in StarCraft. In: AIIDE. Citeseer (2013)
7. Waltham, M., Moodley, D.: An analysis of artificial intelligence techniques in multiplayer online battle arena game environments. In: Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists. ACM (2016)
8. Barriga, N.A., Stanescu, M., Buro, M.: Building placement optimization in real-time strategy games. In: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference (2014)
9. Si, C., Pisan, Y., Tan, C.T.: Understanding players' map exploration styles. In: Proceedings of the Australasian Computer Science Week Multiconference. ACM (2016)
10. Barriga, N.A., Stanescu, M., Buro, M.: Game tree search based on non-deterministic action scripts in real-time strategy games. IEEE Trans. Comput. Intell. AI Games, **PP**(99), 1 (2017). 10.1109/TCIAIG.2017.2717902
11. Stanescu, M., et al.: Evaluating real-time strategy game states using convolutional neural networks, September 2016
12. Chen, W., et al.: GameLifeVis: visual analysis of behavior evolutions in multiplayer online games. J. Vis. **20**, 651–665 (2017)
13. Lara-Cabrera, R., Cotta, C., Fernandez-Leiva, A.J.: A review of computational intelligence in RTS games. In: 2013 IEEE Symposium on Foundations of Computational Intelligence (FOCI), pp. 114–121 (2013)
14. Sourmelis, T., Ioannou, A., Zaphiris, P.: Massively multiplayer online role playing games (MMORPGs) and the 21st century skills: a comprehensive research review from 2010 to 2016. Comput. Hum. Behav. **67**, 41–48 (2017)
15. Ontanón, S.: The combinatorial multi-armed bandit problem and its application to real-time strategy games. In: Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. AAAI Press (2013)