ARTICLE TEMPLATE

A comparison of two computational methods in order to improve the behavior of a Planet Wars game agent

A. N. Author[a] and John Smith[b]

[a]Taylor & Francis, 4 Park Square, Milton Park, Abingdon, UK; [b]Institut für Informatik, Albert-Ludwigs-Universität, Freiburg, Germany

ARTICLE HISTORY
Compiled March 28, 2019

ABSTRACT
Inherited by the real-time aspect, one of the hardest tasks in RTS games is the designing of the agents(Bots). In this work, we present a comparison of three computational methods that will be applied to improve the behavior of the bot during the game. The agent will react under a game called planet wars, which in this game, the bot need a set of parameters that will determine the actions during the playing match. In the other hand, the two approaches presented in this study will try to tune the set of parameters the Bot will be based on in the aim of improving the general behavior.

KEYWORDS
Computational intelligence, real-time strategy games, planet wars, artificial immune system, particle swarm optimization.

1. Introduction

Nowadays, there has been an increasing interest in the field of video games. The influence of these video games in our life and culture led the researchers to try to design more realistic games for more entertainment. In the other hand, there is more than one kind of video games, such as chess, go, non-player character (NPC) and first person shot (FPS) to mention a few.

In this study, we interested in a different subgenre of video games called strategy video games. in this kind of games you have to control a set of units distributed on a multidimensional space when contenders have to move in the aim of destroying all the enemies structures in the minimum number of turns or to protect the self-unit till the end without losing the game. those games, by their nature, are real-time games, that mean all players have to act simultaneously without waiting other to move[3] like chess or go.

Existing research in real-time strategy games (RTS) recognizes the critical role played by computational intelligence in this major. Moreover, inherited by their nature, RTS games are a good tool for testing the different methods and technics of computational by creating new challenges to improve the designing process of this

CONTACT A. N. Author. Email: latex.helpdesk@tandf.co.uk

kind of games. In the other side, the contribution of Computational in that genre of games lies in the conception of the agent that will play the game and interact with the human side.[1,3–5,7]

this research focuses on one of the RTS games called Planet Wars. In this game, two players will be faced on multiple types of maps in the aim of overwhelmed the opponent or to protect the self-units without losing the game. While Each player can generate multiple attacks against the other contenders including a specific number of troops in several strategies to bit the enemy. Moreover, the players on our case are autonomous agents that will react in a real scenario under this game, those agents named Bot. As we mentioned previously, one of the most challenges in RTS games is the Bot conception. This intelligent agent will act under a computer-based framework[3] against a human player in order to compete it.

The Bot conception is a major area of interest[4] within the field of computational intelligence in order to design an RTS game. Which, the problem is the real-time aspect related to this type of games. In addition, to control a Planet Wars agent, a set of parameters previously improved will determine the reaction of the Bot under the game. Among all Planet Wars parameters, we have to choose a set of the highly effective ones[1] that will be used by the Bot for best behavior during the game.

This process already an optimization issue[1], whilst, our work is to improve some parameters predefined by experts in the domain. So, the optimization process will move in three fields: the first step is going by using an algorithm inspired by the immune system of the human body called an Artificial Immune System (AIS). The second step is to run another nature-inspired algorithm named Particle Swarm Optimization (PSO), this algorithm behaves in the same way with nature animals swarms as bird swarms, fish schooling ...etc. The last step is to make a comparison between those two optimization methods and the one used in [1] that implemented the Genetic Algorithm (GA).

This paper begins by describing Planet Wars, the game used in this work and all the rule related to. It will then go on to explain the AIS and POS algorithms and how it works. The third part is about the general based architecture of the bot to move intelligently during the game. The last section tries to define the comparable results between the optimization methods used.

