2018 International Conference on Applied Smart Systems (ICASS'2018)
24-25 November 2018, Medea, ALGERIA

Optimizing Planet Wars bot using an immune-based algorithm

Mohamed Amine Chikh Touami
Faculty of Exact Sciences
University Mustapha Stembouli
Mascara, Algeria
ma_chikhtouhami@hotmail.com

SALEM Mohammed
Faculty of Exact Sciences
University Mustapha Stembouli
Mascara, Algeria
salemohammed@gmail.com

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

***Abstract*—This work presents an Artificial immune system algorithm (AIS) for a real-time strategy game: Planet Wars. This game requires the bot to reach some adaptabilities with the aim to beat different enemies in different maps. This Algorithm uses an evolutionary strategy to evolve a set of parameters for an autonomous agent (bot) to achieve some goals in the game. In order to optimize the behavior of the bot, the AIS was applied to tune the set of rules. We have run several experiments in different situations obtaining encouraging results.**

***Index Terms*—Computational intelligence, real-time strategy games, planet wars, artificial immune system.**

I. Introduction

The large entertainment industry sector is the video games. The commercial video games budget is rising every year, it reaches 25.1 billion dollars in 2010[8]. Many designers and developers are interested in the community of video games around the world. The main objective of these video games community is to provide amusement to their players. The question is how to do this; there has been no detailed investigation of the full mechanism to put the work in the correct way[1]. In the past, the developers overlooked their games artificial intelligence and focusing their works on high-quality graphics games. Moreover, the human players are able to defeat these games without much effort and lose the interest[1].

In light of recent events in video games, it is becoming extremely difficult to ignore the existence of one paradigm that is fast becoming a key instrument in most video games, which is the Artificial Intelligence(AI). The AI is an important aspect of the conception of this kind of games. Moreover, it has seen the rapid of development in many fields such as human player imitation, procedural content generation(PCG), automating game testing, opponent modelling, and computational narrative, among others[1].

Recently, researchers have shown an increased interest in developing AI for RTS games. RTS games are a sub-genre of strategy video games, which all the actions made in real-time. The contenders have the capability to control (make a decision) a set of distributed units and structures during the game with the aim of (a) destroy the opponent assets (b)create additional structures to reached some goals in the game or to secure area[1], [5], [8], [3], [4]. At the most of RTS games they employ two levels of AI[5], while can classify these two levels by (a)strategic when it will make a decision over the whole set of units (b) tactical when they decide the behavior of each these small units. In addition, with the real-time aspect challenge, these difficulties are increased. PlanetWar is a real-time strategy game, where contenders presented by autonomous agents play against each other on multiple maps. with the aim of winning the game, the players will generate multiple attacks including a number of starships from their planets against the other planets and use several strategies to defeat the enemy.

One of the hardest tasks in PlanetWar games is the bot's conception. For simulating the other contenders, designers use the bot concept. Bots are intelligent agents interact with human players in order to compete them within any computer-based framework[4]. This challenge is made by experts human to design bot's behavior from their life experiences and experimentation in order to increase the game's challenges[3]. The intelligence bots have a pivotal role[4] in the conception of an RTS games. In addition, for a PlanetWar agent(bot), a set of parameters are required previous the running of the bot, moreover, a good parameter's values are making a better behavior from the bot. So, tuning the parameters previously cited, is wide research area like optimization problems[5]. This technic is used to determine which parameters are more important for the bot to tune[5].

In the other side, around our bot's optimization, we use an artificial immune system algorithm (AIS) to improve the set of parameter. AIS is an evolutionary algorithm which inspires its technics from the biological immune system that used to defend the body against pathogens. In this paper, we aim to design an autonomous bot for the planet war game based on AIS algorithm to improve the set of parameters trying to maximize an objective function formed by the turns played by the bot and its number of wins.

This paper has been divided into four parts: the first part deals with the description of the planet war game and its rules. The next section is dedicated to present the AIS algorithm.

