

Artificial Immune based Bot for planetWar game

Mohamed Amine CHIKH TOUAMI
and SALEM Mohammed
Faculty of Exact Sciences,
University Mustapha Stembouli
Mascara, Algeria
Email: ma_chikhtouhami@hotmail.com

KHELFI Mohammed Faycal
RIIR Lab
University of Oran 1, Ahmed Ben Bella, Oran, Algeria
Email: mf_khelfi@yahoo.fr

Abstract—Planet war is a strategy game which requires the bot to reach some adaptabilities with the aim to beat different enemies in different maps. This paper will represent an evolutionary autonomous agent based on Artificial Immune system (AIS) to evolve and define a set of rules for the bot to achieve some goals in the game. AIS is applied to tune multiple parameters to define the bots behavior within the game. The obtained evolutionary bot is compared to multiple opponents in different situations giving encouraging results.

I. INTRODUCTION

Strategy video game is to make a decision under a set of units during the game. A real-time Strategy game is a sub-genre of strategy game, where players move (control multiple structures) in real time. Moreover, RTS game is a complex competitive domain where allowed to virtualize battles against players with a purpose of destroying other enemies units or to generate more self-units to secure an area.[1]

The complex nature of these games makes their conception more challenging because :

- Their real-time nature: the player has to move without waiting for other players to act.
- The player has to control a large number of units simultaneously.
- Make a decision under uncertainty: partial observability, time need to move, units need to react, ...etc.

Recent development in the field of RTS video games led to making the paradigm of Artificial Intelligence as a key instrument in the designing of the agents(bots), that will react in a real scenario against other players during the game. In addition, the RTS games posing new challenges to AI and it is a wide area to test AI concepts and methodologies.[1], [5], [7], [3], [4]

The bot's design makes a significant challenge to RTS games conception process. A bot is an autonomous agent act with human player in a real scenario under any computer-based framework. [4] In the other hand, the huge progress of video games and their international competitions, it's becoming extremely difficult to ignore the existence of human expert players and their experience within the conception of bot's behavior. While, including expert experimentation under conception process will make the bot more challenging and hard to beat that implies more challenging game and more interesting.[3]

Google Artificial Intelligence (GAIC) Challenge is one of the international RTS game competition that was sponsored by Google Community in 2010. In this competition, Google chooses planet war as a competitive RTS game. While this paper will study planet war game as a subject. Planet war is an RTS game which simulates battles against autonomous agents under several situations (maps) with the aim of winning the game by owning the maximum number of planets.

The bot concept plays a critical role in RTS game progression, designing a challenging bot mean designing a challenging game. In case of planet war, a set of pre-determined parameters need to make the bot under real game. Moreover, the parameters selection will affect the behavior of the bot later, so, good selection process led to the better reaction during the game. In the other hand, the previous parameters it's a large research area that treated as an optimization itself.[5]. However, the selection process is trying to find the most important parameters for best bot performance[5].

The specific objective of this study was to design a planet war bot using an Evolutionary Algorithm named Artificial Immune System (AIS) to determine parameters weights (values) by maximizing an objective function based on turns number played by the bot and the number of wins.

The paper has been organized in the following way: First of all, this paper gives a brief description of planet war and all rules applied during the game. In the second section, AIS algorithm was defined. The third chapter concerned with the approach used for this study by defining the global architecture of the bot. In the remaining part, all experimental results are presented.

II. PLANET WAR

As we mentioned previously one of the international competition was organized by google community. In this paper, we worked with planet war game chosen as RTS game in this competition with the aim of designing a fighter bot. During a planet war[5], firstly you have to choose a map to run a match, each map has a number of distributed planets in different positions with a specific number emerged the starships amount situated in the planet. Each planet in the map owned by player, opponent or neutral (owned by no one), moreover, each planet owner by players will increase

