Evolving Combat Algorithms to Control Space Ships in a 2D Space Simulation Game with Co-evolution using Genetic Programming and Decision Trees

Tiago Francisco
School of Technology and Management
Polytechnic Institute of Leiria, Portugal
tiago3f@gmail.com

Gustavo Miguel Jorge dos Reis
School of Technology and Management
Polytechnic Institute of Leiria, Portugal
gustavo.reis@estg.ipleiria.pt

ABSTRACT

Developing artificial behaviours to control artificial creatures or vehicles is a task that can be employed by means of Evolutionary Algorithms. A game's artificial intelligence is usually developed by seasoned game developers, which need critical knowledge of the games mechanics and rules. This paper presents an alternative, using evolutionary computation to evolve combat algorithms that will allow spaceships to fight effectively in a 2D space simulation game. These combat algorithms will take into account the spaceships characteristics, using them to gain the advantage needed to fight effectively

Categories and Subject Descriptors

I.2.4 [**Artificial Intelligence**]: General; D.2.m [**Software Engineering**]: Miscellaneous

General Terms

Algorithms

1. INTRODUCTION

Game developers are normally challenged with the creation of Artificial Intelligence (AI). This normally takes time and deep knowledge of the game's rules and mechanisms. Also hard coding the algorithm presents various problems: if by some reason the game mechanics change or the rules suffer a slight alteration, the algorithm may need to be updated and more time is spent.[1, 2]

Evolutionary Neural Networks can also be used to obtain the human-like A.I [1] greatly reducing the need to have dedicated game A.I developers. Also if by some reason the mechanics/rules of the game change it is easy to adjust the neural network to work with the new rules.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

Evolutionary Computation can also be used in order to tune the parameters of a finite state machine. Nicholas Cole et al.[2] created a solution that used a genetic algorithm to tune the parameters of a Counter Strike[1] bot, creating a human-competitive result. Those results showed that a bot tuned with the G.A solution was better than the human tuned bots. David B. Fogel et al. [3] developed a platform that, using co-evolution and genetic algorithms, evolved simulated combat algorithms for tanks and robots. These tanks and robots had various traits and cognitive characteristics that led to very interesting results.

This paper aims to present an alternative approach: using genetic programming to create combat strategies, that will allow agents to fight each other in a 2d space simulation, taking into account the agents spaceship characteristics, using them to gain advantage

1.1 Bellerophon

The basis for this paper is a game we developed. Bellerophon is a space simulation game in which the player as freedom to do pretty much anything he wants. Normally computer games have various NPCs[2], each with its own personality. In the case of Bellerophon, 3 major NPCs were created: Military, Pirate and Merchant.

Pirates The outlaws of the game. They pillage, steal and destroy anything they want. Normally they leave powerful ships alone, since their own ships are fragile. Also the pirates are normally cowards, in the sense that they tend to flee if they ship is about to be destroyed.

Merchants The backbone of the game's economy. They roam from system to system, buying and selling they products to make money. Their are the normal targets of pirates.

Military The "good guys". They ensure the law is obeyed by everyone, and will attack pirates on sight. They tend to have a powerful ship to dissuade any attempts of attack. Also they never surrender since they have training, meaning they fight to the death.

[1]Counter Strike is a First Person Shooter game.
[2]Non Player Characters

Most of the NPCs behaviours were hard coded, with the help of Finite State Machines. For instance: the patrolling by the military and the call for help by merchants. The Pirate, for instance, patrols a sector of the universe, always keeping an eye out for potential targets or dropped cargo. If it sees a merchant, or the player, it may attack, threaten or leave them alone.

Since combat is an important part of Bellerophon, we wanted each NPC's combat tactic to reflect its personality. For instance, we wanted the pirate to employ hit and run tactics, using its ships speed and agility as an advantage while dealing the most damage possible.

1.2 Vehicles

All the inhabitants of the universe control a spaceship that can move around the universe.

Ships have a given number of hit points which represent the structural integrity of the spaceship, and shield points which represent an energy shield that surrounds the spaceship. When a spaceship is hit with a projectile, the shield energy is decremented. When the shield energy reaches 0, the hit points are decremented with each hit. When the hitpoints reaches 0, the ship is destroyed.

