

Evaluation of proper methods for parameter optimization and skilltesting of a playbook-based artificial intelligence in regards of the small-size-league team Austrian Cubes 2009.

Martin Trapp¹, Raphael Mayer¹, Melanie Proksch-Weilguni¹, Alexander Hofmann¹, Robert Pucher¹

University of Applied Sciences Technikum Wien
Hochstaedtplatz 5, 1200 Vienna, Austria

email: software@austriancubes.at
web: www.austriancubes.at

Abstract. The artificial intelligence software of the Austrian Cubes 2009 is skill based. To adjust skill's behavior different parameter optimization and skilltesting methods have been evaluated. This paper describes the evaluation of proper methods in regards of finding the best property values for a efficient game play and test the functionality of a skill.

1 Introduction

The Austrian Cubes Project is a cooperation of the University of Technology Graz and the University of Applied Sciences Technikum Wien. This cooperation exists since 2008 and is the revival of the Vienna Cubes. The Austrian Cubes Project combines the knowledge of the University of Technology Graz in the field of Mechanics and Electronics with the knowledge of the University of Applied Sciences Technikum Wien in the field of Computer Vision and Artificial Intelligence (AI).

The hardware of the small-size league team Austrian Cubes will fundamentally enhance till the RoboCup 2009. In the course of the new hardware design, the AI will be enhanced and improved. The AI of the Austrian Cubes uses a playbook based system to identify the next decision to be made. To get the best modularity of the AI components, the Austrian Cubes use a three-level architecture where a strategy represents the highest level and a skill represents the lowest level. These skills can be understood as small decision making processes telling the robots what to do. These skills can be very complex, e.g. deciding whether to shoot, chip kick, pass, high pass or get a better position.

2 Technical Background

The components of the three-level architecture, Fig. 1, are strategy, function and skill. At the AI Software of the Austrian Cubes, the strategy defines the

team play, depending on the global situation. During the play each function - e.g. attacker, creator, helper, defender or goalkeeper - will be assigned to a robot on the field. Depending on the strategy each function executes a different skill. These skills can be neutral, function independent, or related to a function. These skills work on the lowest level of the architecture and directly communicate with the orders given to the robot by the radio module. A simplified example of the execute method of the skill `pass to player` is shown in listing 1.1.

Listing 1.1. execute of pass to player skill

```

program execute
  if robotWithBall or robotPassTo == null
  then
    exit
  end

  if passquality > min passquality
  then
    set dribbler control backward
    set speed for pass depending on the distance
    set kicker to lowkick
  elseif chipkick passquality > min chipkick passquality
  and distance to nearest oponent < distance threshold
  then
    set dribbler control backward
    set speed for pass depending on the distance
    set kicker to highkick
  end
end

```

Possible parameter for optimization could be the `min passquality` shown in listing 1.1. These parameters hardly influence the results of the skills and so has to be optimized.

3 Importance of parameter optimization and skilltests

With skilltesting, it is possible to show and review every single skill separately. Otherwise, a whole game has to be played on the simulator and wait until this special skill will be used - of course, that's not very efficient. At the moment, the existing AI of the Austrian Cubes is still in an improvement process, especially the offense. That's why skilltesting is really important at the moment helpful for the improvements. With skilltesting, it is possible to find existing errors and to optimize existing parameters. One of the big aim is to improve the whole AI by getting the best possible property values for each skill. Another point is that the Austrian Cubes got new robots. With the new hardware, new skills are possible. So these new skills - like chip kicking for example - were implemented and it is really important and helpful to test them during the development process. Moreover, the AI is not perfect in every different situation that could occur

in the game. To work on it, it is also important to have the possibility to do skilltesting depending on different situations. Nowadays there are more and more reasons for the implementation of testing skills. The most important one is the Robocup 2009. For this reason alone, skills have to be optimized as soon as possible.

4 Discussion of parameter optimization methods

There are different proper methods for parameter optimization. These methods are divided into manual and automatic optimization methods. To evaluate the best optimization method for the AI, the different methods have to be discussed concerning the best efficiency for the AI of the Austrian Cubes.

4.1 Automatic methods

The key to parameter optimization is to optimize special properties defined in the skill. These properties directly affect the behavior of the skill. For example a possible property could be the shoot quality threshold.

The idea of an automatic method is to use an intelligent agent, e.g. a genetic algorithm described in [michalewicz97], to find the best value for a specific property. It's apparent that this method can be very fast and can find almost perfect results in some cases. Because of the high iteration rate it's possible to test a wide interval of possible values.