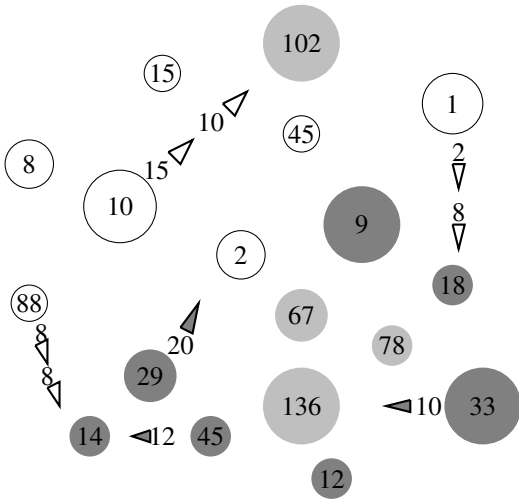


Fig. 1. an early stage of a planet wars game, as shown here there are planet categories, the darker grey (red in the game) ones related to the player, the white ones (green in the game) related to the opponent and the neutral planets are coloured by light grey. Fleets presented by triangles and the numbers are the starships included on them.

the number starship per turn based on their growth rate except the neutral planets that will not be able to generate new additional starships. The main purpose of planet war[5] is to compete with other players and try to beat all of them during the game by limited turns number.

Moreover, the planet war fighter bot face two important problems that make the agent's conception more challenging: The first problem [5] that the bot can not save any previous stats from previous turns as actions it made, map state...etc. Since in each turn, bot finds under an unknown map the same thing when it begins the game in the first time. The second problem[3] is inherited by the RTS game nature that is shown as the time required to act (make an action) is defined by 1 second.

The environment of planet war forced to assign properties to every single object in the game.[5] Such as, each planet in the map has two coordination points X and Y for its location, the ownerID, number of troops situated on it and the growth rate. To make an attack against enemies, the bot has to include starships and fleets under the attack. Each fleet has playerID, starships number included on it, source planetID, destination planetID and the number of turns need to achieve the goal, While the player is able to make one action per turn.[5]

After each action, the planets will increase their ships number based on their growth rate.[1] Moreover, if the source planet and the destination planet owner are the same ones, so we will consider that as a reinforcement by adding extra ships to the destination planet. Otherwise, if the target is a neutral planet, the bot has to calculate the exact starships need to win conquest the planet, whilst, if the target is an opponent, the two contenders will fight until one of them who has the highest number of starships will win the planet.[5], [1]

When the player makes an action and sends fleets, they can

not change their direction until they will reach the destination they sent for it. As we mentioned previously, each match has a limited number of turns cannot go throw. So, by the end of the match, the player with the highest number of ships win the game. If one of them lose all their ships without finishing their total turns number, the game will complete faster, whilst, if the two players have the same number of troops, we have a draw.[1].

III. IMMUNE SYSTEM OVERVIEW

The biology is one of the most inspirational domains that help the computational to solve many hard problems.[2] Among the natural concepts, the immune system is one of the important aspects that provide several technics to design many optimization algorithms. An optimization problem[6] is to search for the optimum solution from a set of potential solutions. The most algorithms used in this wide research are Heuristics.[6] That include divers algorithms such as firefly swarm optimization, artificial bee colony, genetic algorithm, just to mention a few. While the Artificial Immune system (AIS)[6] is a population-based algorithm. However, the application of the Artificial Immune System within technologies areas is a challenge is a self. Likewise, the researchers could exploit just a few immune's mechanisms because even in biology, this system still under a wide research area itself.

A. natural immune system

The biological immune system including human system contains cells, molecules, and organs in the global structure to defend the body against diseases.[6] The immune system has the ability to recognize the self-body cells and the nonself cells. In addition, this technic is very important to make an immune response by activating a suitable process to defeat the nonself cells.[6] This response process will be activated according to antigens type, every antigene type activate a specific immune response process. whilst, memory cells are produced and activate when there is the same immune response type later.

Clonal selection: To activate a response against the pathogens, the immune system uses multiple mechanisms. One of the interesting immune response is the clonal selection, this response type describes the process how the immune system can stimulus against a specific type of pathogens by proliferating a specific type of cells who can recognize the antigen.[2] When a clonal selection response was stimulated, a specific type of cells (the B-cells) be proliferating according to affinity maturation, that's mean the high affinity will generate a hight clone number. The clonel selection process pass with multiple stps:

- a) : The cloned cells undergo to a mutation process.
- b) : the self-reactive receptor will be eliminated.
- c) : proliferate the mature cells those can detect the antigen.