Ships also have maximum speed, acceleration, turning speed, and a weapon. The weapon fires, at an interval, a projectile that travels at a given speed. Ships also have a radar, which gives its controller a view of its surroundings. The radar also serves another purpose; it functions like an array of sensors, allowing the ship's controller to know where its target is.

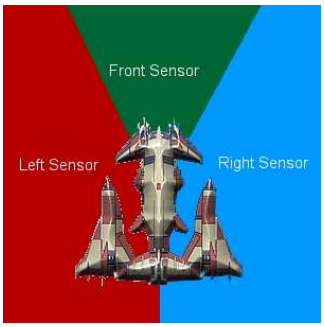

Figure 1: Example of sensor layout

This sensor layout represents what the ship's controller can see at any given time

2. PROBLEM AND APPROACHES

There are two species: Military and Pirate, each with different ships and consequently different play styles. The military has a much more powerful ship than the pirate, has more hit points and more shield power providing the basis for a more relaxed type of combat, in which the military rarely flees its attacker. The military also makes use of its superior ship's hp, soaking up some damage so that it has more time to shoot at it's target.

The pirate on the other hand has a much more fragile ship, but its ship is much faster and more agile. This provided the basis for a hit-and-run combat tactic, where the pirate made use of its faster ship to its advantage. Both ships have the same controls, meaning the only real difference between the two species is the space ship characteristics. One is fragile but fast, the other slow and powerful.

In order to evolve the combat algorithms of both species, two approaches were addressed: classic co-evolution and Random Opponent Evolution. The goal was to evolve a behaviour - decision tree - that enabled the individual to fight its opponent using various sensors and two basic behaviours that were previously evolved: attack and flee. The sensors are no more than function sets like if-enemy-ahead, if-enemy-weapon-range, etc. The evolution process evolved two distinct species that achieved the main goal: deal more damage than it receives.

2.1 Evolution Process

We used two evolution processes to obtain results: Classical Co-Evolution and another one we called Random Opponent Evolution. We then used both results to compare the evolution processes while comparing the same results with human-coded alternatives.

Random Opponent Evolution was used in order to find a faster alternative to the classic co-evolution: the species still evolve together, however instead of each individual fighting all the individuals of the opposite species, each individual fights one random opponent per generation. This produce much faster results, but not as generic.

Each generation from both evolution process were divided into 35 second "rounds", much like the rounds in a box match. The classic co-evolution had X number of rounds, in which X is the number of individuals in each species. In the random opponent evolution, each generation had one round. All the individuals fought in that round against a random opponent.

3. EXPERIMENTS AND RESULTS

3.1 Environment

The environment is a "system" of the game's universe. It is 5000 by 5000 pixels and can have an unlimited number of spaceships. For the evolution process the configuration presented in Table 1 was used.

3.2 Fitness Functions

The fitness function of the both species is done by the Equation 1, where $DmgD$ stands for Damage Dealt, $DmgR$ damage received, and Pen for potential penalties.

$$Fitness = Constant - (DmgD - DmgR + Pen) \quad (1)$$

The penalty is applied when there is no damage dealt and/or the ship is destroyed, which in this case are 1000 points.

3.3 Function set

We developed functions that would be easily understood by humans, to allow quick-debug and a fast analysis. Another reason for this type of functions is the fact that we

Table 1: Configurations used to insure that only the best will reproduce

Number of military ships	10
Number of pirate ships	10
Military ship	Speed 200 px/s Acceleration 10 px/s Turning Increment 9^o Turning Speed 50ms Hit Points 100 Shield Points 50
Pirate ship	Speed 300 px/s Acceleration 3 px/s Turning Increment 9^o Turning Speed 25ms Hit Points 50 Shield Points 25
Projectile damage	5

only wanted to use functions the player would have "access" too. The attack and flee terminals are behaviours that were previously evolved[4]

- Terminals:
  - Attack
  - Flee
- Functions
  - ifTargetLeft
  - ifTargetRight
  - ifTargetFront
  - ifTargetWeaponRange
  - ifLeftTarget
  - ifRightTarget
  - ifBehindTarget
  - ifAggressorBehind
  - ifAggressorFieldOfView
  - ifWeaponRangeAggressor
  - ifShieldPower
  - ifHullPower

3.4 Random Opponent Evolution

We used random opponent evolution to compare results with the classical co-evolution approach. The random opponent evolution is faster but not as good as the co-evolution approach since the individuals do not combat every individual from the opposite species. Figure 2 illustrates this process. This enabled us to get faster results, but not as generic as the results from the co-evolution approach.

