The magmaOffenburg 2013 RoboCup 3D Simulation Team

Klaus Dorer, Stefan Glaser[1]

Hochschule Offenburg, Elektrotechnik-Informationstechnik, Germany

Abstract. This paper describes the magmaOffenburg 3D simulation team trying to qualify for RoboCup 2013. While last year's TDP focused on different ways how robot behavior can be defined in the magmaOffenburg framework this year we focus on how we statistically evaluate new features on distributed systems. We also show some results gained through such analysis.

1 Introduction

This year our focus is on our approaches to get results from changes to our agents. We have developed two ways how to get statistically significant results. For learning runs within one virtual machine as well as for one versus one games on one virtual machine for each agent but located on a single computer we use bash scripts to automate runs and games. For games 11 versus 11 distributed over a cluster of machines we created Codie a Java based application based on the Java job distribution framework (JDF).

Section 2 shows how optimization and learning is speed up by parallel evaluation of utilities as well as some results gained. In section 3 we show how new features are evaluated in one versus one games. Section 4 introduces our job distribution application Codie for distributing game runs on a cluster.

2 Learning Runs

A major source of time consumption for learning runs in non-deterministic domains like RoboCup is that utility calculation only gets reliable after running the same experiment multiple times. In order to speed this up, our existing framework for learning (see TDP 2012) was extended to allow parallel measurement of utilities for a single individuum. The major change has been to separate utility calculation from the representation of an individuum (see Figure 1). Each UtilityCalculation component runs in its own thread of control. It launches its own soccer server, measures the utility and reports back the result to its individuum. It also cares for hanging server instances, by killing and restarting it again.

Results of different optimization runs are shown in Figures 2-4. In this setup we used simulated annealing, genetic algorithms and tabu search to optimize the last phase of our kick behavior. We vary start positions of the agent so that

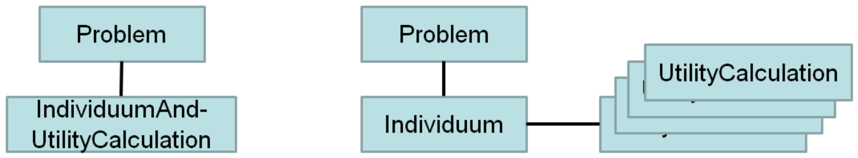


Fig. 1. Before and after separation of inidviduum and utility calculation.

it has to make some steps to the ball before kicking. Since kicking is somewhat unreliable, we have to run many measurements to get a useful utility. On our i7 Hardware we can typically run 30 utility calculations in parallel which is what we did for each individuum.

The simulated annealing run started with a manually found parameter set for kicking with an average utility of 3.7m. Running 1000 annealing steps worsened the result no matter what cooling strategy was used (Figure 2). An overall 30.000 kicks have been performed in this run without any interruption of the run.

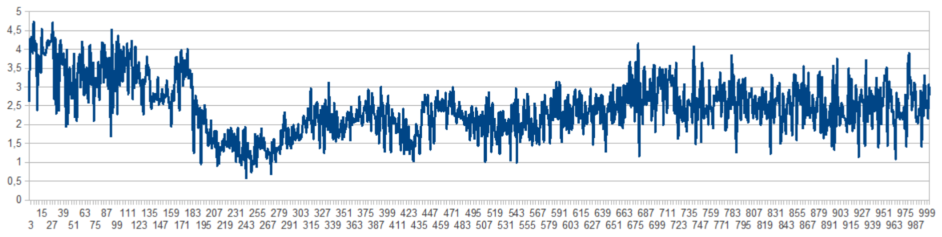


Fig. 2. Utility of a simulated annealing run to optimize kick parameters.

In the genetic optimization run each population consisted of 50 individuums breeded over 100 generations using Monte Carlo selection, multi-crossover reproduction and random mutation (0.2 individuum selection, 0.1 gene selection). Figure 3 shows the decreasing diversity of the agents and the increasing utility (distance) of the kick. The diagram was created from 300.000 kicks running over one weekend without interruption. The final utility was an average kick distance of 3.9m.

In a third run, tabu search was used to optimize kick parameters. Start state has been the manually found working parameters mentioned above. Figure 4 does not show single results of one parameter setup averaged over 30 runs. Instead to also get insight into what parameter area works good on average, 100 consecutive parameter settings were averaged. Each point in the diagram is an average of 3000 kicks performed by 100 slightly different parameters variations.