However, there are a few major drawbacks. First of all an automatic method, e.g. a genetic algorithm, uses some kind of fitness function to evaluate the best matching values. "What's the Best Answer? It's Survival of the Fittest." [wbaisof90]. The difficulty of this fitness functions is to define how to measure the fitness of a found result. To do so a highly realistic simulator and a possibility to assess the outcome is needed. Regarding the benchmark it has to be defined if it's good when the robot passes a ball or shoots a ball. It's quite challenging to find a well-working fitness function. As said, in order to get good results it's necessary to have a highly realistic simulator. The Austrian Cubes use a really simple simulator without realistic physical simulation, e.g. rigid body simulation. In consideration of the fact that the simulator has no realistic physical simulation it would be a lot of work to setup an automatic parameter optimization method.

4.2 Manual methods

Besides the automatic methods it's also possible to use a manual optimization method. The advantages and drawbacks of this method are that it can be implemented quite easily and can get highly efficient but because of the small iteration rate it's only possible to test a really small interval of possible values. Besides the small interval it's important to mention that the results of this method are still quite good because the user can easily benchmark the outcome.

5 Implementation

The Austrian Cubes, implemented the method of manual testing. A GUI makes it easier to manage testcases and execution. Skills are chosen by functions. This is the reason why the skills get registered in RegisterFunctions. To find the right skill the Austrian Cubes take the actual strategy. Then the robot gets assigned to a special skill. A possible testing would look like this:

- First of all the interval of every property which should be tested has to be defined.
- The possible number of permutations has to be found. It is important to know that permutation means the arrangement of n possible values for the different properties of a skill.

Because of the three-layer model in our AI framework (see Fig. 1) the Austrian Cubes created a new strategy type. The new Framework for the parameter optimization and skilltests registers the strategy types in the robot functions. So, the skilltest framework allows using the skills by selection. Moreover, each skill has its own skilltest strategy whose build-up is similar to the normal strategies. This thought works quite well, but there's one thing not to forget: For example, if the skill "pass" should be tested, the ball's location should be known. Ergo, if the selected robot that has to achieve the skill "pass" does not have the ball then he has to get it with the skill "getBall". Therefore test strategies are needed for each skill but they can and sometimes they have to contain more than one skill too. For upgrading the skills during testing and parameter optimization, property files of a skill were parsed by Java. Each of this property files can be changed and saved during runtime. Therefore its possible to instantly benchmarking and change until the skill works quite well without any interruptions.

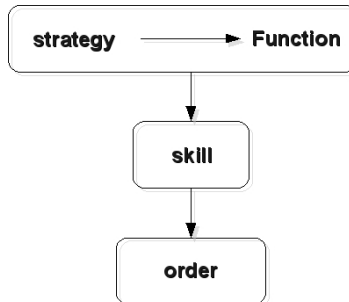


Fig. 1. three-layer model

6 Results

The results of the implemented framework are mainly known bugs or issues. E.g. known issues of potential fields could be recapitulated, described in [KorBor91] which is based on trap situations due to local minima, shown in Fig. 2. These trap situation could be resolved by heuristic recovery. Another significant problem inherent to potential fields would be that the robot is not able to pass among two closely spaced obstacles. This effect can also be identified by Fig. 2.

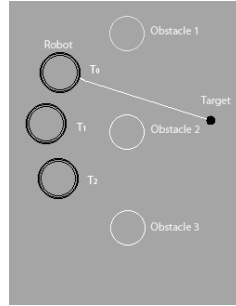


Fig. 2. trap situation of potential field due to local minima.

Other issues are known bugs of the movement system and the attack. Also the outcome of the skills were improved by benchmarking different values for specific properties a lot.

value:	1	10	20	30	40	50	60
rating1:	5	5	4	3	2	1	5
rating2:	4	5	3	4	2	2	5
rating3:	4	5	4	5	1	1	5
rating4:	5	5	5	2	1	1	5
rating5:	3	5	3	3	1	2	5
rating6:	5	5	4	3	2	1	5
rating7:	4	5	4	2	1	2	5
rating8:	5	5	4	3	2	1	5
rating9:	5	5	3	3	1	1	5
rating10:	5	5	5	3	1	1	5
rating avg:	4.5	5	3.9	3.1	1.3	1.2	5

For example here is a scheme for benchmarking the property values. In the first column are the possible values of a specific thresholds of a skill, for example the skill "move to ball". The task is to optimize the value for this threshold. For benchmarking the initial situation has to be the same. Each value should

be tested as often as possible for getting a reasonable average rating. With the previous value and the average rating it is possible to approach to the best feasible value.

Besides bug detection, this framework was used for testing the new decision process of the attack during development.

References

- KorBor91. Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE Conference on Robotics and Automation*, pages 1398–1404, April 1991.
- michalewicz97. Zbigniew Michalewicz, Robert Hinterding, and Maciej Michalewicz. Evolutionary algorithms. *Fuzzy Evolutionary Computation*, (1):336, June 1997.
- wbaisof90. What's the best answer? it's survival of the fittest. The New York Times, August 1990. <http://query.nytimes.com/gst/fullpage.html?res=9C0CE1D6153BF93AA1575BC0A966958260>.