With this approach we were able to evolve individuals faster than co-evolution, 1 hour compared to the 10 hours co-evolution took, and at the same time obtain some interesting results.

In almost all the runs, the pirates best behaviour was the one were the pirate used its speed to combat the military, employing hit and run tactics. The Pirate normally gets

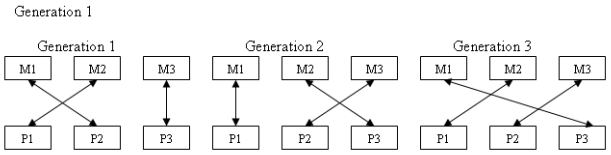

Figure 2: Random Opponent Evolution

close to the military, shoots a few projectiles and then puts some distance between itself and its target, avoiding the military weapon range. Since the pirate's ship is faster and more agile the military cannot fight back and this normally means that the pirate caused severe damage without getting hit.

Figure 3 shows the evolution process of the Pirate species in Random Opponent Evolution.

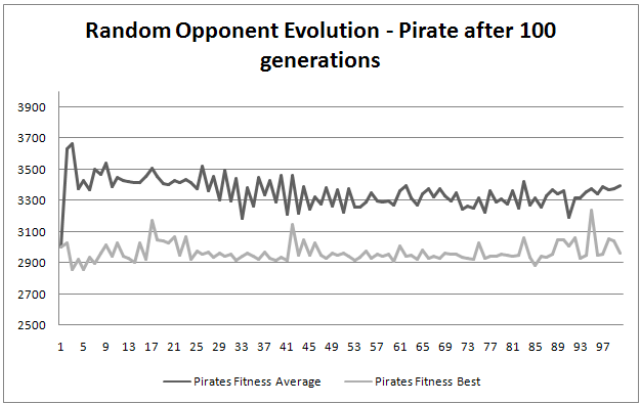

Figure 3: Random Opponent Evolution Fitness Results - Pirate

Best R.O.E Pirate decision tree:

```
(if-left-target (if-aggressor-behind Retreat (if-target-left
    Attack (if-left-target Attack (if-left-target (if-aggressor-behind
    Retreat (if-target-left Attack (if-left-target
    Attack (if-left-target Attack Attack)))) Attack))))
    Attack)
```

The best military result, was one where the military would let the pirate get close and then try to lay some hits, using its superior ship to his advantage. The military lured the pirate, by "faking" a retreat, and as soon as the pirate was upon it the military would start its attack run. This is a very interesting result, and one that is plausible in the real world.

Figure 4 shows the evolution process of the Military species in Random Opponent Evolution.

Best R.O.E Military decision tree:

```
(if-aggressor-weapon-range (if-target-weapon-range
    Attack Retreat) (if-target-right (if-hull-power
    (if-hull-power (if-target-front Retreat
        Retreat) (if-target-left Retreat Attack) (if-behind-target
        Retreat (if-shield-power Retreat Retreat Retreat))) (if-aggressor-behind
    (if-behind-target Attack Attack) (if-target-weapon-range
    (if-target-weapon-range (if-aggressor-weapon-range
        (if-left-target Retreat (if-target-left Retreat
            Attack)) (if-aggressor-weapon-range (if-target-front
        (if-shield-power Retreat Retreat Attack) (if-shield-power
        Retreat Retreat Attack)) (if-aggressor-weapon-range
        Attack Retreat))) (if-aggressor-behind (if-aggressor-behind
        (if-shield-power Retreat Retreat Retreat) (if-aggressor-weapon-range
```