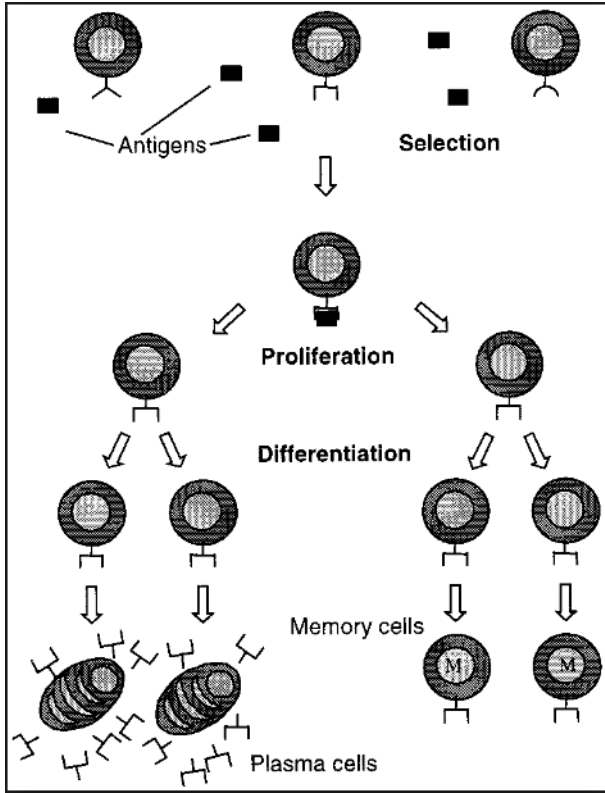


Fig. 2. The biological clonal selection mechanism and its steps in order to defend the body, starting by detecting of the antigen until removing it.

B. Artificial Immune System

In light of recent event in AIS, many algorithms have been developed trying to simulate the clonal selection process. In 2002 Castro and Zuben proposed an optimization algorithm named CLONALG. The first step of this algorithm is to initialize an N random antibodies present the N potential solutions. In each iteration, the antibodies are cloned, mutated and selected the best of them preparing for the next generation. The proliferation of the antibodies is according to their affinities. While the maturation applied to equation (1):

$$x_{id} = x_{id} + k(x_{d_{max}} - x_{d_{min}}) \cdot N(0, 1) \quad (1)$$

where x_{id} represent the dimension d of the antibody i , x_{max} and x_{min} represent the min and the max bounds of the variable i , $N(0,1)$ is the standard distribution and k is the scale factor.

New random antibodies will be added to the original population by replacing a percentage of worsting antibodies and preparing to next generation.

C. AIS pseudocode

- 1) initialize N individual with random values between 0 and 1.
- 2) For iteration = 1 : maxIteration,
 - Compute the affinities.
 - Clone the antibodies.
 - Mutate the antibodies cloned in the previous step.

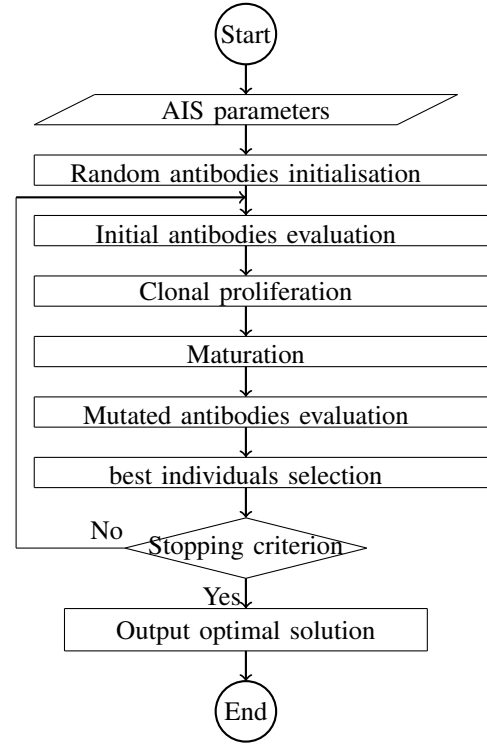


Fig. 3. The diagram showed the ordered steps for running an AIS algorithm based on clonal selection algorithm (CLONALG)

- Compute the affinities for the new antibodies mutated.
- Applied the selection process to select the next generation

End For

Display the optimal solutions.

