Development of a Optimization Assistance Tool for 3D Simulation League

Marco A. C. Simões[1] Claudia Elizabete Emmanuel Argollo[1] Mateus Nascimento Sergio Souza Jr[1]
Wellington Santos[1] Josemar Souza[1]

A well developed team for the 3D Simulation league uses a wide variety of motion behaviors to execute complex strategies during a match. While the behaviors are developed either with a mathematical dynamic model or a sequence of static poses to be executed, the initial movement parameters to perform the behavior are defined mostly empirically. In order to fine tune these parameters, to achieve an optimal version of the behavior, the team must employ an optimization algorithm.

The process of optimizing a behavior consists of determining the desired parameters for the optimization, developing a situation for the team to execute the behavior, and creating a fitness function to evaluate how well was the execution of said behavior. The optimization algorithm is then responsible for using the fitness value along with the tested parameters to generate a new set of parameters, in order to find a better fitness value. This execution is then performed in a loop until it reaches an acceptable value.

With this generic optimization cycle in mind, we decided to create a tool with which we could easily create optimization scenarios and run the optimization loop. To do so we used the *MagmaChallengeBenchmark* [1], developed by the Magma Offenburg team as the basis for our tool, as it can be easily used to create scenarios for the optimization and attribute its fitness value. As for the optimization we decided to create this tool initially with the Covariance Matrix Adaptation Evolution Strategy (*CMA-ES*)[2], as it could later be used to perform the optimization in parallel. Previous researches have also found this algorithm to be efficient in a setup similar to ours[3].

With these two processes, the tool to perform the tests, and the optimization algorithm to generate new parameters, we modified them using a simple pipe file as the interface between them. Starting with the *CMA-ES*, it would generate a determined amount of sets of parameters, as a population, which would be tested by the modified challenge tool, that would write the fitness results to a file, to be read by *CMA-ES*, continuing the loop. The tool GUI can be seen in figure 1 an its process can be seen in the figure 2.

Then, in order to decrease the optimization time, we modified the testing tool to be able to perform the optimization in parallel. The *CMA-ES* side of the optimization would run the same way as before, however, the testing tool would be run as multiple processes, following a master slave paradigm.

[1]ACSO, DCET-I, UNEB, Salvador, BA, Brazil. teambahiart@gmail.com

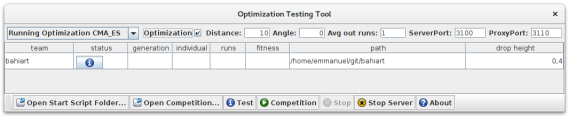

Fig. 1. Optimization Tool GUI

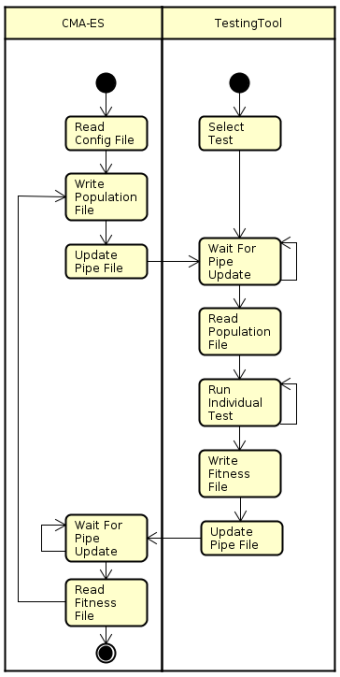

Fig. 2. Optimization proccess with tool

The master process would divide the population among the slaves, which would run individual tests in parallel, and then return the fitness to the master, that would order the results into the file to be read by *CMA-ES*.

References

[1] K. Dorer, J. Fischer, S. Glaser, D. Nguyen, M. Obrecht, and D. Weiler, "The magmaoffenburg 2016 robocup 3d simulation team," in *Proceedings of RoboCup 2016*, 2016.

[2] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[3] D. Urieli, P. MacAlpine, S. Kalyanakrishnan, Y. Bentor, and P. Stone, "On optimizing interdependent skills: A case study in simulated 3d humanoid robot soccer," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 769–776.