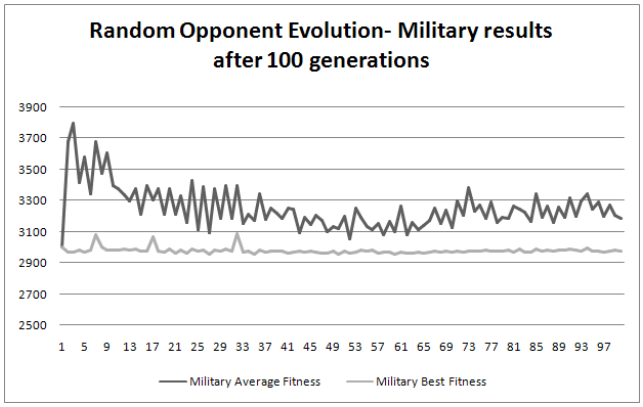

Figure 4: Random Opponent Evolution Fitness Results - Military

```
    Attack Retreat)) (if-target-weapon-range (if-behind-target
    (if-aggressor-fov (if-target-weapon-range
        Attack Retreat) (if-aggressor-weapon-range
        Attack Attack)) Attack) (if-aggressor-weapon-range
    Attack Retreat)))) (if-aggressor-weapon-range
(if-shield-power Retreat Retreat Retreat) (if-aggressor-weapon-range
(if-aggressor-fov Retreat Retreat) (if-aggressor-weapon-range
Attack Retreat))))) Attack) (if-aggressor-fov
(if-right-target (if-target-left (if-hull-power
    (if-right-target Attack Attack) (if-target-right
    Attack Attack) (if-aggressor-weapon-range
    (if-aggressor-behind (if-aggressor-weapon-range
        Attack Attack) (if-hull-power (if-target-front
        Retreat Retreat) (if-aggressor-weapon-range
        Attack Retreat) (if-aggressor-fov Retreat
        Attack))) (if-hull-power (if-aggressor-behind
    (if-shield-power Retreat Retreat Retreat) (if-hull-power
    Attack Attack Retreat)) (if-target-weapon-range Attack
    Retreat) (if-target-weapon-range (if-target-right
    Retreat Retreat) (if-target-weapon-range Attack Attack)))))
    (if-aggressor-weapon-range (if-aggressor-fov
        Attack Retreat) (if-aggressor-fov Retreat
        Attack))) (if-target-weapon-range (if-aggressor-weapon-range
    Attack Retreat) (if-target-weapon-range Attack Attack)))
(if-aggressor-weapon-range Attack Attack))))
```

Although this approach evolved some interesting results, it revealed some problems. Since the individuals are not tested against every other individual of the other species, as in the classical co-evolution process, the evolved behaviour is not as generic as it should be. This means that a good random opponent evolution result maybe better than a co-evolution result for a specific problem, but when a new problem arises it can't cope with it as well as the co-evolution results can. This was observed when we tried to use a capable pirate behaviour and pitted it against a military behaviour from another run.

We also evolved behaviours using other attack and flee trees[4], and the results achieved the main goal while employing tactics that took advantage of the attack and flee trees strong points. Even the worst attack and flee trees could be used to achieve the main goal.

3.5 Classic Co-Evolution

Since we decided to use co-evolution, we created an evolution process in which each individual of a species fought against all individuals of the opposing species. Every Military had to fight every pirate, and vice versa and only then would a round be completed and the evolution process allowed to continue. Figure 5 illustrates this process. This approach was very time consuming, nearing 10 hours per run, but wielded the most interesting results since the best individual from a generation was the individual that had the best average score, since it fought against every opponent.

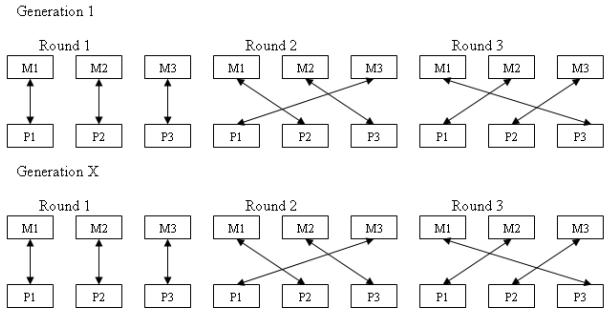

Figure 5: Classical Co-Evolution

In terms of the pirate results, almost all of the results showed that the pirate evolved in a way to use its ship speed to gain the advantage and at the same time to value its ship.