End

IV. AISBOT: THE CANQUER BOT

To design planet war conquest bot, you will face two major problems as we said in previous sections. The primary constraint is related to time, the bot has just 1 second to make an action. The second main problem is the memory restriction, the bot couldn't store any pieces of information from previous turns. This paper attempt to study these restrictions that limit the design performance of the bot's behaviour by using a set of rules in order to optimize the decision engine.

Anyway, the bot will perform one single action during the game is to make fleets move around the map from a planet to another. before the decision engine makes the bot move, it should distinguish between the self and the non-self planet. based on this simple action the greatest challenge is to determine which planet will generate fleets, how many starships will be included on it and which planet will be attacked. Next, we will describe the parameters used in the optimization following by the technics governing the bot.

A. AisBot

In order to improve our experimentations, we used the AisBot as a fighter bot during planet war game and it works as follow: The first step, the bot tries to determine the base planet using score function. Next, the bot tries to find a planet to send a fleet to it, if the target is an enemy, so the action made by the bot considered as a conquest. If the target planet is a colony, the action its a reinforcement; while, if the target owned by neutral, the action is taken as expansion. The fleets take more time to reach the target planet and cannot change their direction. Once the base planet will be reinforced by the colony, the action will be considered as a Tith. In addition, the colony can make an attack against the target planet if it is closer instead of sending starships to the base planet with the aim of attacking the target. Besides that, if a planet has been targeted, the bot can not decide an additional attack against the same planet till ending the first one.

When the attack mode is an expansion, the bot can know the exact number of starships need to win the planet, while, if the attack mode is a conquest, the bot should estimate the number of troops needs to defeat the enemy and quest the planet.

As we mentioned previously, the purpose of this study is to try to optimize the behaviour of a bot during the planet war RTS game. In order to improve our experimentation, we use the AisBot as a conquer bot in this game, and it works as follows: first of all, when a turn is started, the bot tries to determine the base planet based on a score function and the rest of planets are used as colonies. Secondly, the bot decides which planet to attack for the next turn or if it owns additional planet the action will be considered as a reinforcement, in order to reach the target planet, it can take some number of turns. If the bot trying to attack a target is owned by a neutral, the action will be considered as an expansion; however, if it's owned by the enemy, the action will be considered as a conquest. Once the base planet is reinforced by starships coming from colonies, the action will be considered as a Tith. In addition, when colonies that are closed to the target planet than the base is allowed to attack the target by sending fleets, instead of reinforcing the base and this one sends those fleets to the target planet, but move directly to the target. Besides, when a planet was targeted by the bot, it is no possible to decide another attack against the same planet until the first one finished, that is mean, for each fleet sent to perform an attack, it knows the target planet in its data structure.

In order to improve the behaviour of the bot, a set of rules is defined included some parameters classified by weights, probabilities and amounts. the parameters signification is :

- $Tith_{perc}$: starships proportions that the bot can sends.
- $Tith_{prob}$: Tith probability that a colony sends to the base planet.
- w_{NS} : weight of starships number hosted on the planet.

- w_{DIS} : weight of distance between the base and the target planet.
- w_{GR} : weight of growth rate according to the target planet.
- $Pool_{perc}$: percentage of extra starships can send from the base to the target planet.
- $Support_{perc}$: Percentage of extra starships from the colony to the target.
- $Support_{prob}$: Probability of sending extra starships from the colony to the target.

Furthermore, each parameter takes some values during the optimization process depends on it meaning in the game, our conquer bot will be based on these values to make decisions during the game. To determine the target planet the bot will use a score function define as follow:

$$Score(p) = \frac{p.NumStarships.w_{NS-DIS}.Dist(base,p)}{1 + p.GrowthRate.w_{GR}} \quad (2)$$

Where the w_{NS-dis} and the w_{GR} are weights related successively to starships number, the growth rate for the planet and the distance to reach the target. As we mentioned above the *base* is the planet which has the highest number of starships and *p* refer to planet want to evaluate.