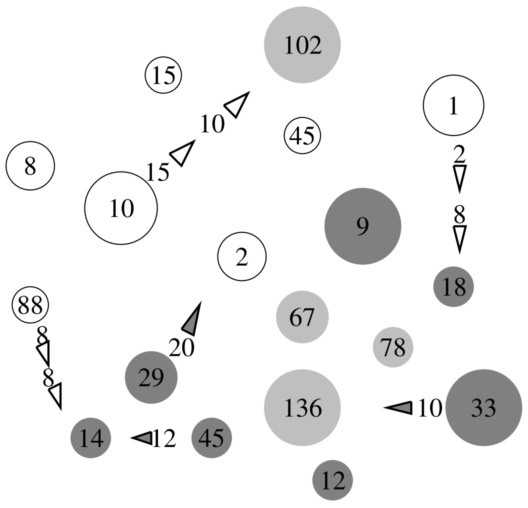

Fig. 1. an early stage of a planet wars game, as shown here there are planet categories, the darker grey (red in the game) ones related to the player, the white ones (green in the game) related to the opponent and the neutral planets are coloured by light grey. Fleets presented by triangles and the numbers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

The third section tries to define the proposed approach and the architecture used to develop the bot based on an AI behavior. The remaining part of the paper presents the experimental results.

II. Planet War

In 2010 google choose like a game planet wars for it AI challenge competition. In this pager, we work with a version of Glacon game in order to design a fighter bots. A planet war match[5] runs on a map with several distributed planets in different positions, such as each of them has a number describes the quantity of starships its owns. In each turn, the number of starship will change. Each planet is owned by a player, opponent or neutral planet(owned by nobody), moreover, each planet belongs to the player (no neutral one) will increase their number of starships after each step according to their growth rate, this rate explain the additional starship's number will generate each planet in each step. The main purpose of the planet war game[5] is to challenge all players and try to defeat all of them, that mean to destroy all enemy's starships. Planet war is an RTS game by its nature. To achieve a specific objective each player has a limit number of turns to do it and try to win (beat opponent).

Moreover, the bot's design faces two major problems: the first one [5] is that the bot can't store any information about its previous turns like actions it did, actions of its opponent or even the map's state. Since in each turn, the bot meets with an unknown map as it just begins the game for the first time. The second one[3] is that the time requires to move(make an action) is just one second. So these difficulties make the bot's design like a great challenge. In this kind of game, everything has a sum of properties[5]. Such as, planets have the x and y coordination for their locations in the map, ownersID, number of starship and growth rate. To conquest other planets, the player needs to send fleets for this aim. Each fleet has a playerID parameter, number of starships, source planetID, destination planetID, turn's number need to reach the target. In each simulated turn[5], the player has the ability to send fleets in a battle with other planet's starships with the source planetID and target planetID, while, the player has the ability[5] to make just one action per turn. After each move (action) the planets belong to the players[1] increase their ship's number according to their growth rate. To make fleets previously sent reach enemies planets its take time of many turns according to the distance between the source planet and the target planet, moreover, if the source and target planet owned by the same player then it considered like a reinforcment[5], [1] by adding the ship's number on the fleet to the number hosted on the target planet. when the fleets reach the target its will make battles with local target ships, otherwise, if the target is a neutral planet, the player must to sends at least one additional ship to own the planet. While, if the target is owned by the enemy the two players will fight until one with the highest starship number owns the planet. When the player sent fleets to reach some goals[1], it can't change their directions. In addition, the player can send more fleet in the next turns even that the first ones do not yet arrive at their destinations. In the end of the match[1], the player with the highest number of starships wins. If a player loses all their starships then the game will end also faster. whilst, the players have the same starship's number at the end of the game there is a draw[1].