In all the runs not one pirate was destroyed. There was also one result in which the pirate employed hit-and-run tactics, which was chosen as the "best result". This pirate result used its ship speed to stay out of the harm's way (aggressor's weapon range) while manouvering its ship so that it would be behind its target. We can say that this fitness function and evolution process is ideal to evolve a Pirate.

Figure 6 shows the evolution process of the Pirate species in Co-Evolution.

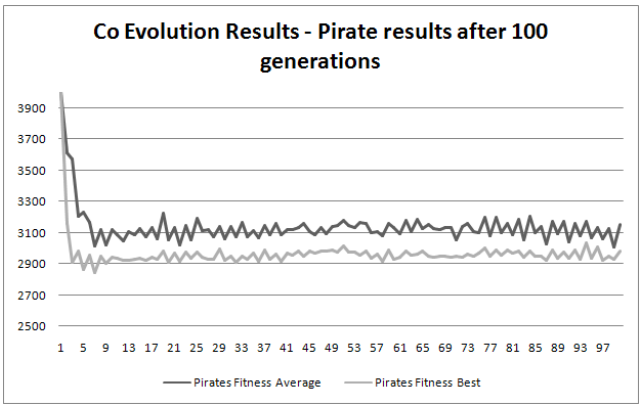

Figure 6: Classic Co-Evolution Fitness Results - Pirate

Co-Evo Pirate Decision tree:

```
(if-aggressor-fov (if-left-target (if-aggressor-weapon-range
    (if-hull-power Attack Attack Retreat) (if-target-front
    Retreat Attack)) (if-hull-power (if-target-weapon-range
    Retreat Retreat) Attack (if-right-target Attack Attack)))
    (if-left-target (if-aggressor-fov (if-left-target
        (if-aggressor-weapon-range (if-target-right
            (if-left-target Retreat Attack) (if-aggressor-fov
            Retreat Attack)) (if-target-front Retreat
            Attack)) (if-hull-power (if-target-weapon-range
        Retreat Retreat) Attack (if-right-target Attack Attack)))
        (if-agressor-tras (if-target-right (if-target-left
            Retreat Attack) (if-aggressor-fov Retreat
            Attack)) (if-target-weapon-range (if-target-left
            (if-hull-power Attack Attack Retreat) Attack)
            (if-tras-target Attack Retreat)))) (if-hull-power
        (if-target-weapon-range Retreat Retreat) Attack (if-right-target
        Attack Attack))))
```

The military on the other hand did not fare, fitness wise, so well since its ship is slower than the pirates ship. The best

result was a military that tried to intercept its target, and at the same time use its ship hit points has an advantage. Upon inspection of the resulting tree we found that the best military result only used the attack pattern, suggesting the best way to deal with a pirate is just to attack. We believe this is so because of the way the attack behaviour deals with fast targets. This shows that this evolution process can create a combat algorithm that takes advantage of the basic behaviours strong points. While not as complex as most of the other results, this also shows that the fitness function was correct and that one could use this evolution process to evolve a "peace keeper".

Figure 7 shows the evolution process of the Military species in Co-Evolution

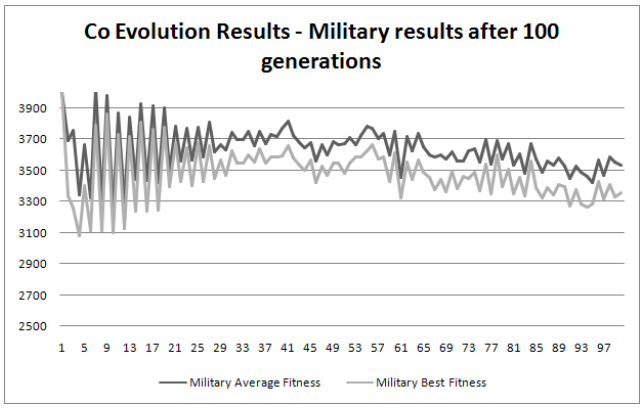

Figure 7: Classic Co-Evolution Fitness Results - Military

Co-Evo Predator Decision tree:

```
Attack
```

When compared to the Random Opponent Evolution, the results are less complex but much more generic. A pirate behaviour evolved via co-evolution could for instance fight a random military behaviour and still achieve its goal, whereas a pirate from the Random Opponent Evolution could not.