When the Tith and the attack process are performed by the colonies, the base planet also sends starships to attack the target. The number of starships sends to attack is estimated according to the attack mode, if the bot attack mode is expansion where the target does not generate starships during the game that implies the bot will integrate a very specific starships number to be able to defeat the planet target, in the other hand, if the bot attack mode is a conquest, the engine will estimate the number of troops needs to beat the enemy.

V. AISBOT: AN AIS OPTIMIZATION

The purpose of our optimization is to make the bot move faster and try to minimize the number of turns before destroying the enemy. In the other hand, a CLONALG algorithm will be applied in the set of parameters that will determine the behaviour of the bot, once the parameters are improved, the bot will upload them in order to go under real game scenarios.

Therefore, the purpose is to determine the parameters values in order to evaluate the bot's behaviour. The AIS proposed in this study use a floating array to represent all parameters defined previously, while, the clonal proliferation process works on a totally new array with a specific proliferation factor for each parameter according to its fitness. The maturation mechanism is applied to the cloned antibodies by mutating the parameters values using the equation (1) with a probability amount to minimize the bad antibodies.

The selection process based on an evaluation technic by implementing tournaments where each cloned individual used to compete with the aim of being selected for the next

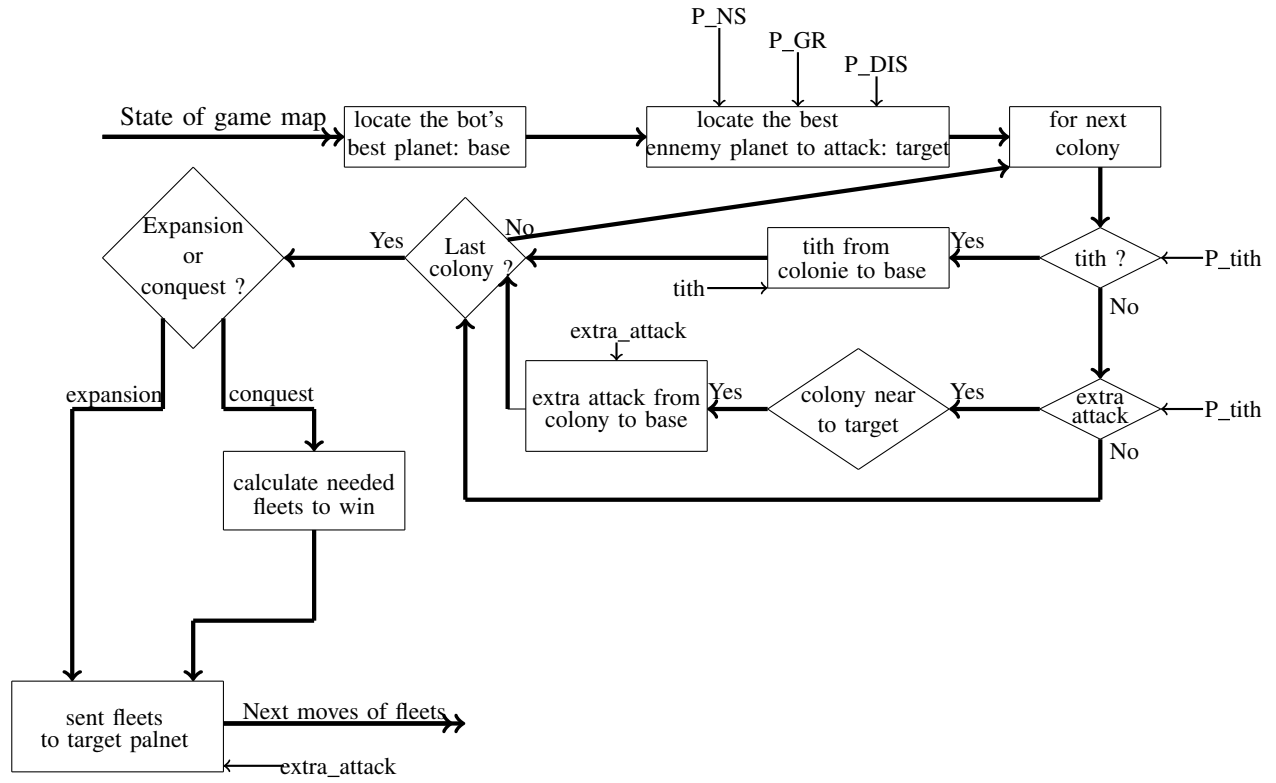


Fig. 4. The states that are governing the behaviour of the AisBot including the parameters used on the evaluation of the bot's behaviour. these parameters will improve by the clonal selection algorithm (CLONALG).

