Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 1. pp. 71–77.

Virtual Military Robot Controlled by Emotical Agents Using Reinforcement Learning*

András Fülöp[a], Márton Ispány[b]

[a]University of Debrecen Faculty of Informatics Department of Information Technology
e-mail:fulibacsi@gmail.com

[b]University of Debrecen Faculty of Informatics Department of Information Technology
e-mail:ispany.marton@inf.unideb.hu

Abstract

Remote controlled and (later) intelligence driven military devices have been existed for a long time. Probably, in the wars of the future robots will fight each other instead of soldiers, avoiding human casulties. An abstract view of the tank's battles is the Jrobots competition, in which two or more virtual tanks fight against each other. These virtual robots are programmed in advance (in Java language) using built in heuristics in order to win. However, these programmed artificial intelligencies can be exceeded by machine learning methods. In this talk we introduce an emotical agent which learns the optimal strategy and therefore the optimal emotion, using a reinforcement learning method called SARSA(λ). Using three different behaviors - called emotions - which are pre-programmed basic heuristics, the agent can defeat a more complex artifical agent (which defeated the behaviors one by one).

Keywords: Reinforcement Learning, SARSA(λ), Emotical agent, Neurotic agent, JRobots, Artificial Intelligence

MSC: 68T05

1. The problem and related work

Creating an artificial intelligence as an opponent is a challenging task for programmers. The easiest way to solve this problem is to develop simple heuristics, but the results are usually not as satisfying as it supposed to be. To create "smart"

*Thanks to: Péter Jeszenszky

enemies, the programmers must use much more sophisticated ways which are usually performed better but also takes much more development and CPU time. Also, these intelligences are still predictable by the user.
There are different approaches to this problem, such as different machine learning algoritms, like reinforcement learning [7], artificial neural networks (see Kalalian et al.[4]), the inclusion of emotions to the agents [2, 3, 8], or even the mixture of the two fields (like [6, 9]).
In our discussion we used a new approach to this mixture: a reinforcement learner (subsection 3.1) and an emotic agent (subsection 3.2) which utilize emotions in a different way.

2. The environment: JRobots2k9

The Jrobots2k9 competition [1] is an online java-language programming battle since 1999 and is a successor of the famous crobots c-language programming battle. As its name suggests it is about programming robots to fight against each other.
The main goal is to destroy the opponent's robot on a $1km^2$ large square shaped field. Every bot can move with the same maximum speed ($100\frac{m}{s}$) and can accelerate with the same rate ($5\frac{m}{s^2}$). The battling agents has 180 seconds to destroy their opponent or the current match ends up as draw. To accomplish theirs goal, every bot can scan around to find a target, and in case of it finds one, it has a turret with which can fire 1 rocket per second. Every rocket is shot with $300\frac{m}{s}$ speed and has a maximum range (700 m) where it explodes. Depending on the distance between the center of explosion and the robot, one can take 10, 5 or 3 damage point, respectively. This rule applies to the shooter as well. If an agent reaches 100 or more damage, it is considered to be destroyed and the current match ends up, and the other bot wins.
The system itself has two restrictions based on its structure:

1. To ensure that each programmer has the same chance to win, a robot class was created and the competiting robots must be the instances of its subclass, and only the functions implemented in the forementioned class can be used. Therefore advanced programmers can't take advantage on newcomers just because they are more experienced in the java-language.

2. The robots get status information only about themselves, none about the opponent, and none about if they won or lost.

These two restriction makes the designing of a successful learning agent a hard task. Because of the first rule, writing and reading to and from a file, so thus storing learned weights, is impossible. To avoid this problem the developed agent cannot be used on the competition itself, every opponent must be downloaded and be tested offline. To solve the problem which arises from the second rule, the reward system must be designed in a different way.
During our work we used 5 different robots. Four of them came with the offline version of the system, and one downloaded directly from the project's webpage:

1. **Rabbit:** is only walking in a random path without shooting.

2. **Rook:** randomly chooses from a set of directions (up, down, left right), and start moving towards. It can detect and fires at opponents but only if they are left, right, up or down from Rook.

3. **Sniper:** randomly chooses a corner and starts moving to the selected destination. Once it arrives, starts shooting at enemies till it got hit. Once it happens it chooses an another corner to snipe from and starts its process again.

4. **Counter:** scans for an enemy and once finds one, starts firing upon it. When it got hit, starts moving to evade further damage.

5. **Platoon:** is much more complex then the four default agent. It is an example robot used to show how to program a basic intelligence. It scans around and firing all the time also constantly moving.

3. A solution: Emotical agent - a behaviour based learner

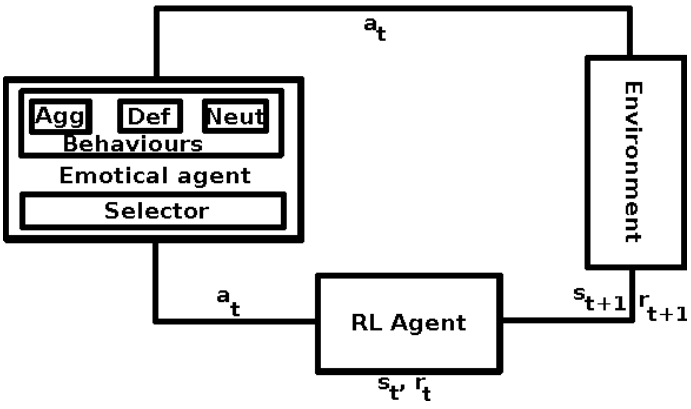


Figure 1: The developed - two agent - system.