4. HUMAN COMPETITIVENESS

To test the competitiveness of the evolved algorithms against human coded algorithms we chose the best results from both approaches and pitted them against simple, yet effective human coded algorithms. We used the same rules as before, to insure no competitor had the advantage. To better explain the test procedure we give an example. For instance, to test the human competitiveness of the co-evolution approach we first chose the best result, Result *A*. We then pitted the human coded algorithms against each other, taking note of their results. Afterwards we tested the pirate from Result *A* against the human coded military, and the military from Result *A* against the human coded pirate. If the evolved algorithms scored better or close results to the human coded algorithms, we considered them human competitive.

The algorithms we develop to test our approaches were based on the following: a Pirate that employed hit and run tactics, and a never-say-die Military.

Our Military only flees when it is on the aggressor's weapon range and its ships hit points are bellow 33%. Otherwise it attacks

Our Pirate employs hit and run tactics, avoiding its aggressor's weapon range at all time. If the pirate is out of the aggressor's weapon range, it attacks using its superior speed

4.1 Co-Evolution Human Competitiveness

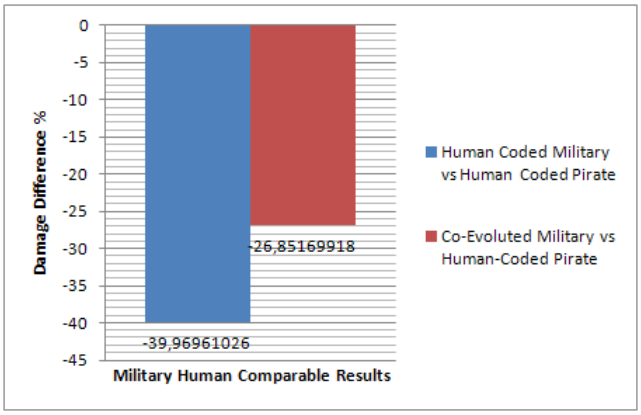

Figure 8: Classic Co-Evolution Military Human Competitiveness

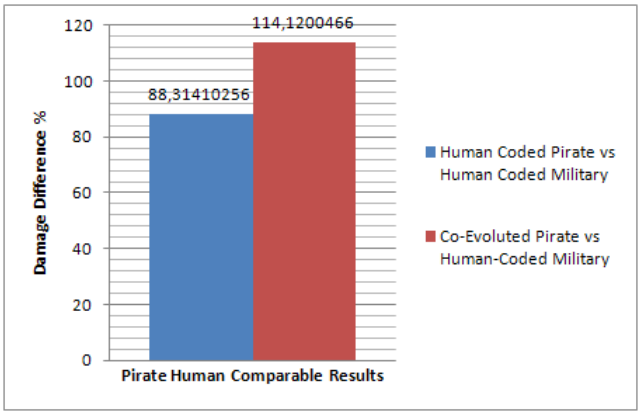

Figure 9: Classic Co-Evolution Pirate Human Competitiveness

When we compared the Co-Evolution approach algorithms to the human coded algorithms we noticed that the co-evolution results were better than their human counterparts. The military, with a negative Damage Difference, had a 12% difference and the pirate a 36% difference.

Based purely on numbers one could say that the Co-Evolution approach is ideal to create an algorithm that beats a human coded algorithm. These results also support the previous co-evolution results, that co-evolution produces very generic behaviours.