2. Planet War

As we mentioned previously one of the international competition was organized by google community. In this pager, 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 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

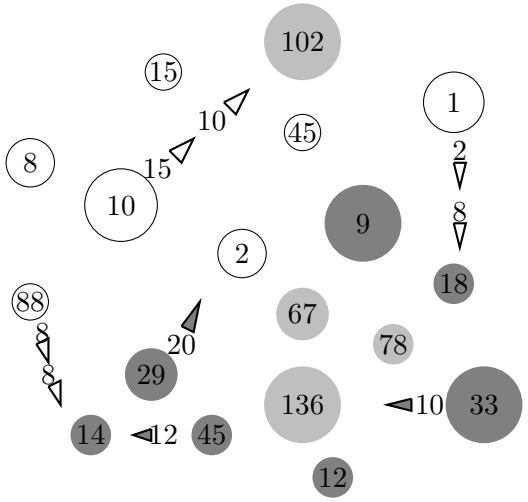

Figure 1. an early stage of a planet wars game, as shown here there are planet categories, the darker grey (red in the game) ones related to the player, the white ones (green in the game) related to the opponent and the neutral planets are coloured by light grey. Fleets presented by triangles and the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

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 bot 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] in 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.[1,5]

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].

3. 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.

3.1. 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:

3.1.0.1. . The cloned cells undergo to a mutation process.

3.1.0.2. . the self-reactive receptor will be eliminated.

3.1.0.3. . proliferate the mature cells those can detect the antigen.

3.2. Artificial Immune System

In light of recent event in AIS, many algorithms have been developed trying to simulate ti 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

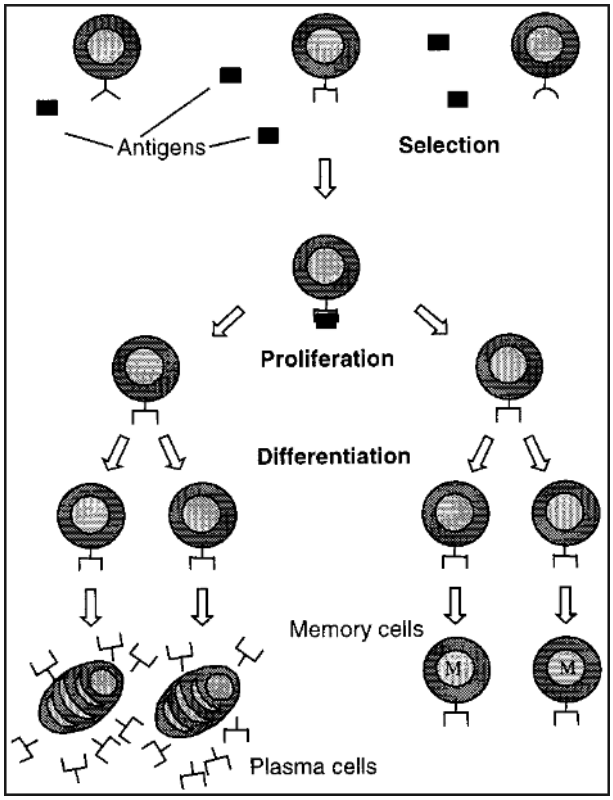

Figure 2. The biological clonal selection mechanism and its steps in order to defend the body, starting by detecting of the antigen until removing it.

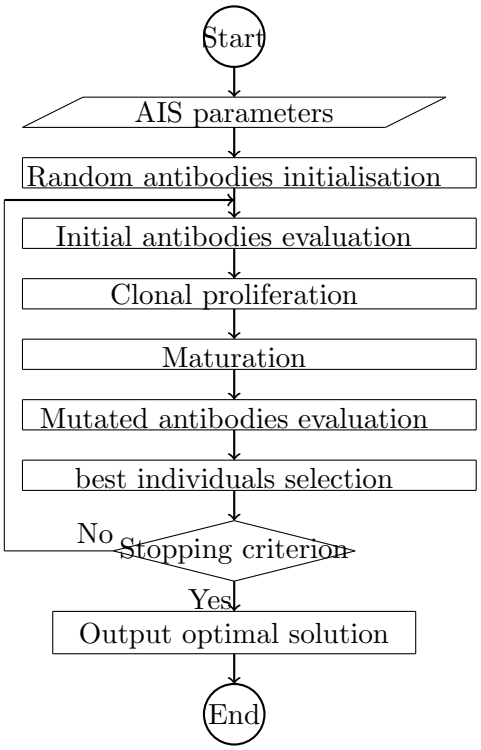

Figure 3. The diagram showed the ordered steps for running an AIS algorithm based on clonal selection algorithm (CLONALG)

proliferation of the antibodies is according to their affinities. While the maturation applied to equation (1):

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

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.