In this paper we present a new approach to the forementioned problem. We built a hybrid model which consists of two different agents: a learning agent and a behavioural agent which ones work together. The connection between these two agent is shown in figure 1. The basic idea is that the reinforcement learner picks an action and based on this, the emotical agent selects a behaviour. Every behaviour is a pre-programmed heuristic whose actions brings the agent system into the next

state, and collects the reward from the environment. Then the learner decides what to do, and so on. In the following we describe the two agents.

3.1. Reinforcement Learner Agent

The built agent is based on the SARSA(λ) method [7]. There are two main questions in connection with the learner; which given parameters could be used as states, and which criterion determines the reward. In order to compute the current state, we selected three components to build from: the current position, the actual health points and the remaining time. If we simply get the raw data from these three parameters, we would have to work with $1.8E+10$ states. We wanted to decrease this enormous number. To achieve this goal we made the following changes in the system.

The current position is not simply the agent's current coordinates, instead, we

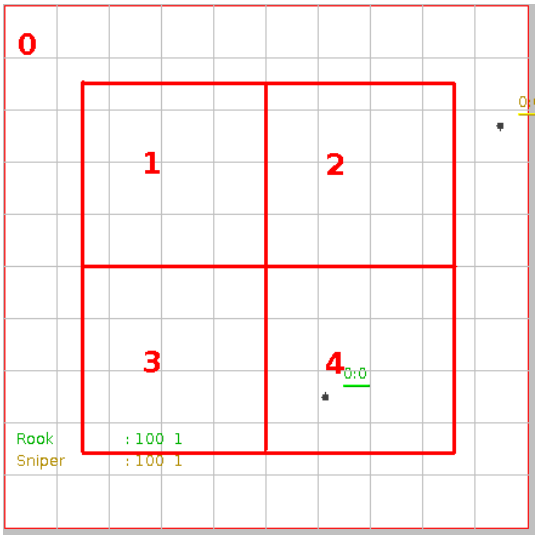


Figure 2: Partitioning the field.

devided the field into 5 section, and the position is the number of the actual section (as depicted in figure 2). The second component came from the amount of health points left, but - again - instead of current values, we devided the health bar into 4 parts (fully healthy, injured, severely injured, almost dead) and the amount of health is transformed into the forementioned 4 values. Last but not least state-building component is the remaining time discretized into 4 value, showing how many quarters are left from the game. With these changes we forced down the space of state's dimension to 80. The other important part of building a reinforcement learner agent is the determination of the reward system. Because of the problems mentioned in section 2, the agent don't know anything about it's

opponent, so the obvious choose as a reward, like a successful hit or destroying the enemy can't be implemented. Instead we measured the damage received between two states and gave the apropriate amount of reward depending on the change of the health points.
The set of actions - given to the hands of the learning agent to choose from - is narrowed down to 3 choises; it gives direct order to the emotic agent to choose a behaviour.

3.2. Emotical agent

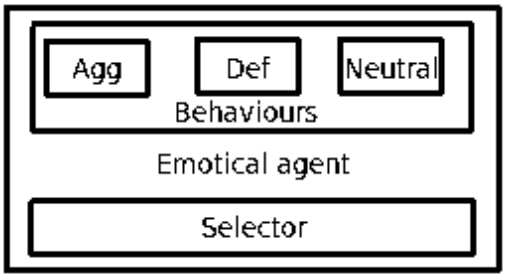


Figure 3: Structure of the emotical agent.

The emotic agent consists of two parts: a selector, and a behavioural part. The selector chooses one from a set of behaviours which are pre-programmed heuristics, each has a different action-pattern. As the figure 3 shows, in our agent system we programmed three different pattern: aggressive, defensive, and neutral. Each behaviour uses the same routines, like scanning, aiming, and targeting, the difference is in the way they utilize their movesets. Before using them, each one was tried out as a separate intelligence against built-in agents (about their performance we talk in section 4).