4.2 Random Opponent Evolution Human Competitiveness

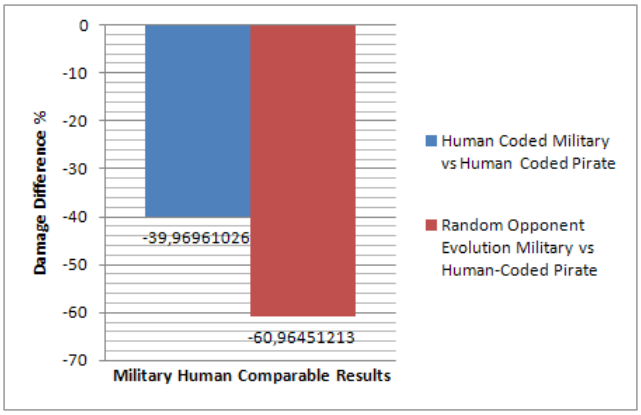

Figure 10: Random Opponent Evolution Military Human Competitiveness

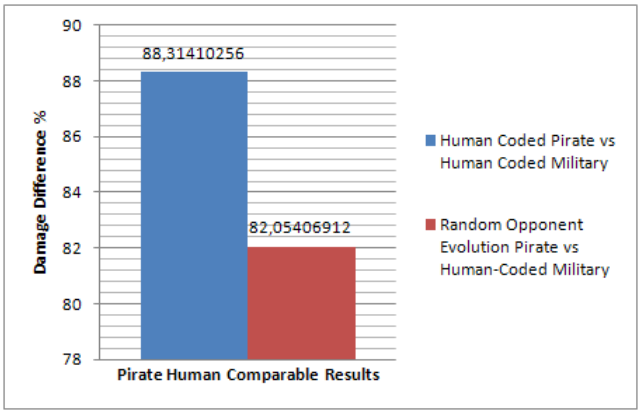

Figure 11: Random Opponent Evolution Pirate Human Competitiveness

The Random Opponent Evolution wasn't so successful in producing human competitive results. While the pirate result was up to par with its human counterpart, with only a -6% difference, the military scored worse with a -20% difference. The low military score is due to the fact we previously mention, that the Random Opponent Evolution produces non-generic results. Also when analysing the "test" we realized that the human coded pirate exploited the military weaknesses. This doesn't mean the approach should be considered invalid, since it was able to produce a very capable pirate and with time, or luck, one could evolve a capable military. Also it took only one hour to evolve these results, whereas the co-evolution approach took 10 hours.

5. CONCLUSION

It can be concluded that it is possible to use genetic programming to create various combat strategies, strategies that take into account the characteristics of the vehicle that the behaviour will control. The algorithms generated with genetic programming can, in some cases, rival those done by hand (hard coded). Also one must take into account that the approaches suggested in this paper evolve combat tactics based on the "basic behaviours" it has. If one were to change these basic behaviours, one would only have to get new results, whereas a hard coded solution would have to be rewritten. We also showed the theory behind the evolution process. When we used various attack and flee tress, all the resulting behaviours achieved the main goal, deal more damage than it receives employing different tactics and strategies, according to its attack and flee behaviours.

Although some results don't fare so well, there are those that are good to challenge hard coded ones. On top of that, with this solution it is possible to evolve any number of algorithms for any number of problems, games or real-life applications.

We also showed that there are two different approaches that can be used to create an acceptable combat strategy. One approach showed that it could evolve decent behaviours that were not generic in a small time frame while maintaining a hint of human competitiveness. On the other hand if there is no time limit, the co-evolution approach seems the best since it produce the best results, both human-competitively and generically.

6. FUTURE WORK

We would like to improve the Random Opponent Evolution approach, in a way that will enable us to get better human competitive results. We would also like to redo the entire experiment with the following characteristics: Instead of different ships, both military and pirate would have the same ship but different fitness functions. We would then compare those results to the ones present in this paper, and see which approach is the best. In the near future we would like to evolve team tactics, so that military and pirate cooperate with their species for better results.

7. REFERENCES

[1] J. Westra. Evolutionary neural networks applied in first person shooters. Master's thesis, University Utrecht, March 2007.

[2] N. Cole, S. Louis, and C. Miles. Using a genetic algorithm to tune first-person shooter bots. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 139–145, Portland, Oregon, 20-23 June 2004. IEEE Press.

[3] D. Fogel, T. Hays, and D. Johnson. A platform for evolving characters in competitive games, 2004.

[4] T. Francisco and G. Reis. Evolving predator and prey behaviors with co-evolution using genetic programming and decision trees. In *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, New York, NY, USA, 2008. ACM Press.