3.3. 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.
- Compute the affinities for the new antibodies mutated.
- Applied the selection process to select the next generation

End For
Display the optimal solutions.
End

3.4. Particle Swarm Optimization

Recently, a considerable literature has grown up around the theme of swarm intelligence which plays an important role in addressing the issue of computational systems inspired its techniques from nature collective intelligence such us fish schooling, ants, flocks of birds to mention a few. The algorithm run in a multi-dimensional space where all particles try to locate the optimum. By first, a set of a collection will be set to represent whole particles in the swarm randomly positioned in the working space, Each particle can move to find the potential optimum pathway according to the personal position of the particle (Pbest) and the global position (Gbest) of all the swarm. In addition, another factor will influent this changing in the position, it's velocity. The change in velocity is measured with the equation (2):

$$v_i(t+1) = v_i(t) + c_1 r_1(p_i^{Pbest} - p_i(t)) + c_2 r_2(p_{Gbest} - p_i(t)) \tag{2}$$

Where $v_i(t+1)$ is the next velocity of the particle i, c_1 and c_2 represent the weight of Cognitive component and the Social component for the personal best and global best successively. $p_i(t)$ is the position i of the particle p at time t. p_i^{Pbest} is the best known position for the i^{th} particle. p_{Gbest} is the best known position for the whole swarm. r_1 and r_2 are random variables in the range [0, 1]. In the other side, the position for each particle will be changed according to equation (3):

$$p_i(t+1) = p_i(t) + v_i(t) \tag{3}$$

Where $p_i(t+1)$ represent the next position of the particle i.$p_i(t)$ is the current position of the particle i at time t and $v_i(t)$ is the velocity factor.

4. 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.