III. Immune System Overview

One of the most inspirational things that computational based to solve their problems[2] is the biology and its various concepts. Among the natural concepts, the immune system has emerged as one of the important aspects to design optimisation algorithms. An optimization problem[6] is to look for the optimum element from a set of potential solutions. There are many optimization problems in all of the scientific areas. For this purpose, many methods and algorithms have been developed. The most useful algorithms[6] are heuristics. They include divers algorithms such as artificial bee colony, firefly swarm optimisation, ant colony optimization, genetic algorithms, just to mention a few. While the Artificial immune system (AIS) is a common algorithm devised by Decastro[6] that uses population-based heuristic concept. The mechanisms used in the biological immune system against various types of pathogens made this system more interesting[7] for computer researchers to exploit its technics. This kind of algorithms is used to solve many different optimization problems in science and technology areas. However, the immune system is more complex and its application in technologies areas is a challenge, likewise, the biological immune system[2] still under wide active research, whilst, the AIS is adopted just for few mechanisms inspired from the human immune system including the clonal selection.

A. natural immune system

The biological immune system including human system contains cells, molecules, and organs in its structure to defend the body against the diseases. In the aim to make an immune response[6], the system has to distinguish between the body cells as a self-cells and the foreign cells as nonself-cells(antigens). The immune system response is activated by a suitable mechanism[6] to defeat the nonself-cells, this mechanism is defined by the type of the antigens, specific antigen implies a specific response and a specific process. In order to make a fast response in case of the similar antigen, the immune system develops memory cells for this purpose.

Clonal selection: One of the important mechanism used to defend against the nonself-cells[6] is the clonal selection. This immune response describes the process how the immune system will stimulus against a specific antigen[2] by proliferating a specific type of cells that only those can recognize the antigen.
When the antigen is inside the body, the immune response[6] by giving the B-cells the ability to secrete antibodies. After that, the T-cells give a signal to the B-cells to proliferate and mature to terminal antibodies secreting cells, it's the plasma cells. The proliferation of the B-cells is according to the affinity level, higher affinity implies more clone will be generated, this whole process named affinity maturation.

The clonal selection process will pass with various steps[2]:

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.

B. Artificial Immune System

Inspiring from the biological immune system an algorithm has been developed called Artificial Immune System (AIS)[6]. The search technic is similar to the natural immune system by the implication between the fitness function and the affinity maturation in the natural system.

1) Initialisation: A random population N is generated in the search space, as a process that used in the other heuristic algorithms. This population considered like antibodies.

2) Clonal proliferation: In this step, the antibodies will clone (proliferate) according to their fitness(affinity).

3) Maturation: The maturation technic is similar to the mutation one with a mutation probability P. This mutation is applied to equation (1) :

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

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.

4) Evaluation: In the evaluation step, the fitness function of every antibody is calculated by computing the affinity values.

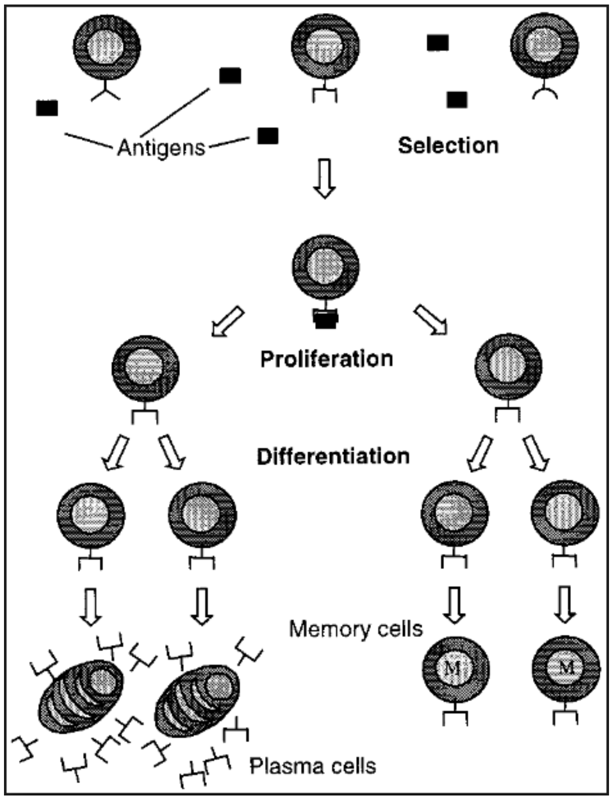

Fig. 2. The biological clonal selection mechanism and its steps in order to defend the body, starting by detecting of the antigen until removing it.