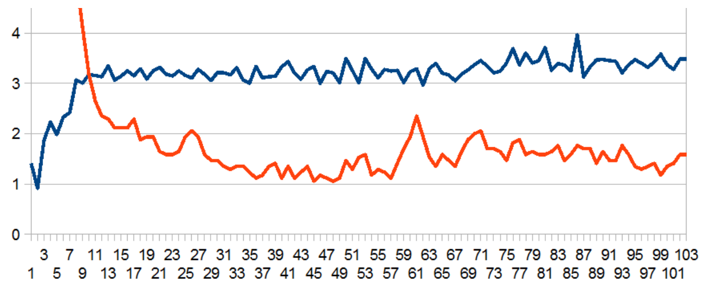


Fig. 3. Utility and gene-diversity of genetic optimization of kick.

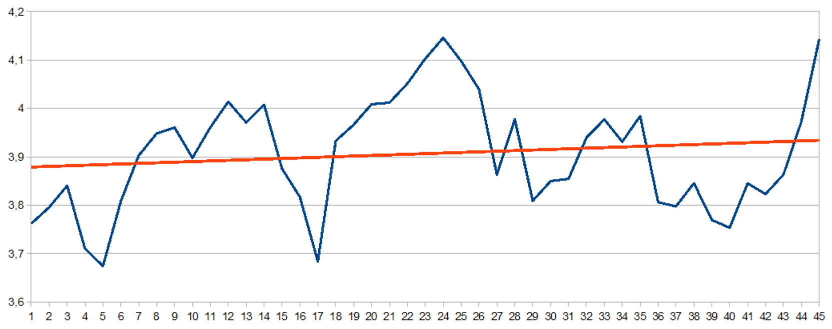


Fig. 4. 100 run average utility of tabu search.

In sum, 132.000 kicks have been performed in this run resulting in an average utility of 4.4m.

3 Evaluation Runs

The learning framework is useful also in evaluation of single game situations. Figure 5 shows the result of evaluating the reliability of our walk. In this scenario agents started to walk straight to the desired forward speed (2 seconds), then switched to the desired forward and backward speed for 3 seconds and continue with just forward walk for another 2 seconds. Each run recorded the speed and if the agent did fall. The x-axis in the diagram shows the intended forward speed (in %), the y-axis the corresponding intended sideward speed. The color represents the number of falls, with green no falls and red more than 5 falls. Each point in the diagram is averaged over 20 runs. It is clearly seen that problems start with 90% forward speed.

In order to evaluate features in whole games, the framework needed to be extended. Figure 6 shows the setup of such game runs. First a trainer component had to be created that is able to start and stop games and get the result.

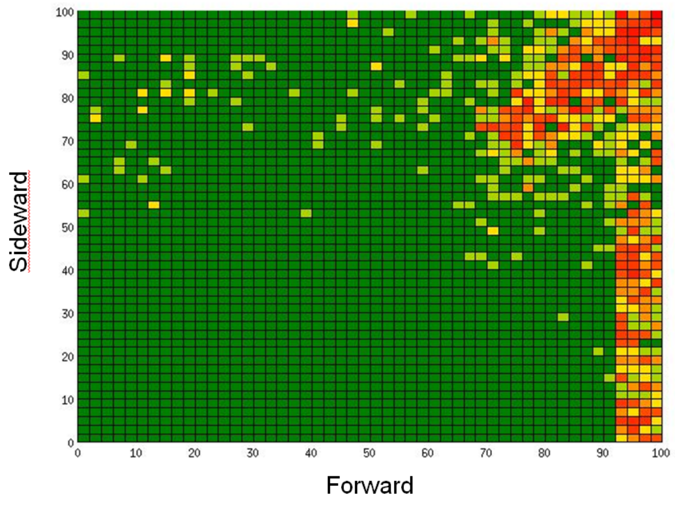


Fig. 5. Reliability of walk with increasing intended forward and sideward speed.

It is built using the monitor protocol of the server. Second, additional bash scripts have to make sure that server, agents and trainer are started and properly cleaned up after the game in all situations and error conditions. Finally, scripts for automatically evaluating the results helped to speedup the process.

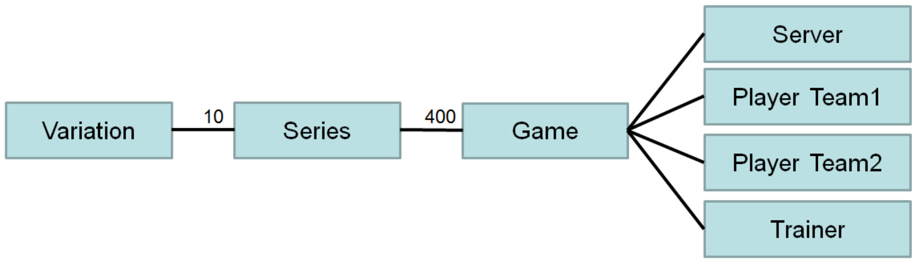
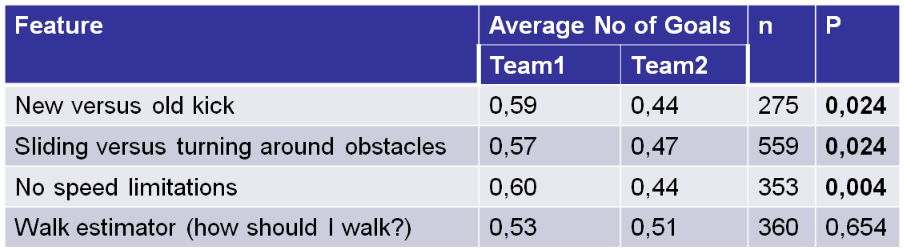