generation from the best results were chosen. The evaluation of an individual is staged by including their parameters for the AisBot behaviour and placing the bot on a real planet war game against the RandomBot on five maps.

The aim of this optimization is to minimize the number of turns needs to win, the best bot is the one that plays fewer turns until win; this kind of ranking bots strongly shown a constraint problem is that each individual was able to win every single game.

VI. EXPREMENTATION AND RESULTS

With the aim of testing the CLONALG algorithm, the bot placed under several situations and play several games against the RandomBot. The algorithm parameters can be found on the Table 1:

Number of antibodies	20
Number of generations	15
maximun antibodies Cloned	72

TABLE I

PARAMETERS USED IN THE CLONALG ALGORITHM

To make a bot under a real game scenario with optimized parameters will take hours, since, to evaluate each individual it took more seconds. The figure (5) showed the evaluation of the number of turns plays by the bot until win, and it appears

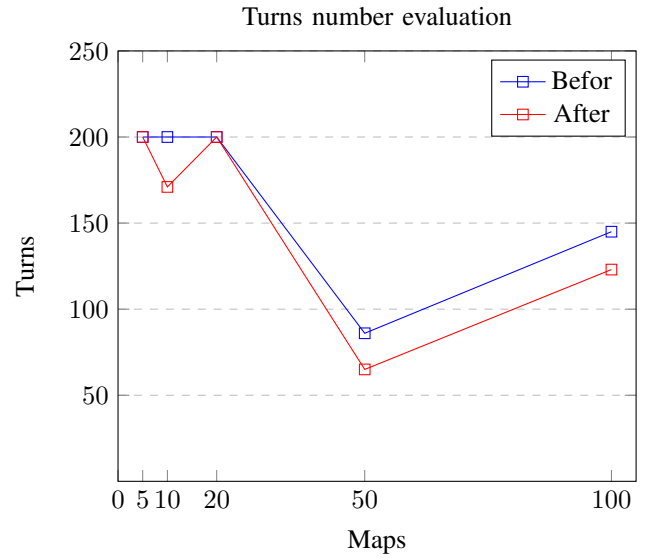


Fig. 5. The evaluation of the turns number needed to win around multiple maps between initial bot and optimized bot

that the number of turns needs to win in five maps decrease.

Figure (6) and figure (7) describes the evaluated results improved by the optimization process where the colonies