5) Ageing operator: This factor is used to eliminate the individuals lost more. Indeed, the Ageing operator leads to upgrading the initial population.

6) Selection process: This section will be applied to select N individual to the next generation.

IV. AisBot: the Canquer Bot

In the previous sections, we explained that there are two primary constraints in the development of the bot's behaviours. The main constraint is the limited time available for the bot to move (1 second). The second fundamental constraint that is no information could be stored from the previous turns. Based on these two major restrictions, the design performance and the implementation possibilities are limited. However, some researchers have been carried out on the memory solutions based on metaheuristic algorithms to make the bot store some information from its previous actions to determine the next move, but most of them are limited by the time factor; the performance of an EA is limited by time, is almost impossible to run an EA in each step of time (1 second). For the bot's behaviour evaluation it is no possible to improve an action without any feedback from one turn to the next.
The purpose of this investigation is to obtain data which will help to address those research restrictions, while these data were collected using a set of rules in order to optimize the bot's behaviour.

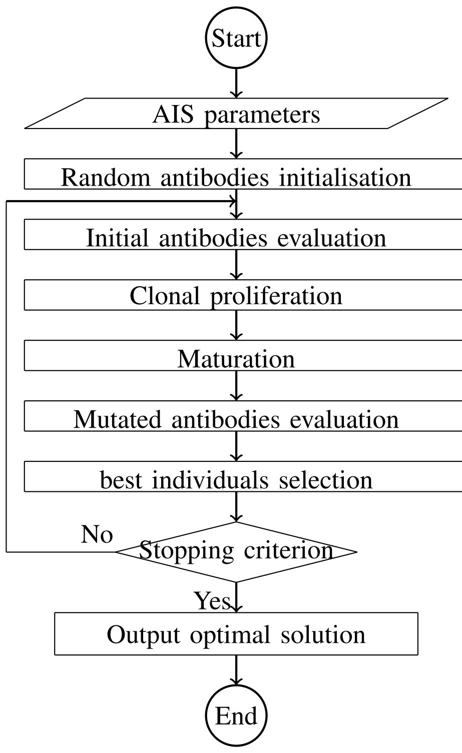
Start
AIS parameters
Random antibodies initialisation
Initial antibodies evaluation
Clonal proliferation
Maturation
Mutated antibodies evaluation
best individuals selection
No
Stopping criterion
Yes
Output optimal solution
End

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

Anyway, in this case of study, we have to perform a single action mode: send starships from a planet to another; to make fleets move in the map, the bot should distinguish between self-planets and the enemy's planets. Based on this simple action, one of the greatest challenges is to determine which planet that will generate the fleets, how many starships will include on it and what is the planet will be targeted. In the next, we will describe our bot's behaviour, following the definition of the parameters used in the optimization. Finally, we will explain the technics governing the bot.

A. AisBot

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:

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

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.