Fig. 6. Architecture for evaluating one versus one games.

Table 7 shows a couple of results gained through running series of games. In such series, two identical agent versions play one half time against each other. Just that the one feature to look into is added to the agent called team1. It shows that the kick learned through tabu search significantly improves game play as did two other features. The Walk estimator feature did not improve game play, but made it nevertheless into our code since it is much more general in its implementation. The game series made sure that we did not introduce any critical mistake. The varying number of games is due to the fact that we did not restart games in which the server was hanging. Initially half of the games were played left to right, the other half right to left. After many games there was no preference to any side visible in the data.



Feature	Average No of Goals		n	P
	Team1	Team2		
New versus old kick	0,59	0,44	275	**0,024**
Sliding versus turning around obstacles	0,57	0,47	559	**0,024**
No speed limitations	0,60	0,44	353	**0,004**
Walk estimator (how should I walk?)	0,53	0,51	360	0,654

Fig. 7. Selected results of game series.

4 Codie

Codie is a Java application that simplifies the distribution of games on a cluster. It makes use of the JPPF framework to distribute jobs on a cluster of computers (Figure 8). Special plugins can be developed to support domain specific requirements of runs. The RoboCup plugin e.g. makes sure that the server is up and running before the players are started. It also cares for collecting the results from the Trainer job.

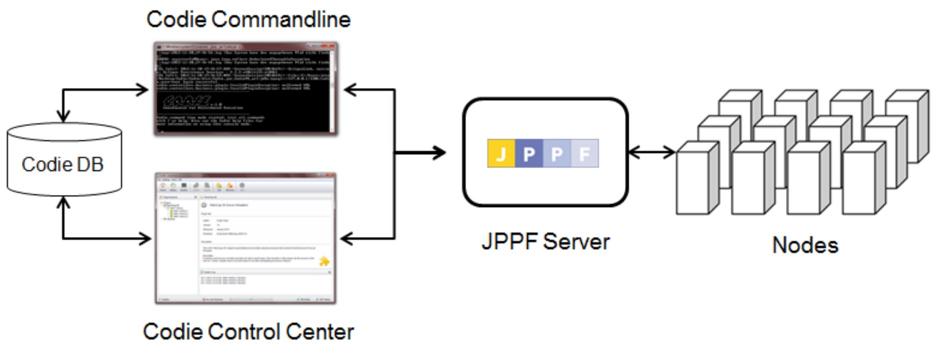


Fig. 8. Architecture of Codie.

Features include automatic runs of games, storing results in a database, statistical evaluation of results, definition of service levels of jobs and many more. A command line version is available to start remote controlled game runs.

5 Team

The magmaOffenburg team:

- Klaus Dorer (Team leader)
- Stefan Glaser
- Maximilian Krög

References

1. Ritter, B.: Steuerung simulierter Roboter mit Kinect. Bachelor thesis, Hochschule Offenburg, Germany (2012)
2. Kurth, M.: Steuerung eines simulierten Roboters mit Hilfe von Motion Capturing Files. Bachelor thesis, Hochschule Offenburg, Germany (2011)
3. Schindler, I.: Laufen auf zwei Beinen in der simulierten RoboCup 3D-Umgebung. Bachelor thesis, Hochschule Offenburg, Germany (2009)
4. Dorer, K.: Modeling Human Decision Making using Extended Behavior Networks. J Baltes et al. (Eds.): RoboCup 2009, LNAI 5949, pp. 81–91. Springer, Heidelberg (2010)
5. Dorer, K.: Extended Behavior Networks for Behavior Selection in Dynamic and Continuous Domains. In: U. Visser, et al. (Eds.) Proceedings of the ECAI workshop Agents in dynamic domains, Valencia, Spain (2004)
6. Dorer, K.: Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten. PhD thesis, Albert-Ludwigs University (2000)
7. Dorer, K.: Behavior Networks for Continuous Domains using Situation–Dependent Motivations. Proceedings of the Sixteenth International Conference of Artificial Intelligence (1999) 1233–1238