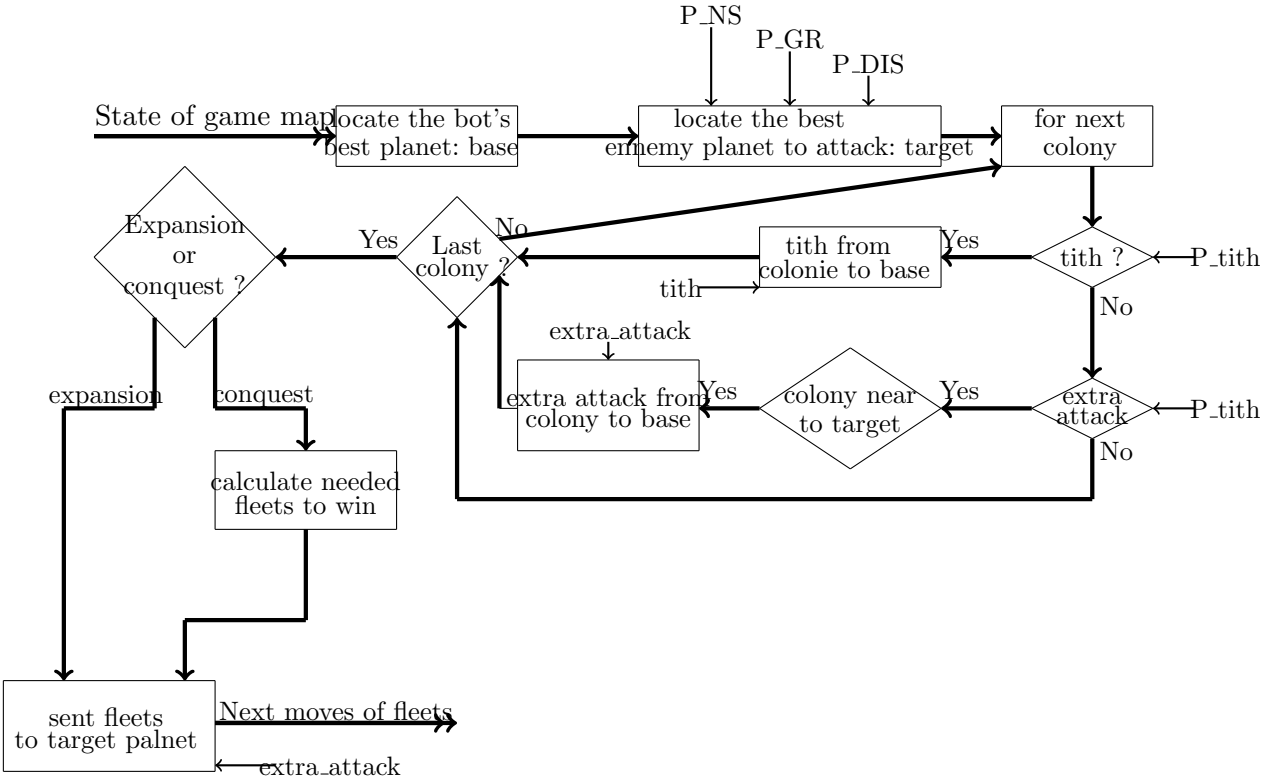

Figure 4. The states that are governing the behaviour of the AisBot including the parameters used on the evaluation of the bot's behaviour. these parameters will improve by the clonal selection algorithm (CLONALG).

4.1. 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,

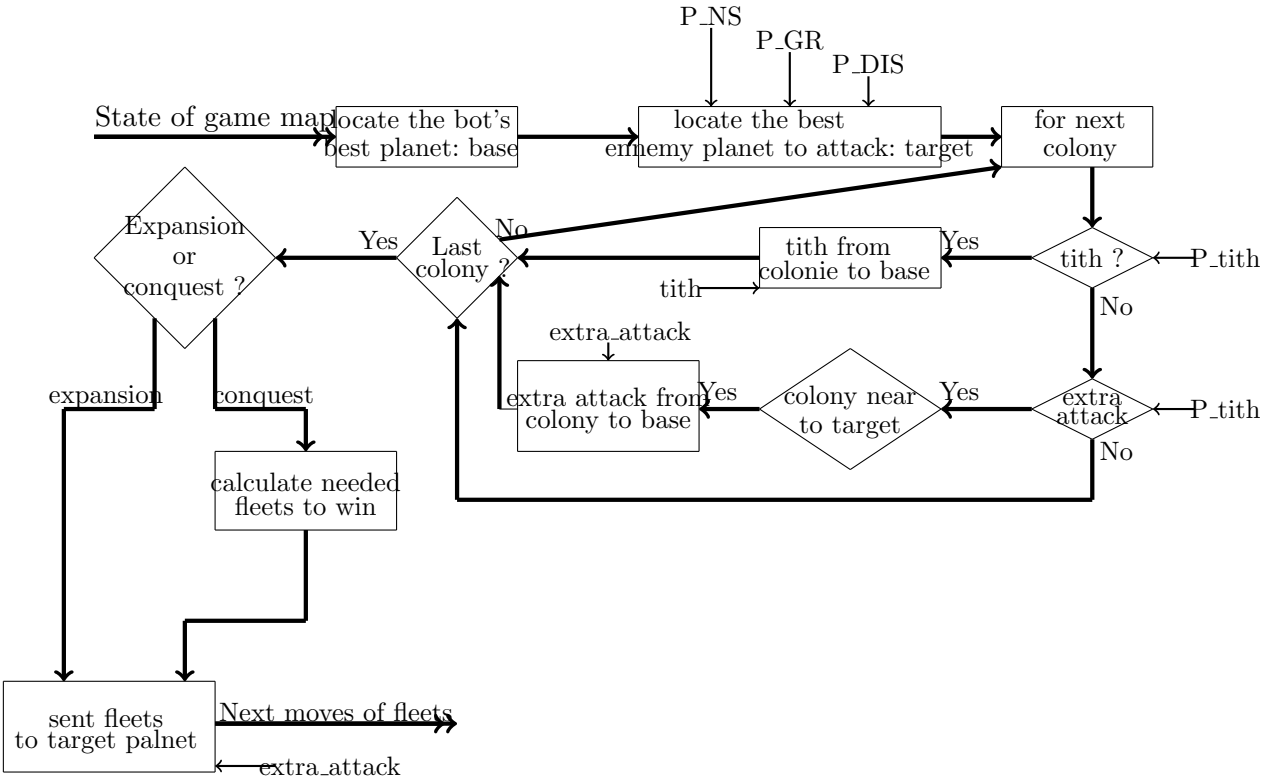

Figure 5. The states that are governing the behaviour of the AisBot including the parameters used on the evaluation of the bot's behaviour. these parameters will improve by the clonal selection algorithm (CLONALG).