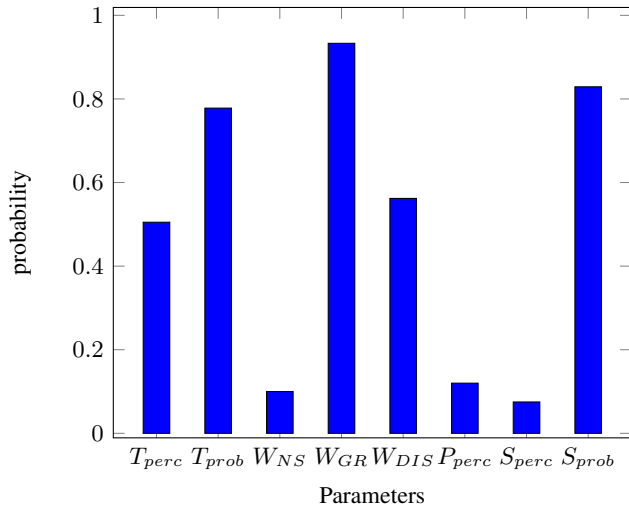


Fig. 6. The parameters values setting for the initial bot

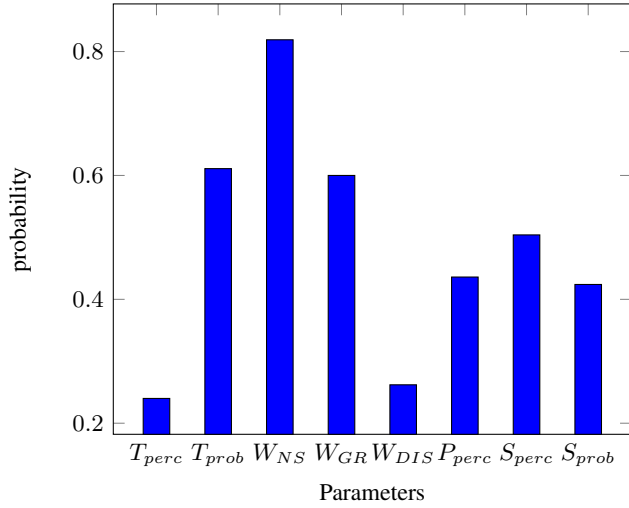


Fig. 7. The parameters values setting for the optimized bot

have a high probability to send Tith to the base (0.61) by sending a fewer hosted starships which can be explained that the colonies have to deserve there starships to defend it-selves in an emergency. On the other hand, the probability for a planet to perform an attack against other ones is around (0.42) and the proportion of sending troops is elevated (50%) when there is an attack action. Based on this, when a planet performs an attack, the base also can send a high extra starships number for the attack around 43%. Finally, when the bot trying to determine the target planet to perform the attack, the most important weights based on is the number of starships hosted on the target and the growth rate related to it, the distance has a less priority in this process.

In general, the optimised AisBot is able to defeat the Dual-Bot in all situations with a minimum number of turns, this can

demonstrate that the application of an Evolutionary Algorithm reach some hight capabilities in optimization behaviour even a parametrize behaviour as the one used on the AisBot. There are a lot of challenges can explore all the capabilities of AIS algorithms in the design of the bot's behaviour can offer.

VII. CONCLUSIONS

The planet wars game it was classified as an RTS game, where the player (bots) fight against others in real-time. The main aim of this study is to investigate the application of the Evolutionary Algorithms (EAs) and the better results can be obtained in real-world problems, by tuning real game scenarios that used a parametrize behaviour of a bot tuning by an AIS Algorithm. This problem showed that the EAs can improve the evaluation of a hand-coded bot by defeating multiple opponents in different situations in fewer turns. In the other hand, the high capabilities of an evolutionary approach can be improved at least in this type of problems.

REFERENCES

- [1] M. N. Collazo, C. Cotta, and A. J. Fernández-Leiva. Virtual player design using self-learning via competitive coevolutionary algorithms. *Natural Computing*, 13(2):131–144, 2014.
- [2] D. Dasgupta, Z. Ji, and F. Gonzalez. Artificial immune system (ais) research in the last five years. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 1, pages 123–130. IEEE, 2003.
- [3] A. Fernández-Ares, P. García-Sánchez, A. M. Mora, P. A. Castillo, J. J. M. Guervós, D. Camacho, M. Gómez-Martín, and P. González-Calero. Designing competitive bots for a real time strategy game using genetic programming. In *CoSECivi*, pages 159–172, 2014.
- [4] A. Fernández-Ares, P. García-Sánchez, A. M. Mora, and J. Merelo. Adaptive bots for real-time strategy games via map characterization. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 417–421. IEEE, 2012.
- [5] A. Fernández-Ares, A. M. Mora, J. J. Merelo, P. García-Sánchez, and C. Fernandes. Optimizing player behavior in a real-time strategy game using evolutionary algorithms. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2017–2024. IEEE, 2011.
- [6] A. R. Jordehi. A chaotic artificial immune system optimisation algorithm for solving global continuous optimisation problems. *Neural Computing and Applications*, 26(4):827–833, 2015.
- [7] R. Lara-Cabrera, C. Cotta, and A. J. Fernández-Leiva. A review of computational intelligence in rts games. In *Foundations of Computational Intelligence (FOCI), 2013 IEEE Symposium on*, pages 114–121. IEEE, 2013.