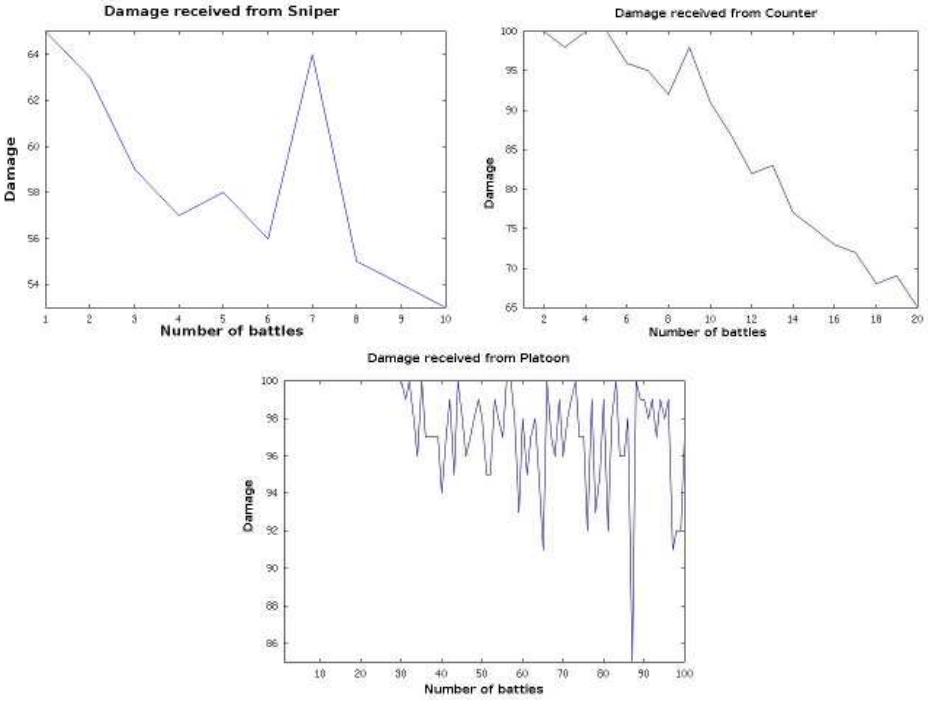
1. The **aggressive** behaviour reflects its name: it chases its enemy and shoots toward it.
2. The **defensive** behaviour is the opposite, it's constantly fleeing and firing.
3. The **neutral** behaviour is patrolling in a random route and firing at the enemy.

4. Results and further work

Our test can be devided into two parts. First we tested how our heuristics perform against different built-in intelligences, and in the second run we tried our dual-agent intelligence against the same bots.

Instead of winning or losing, we measured the amount of damage received from the opponents through the matches, because of the restrictions mentioned in section 2. From these data one can conclude the final outcome. First the heuristics described in section 3 were tested against the built-in robots. After a few testing cycle we dropped out *Rabbit* and *Rook* because every agent won without receiving significant damage. The remaining three agent won against our heuristics in the 90% of time. The next stage was the testing our agent, *Béla3*.

4.1. Performance



The test includes matches against three opponents: *Sniper*, *Counter* and *Platoon*. One can see the results in the figures above. The agent improved over the iterations when fought against *Sniper* and *Counter*, but failed against *Platoon*.
The results shows that our agent failed to defeat a more complex agent, thus learn an optimal strategy, which could have several causes, such as low dimension of states, weak behaviours, low number of behaviours or even wrong concept.

4.2. Further work

In order to improve the effectiveness of our learner we want to develop a more complex agent. First, we are including emotions in a more sophisticated way, changing the process how the selector is choosing between behaviours. We want to

implement more behaviours, and also more complex ones. We also plan to change the states to make the agent more reactive to smaller changes in the environment. Other way of improvement is to implement more simple behaviours (like fleeing) to outcome a complex moving set.

References

[1] Boselli, L., JRobots - Java Programming Battle Since 1999. *JJRobots webpage* http://jrobots.sourceforge.net/, 2010.

[2] Bozinovski, S., Schoel, P., Engineering goalkeeper behavior using an emotion learning method. *KI99: deutsche Jahrestagung für Künstliche Intelligenz*, Bonn, 1999.

[3] Hermann, C., Melcher, H., Rank, S., Trappl, R., Neuroticism - a competitive advantage (also) for IVAs? *IVA2007 Paris*, http://www.ofai.at/~stefan.rank/presentations/HermannETAL2007-IVA.slides.pdf 2009.05.06.

[4] Kalalian, L., Huang, X. H., Fuerth, J., Replacing heuristic programming with machine learning for a robot in an adversarial environment. *Faculty of Mathematics, University of Waterloo*, http://www.student.math.uwaterloo.ca/~lakalali/MachineLearning.ps, 2001.

[5] Meyer, G., Emotional agents: how personalities change behaviour. - *1st Twente Student Conference on IT*, Enschede, 2004.

[6] Seif El-Nasr, M.,Ioerger, T., Yen, J., PETEEI: A PET with Evolving Emotional Intelligence. *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, Washington, 1999.

[7] Sutton, R. S., Barto, A. G., Reinforcement Learning: An Introduction. *The MIT Press*, London, 1999.

[8] Wright, I. P., An emotional agent: the detection and control of emergent states in autonomous resource-bounded agents. *Cognitive Science Research Report RP-94-21*, University of Birmingham, UK, 1994.

[9] Wright, I. P., Reinforcement Learning and Animat Emotions. *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, P. Maes, M. Mataric, J. Pollack, J.-A. Meyer, and S. Wilson (eds.), The MIT Press/Bradford Books, Cambridge, MA, 1996.

András Fülöp
Egyetem tér 1., 4032 Debrecen

Márton Ispány
Egyetem tér 1., 4032 Debrecen