B. RandomBot

The development kit of google for the planet war game include an example of a bot called RandomBot, the behaviour of this bot is very simple and it works independently of any

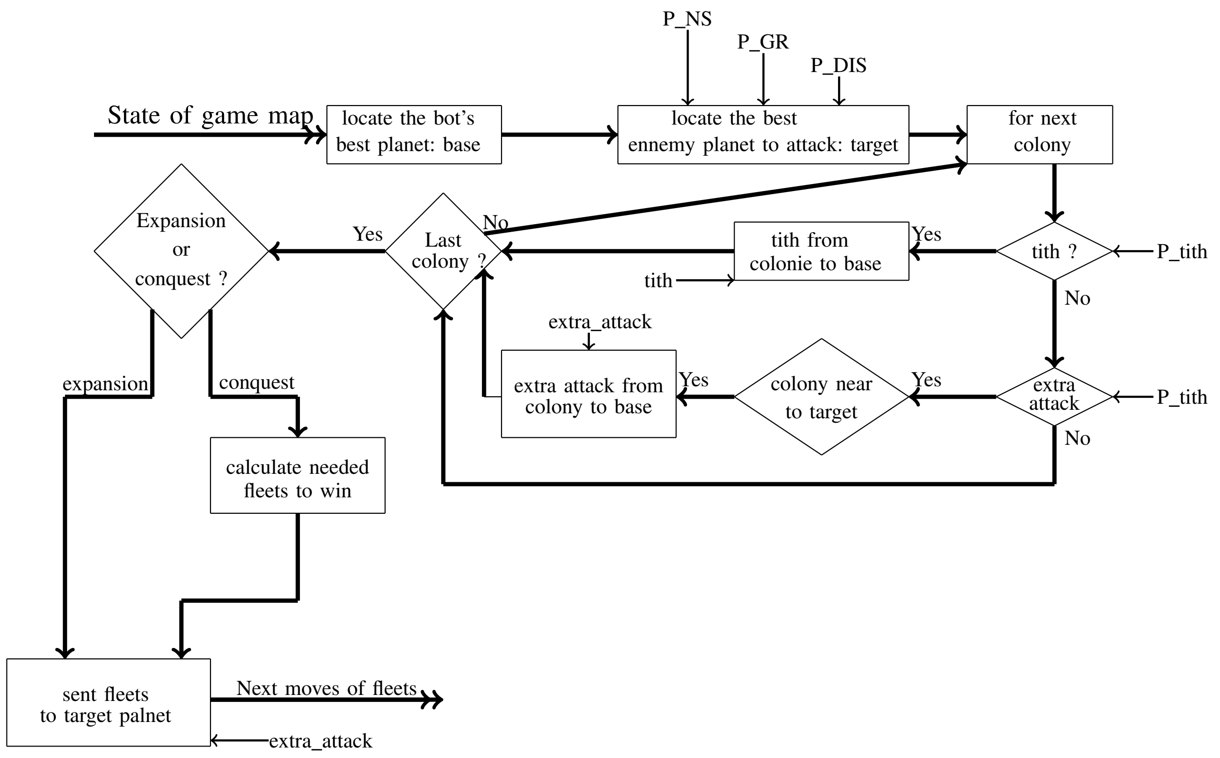

Fig. 4. The states that are governing the behaviour of the AisBot including the parameters used on the evaluation of the bot's behaviour. These parameters will improve by the clonal selection algorithm (CLONALG).

situation (map). Based on it basic behaviour, the bot tries to send fleets randomly and try to perform random attacks without any estimation. If we compare the AisBot and the RandomBot behaviours, we will clearly find that the AisBot's behaviour is more complex than the one on RandomBot, and it be able to defeat it in all situation and maps; however, it needs more turns to beat it; for this purpose, the greatest challenge is to make the AisBot's behaviour quite fast by determining the values of parameters shown in the previous section systematically.

V. AisBot: an AIS optimization

The aim of this optimization is to make the bot behave faster and be able to win in less number of turns, in order to do that, an artificial immune system (AIS) algorithm used on the set of rules included the parameters that will determine the behaviour of the bot; when the parameters are improved, they are fixed and will be uploaded by the bot in order to play a real scenario. 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 generation from the best results were chosen. The evaluation of an individual is staged by including their parameters for the AisBot's 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 antbodies 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 that the number of turns needs to win in five maps decrease when the evaluation progress. Furthermore, in figure (6) two, the parameters evaluation is shown during the optimization process in amoung fifteen generations.