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:

$$Score(p) = \frac{p.NumStarships.w_{NS-DIS}.Dist(base, p)}{1 + p.GrowthRate.w_{GR}} \tag{4}$$

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.

5. 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 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.

6. 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 1. 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.

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

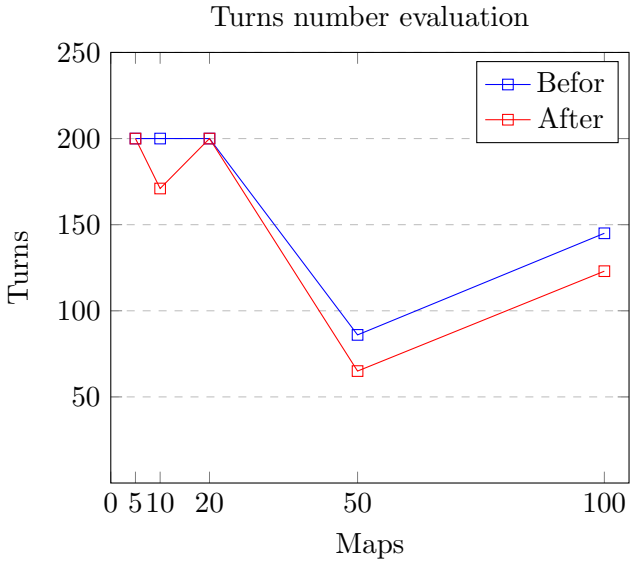

Figure 6. The evaluation of the turns number needed to win around multiple maps between initial bot and optimized bot

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 DualBot 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.

7. 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 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,

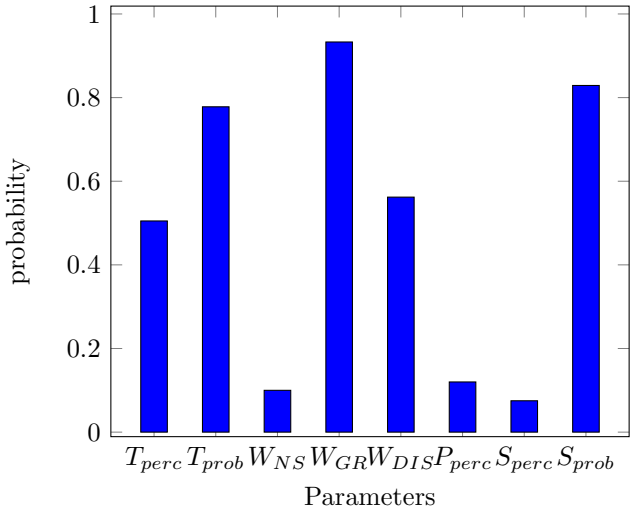

Figure 7. The parameters values setting for the inital bot

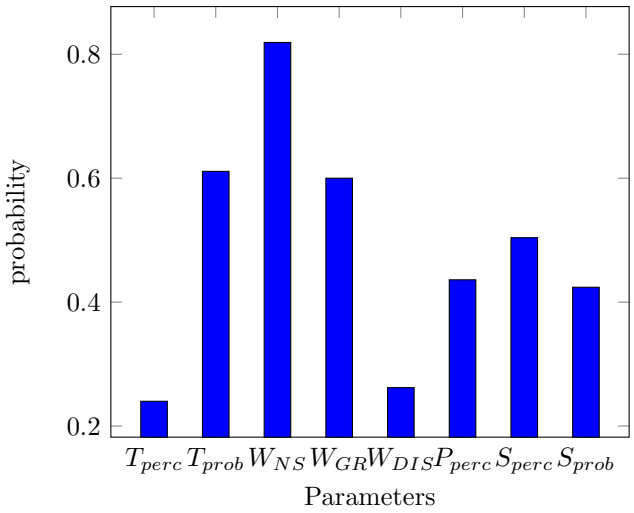

Figure 8. The parameters values setting for the optimized bot

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] 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.