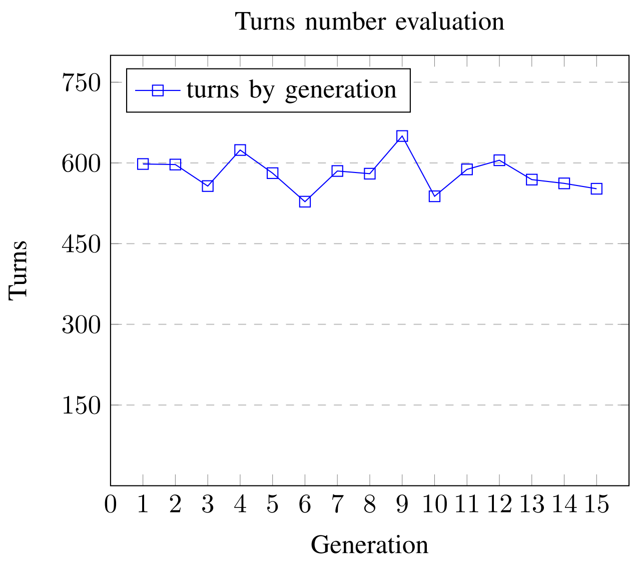

Fig. 5. the evaluation of the number of turns plays by the bot until win in 5 maps, and it appears that the number of turns needs to win in five maps decrease when the evaluation progress

Table II describes the evaluated results improved by the optimization process where the colonies have a high probability to send Tith to the base (0.84) by sending a fewer hosted starships which can be explained that the colonies

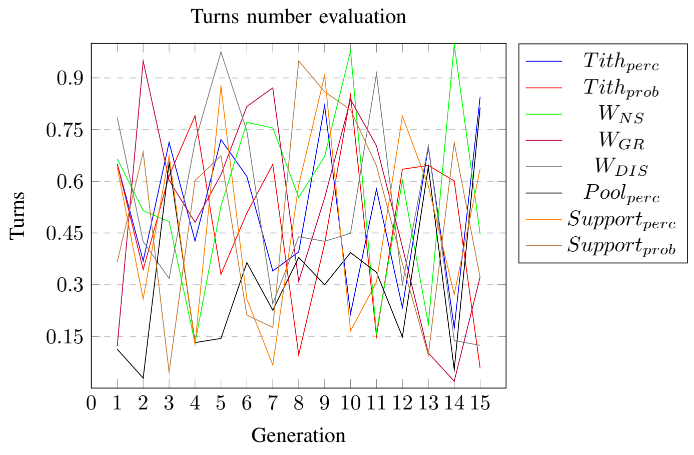

Fig. 6. the evaluation of the Parameters during the optimization process

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 (0.32) and the proportion of sending troops is elevated (64%) 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 81%. 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.

	Before	After
$Tith_perc$	0.959	0.057
$Tith_prob$	0.651	0.845
W_NS	0.665	0.447
W_GR	0.122	0.323
W_DIS	0.758	0.124
$Pool_perc$	0.113	0.814
$Support_perc$	0.64	0.636
$Support_perc$	0.366	0.32

TABLE II
PARAMETERS USED IN THE CLONALG ALGORITHM

Maps	Before	After
5	135	98
10	128	80
20	161	70
50	102	72
100	144	87

TABLE III
RESULTS AFTER 5 GAMES OF OUR AISBOT WITH RANDOM PARAMETERS AND AFTER THE OPTIMIZATION AGAINST THE RANDOMBOT

In general, the optimised AisBot is able to defeat the RandomBot 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 was classified as an RTS game, where the player (bots) fight against others in real-time. The main aim of this study was to investigate the application of the Evolutionary Algorithms (EAs) and the better results can be obtained in real-world problems, by tunning real game scenarios that used a parametrise 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 417–721. 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] J. Kim and P. J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 1330–1337. Morgan Kaufmann Publishers Inc., 2001.

[8] 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.