Optimization of Robot Movements Using Genetic Algorithms and Simulation

Brandon Zahn, Jake Fountain, Trent Houliston, Alexander Biddulph, Stephan Chalup, and Alexandre Mendes(✉)

School of Electrical Engineering and Computing, Faculty of Engineering and Built Environment, The University of Newcastle, Callaghan, NSW 2308, Australia
Alexandre.Mendes@newcastle.edu.au

Abstract. This work describes the optimization of two robot movements in the context of the Humanoid league competition at RoboCup. A multi-objective genetic algorithm (MOGA) was used in conjunction with the real-time physics simulator Gazebo. The motivation for this work was that the NUbots team, from the University of Newcastle, lacked a simulation platform for their soccer-playing robots. Gazebo was the preferred choice of simulator, offering built-in compatibility with the Robot Operating System (ROS). The NUbots robot software, however, uses a proprietary message-passing framework in place of ROS. This work thus describes the pathway to use Gazebo with non-ROS compliant applications. In addition, it describes how MOGA can be used to optimize complex movements in an efficient manner. The two robot movements optimized were a kick script and the walk engine. For the kick script, the resulting optimal configuration improved the kick distance by a factor of six, with 50% less torso sway. For the walk engine, the forward speed increased by 50%, with 38% less torso sway, compared to the manually-tuned walk engine.

Keywords: Simulation · Walk engine · Optimization · Multi-objective

1 Introduction

The use of optimization to fine-tune the movement of robots has been the focus of attention in recent years, particularly in the context of legged robots [3]. Those types of robots normally use controllers that allow them to maintain balance while performing complex tasks such as walking, turning and jumping. Some of those controllers might have several parameters, depending on the tasks that they perform, which need to be tuned. Just as an example, the walk controller used in this work has 46 parameters that can be individually tuned and finding the right trade-off between balance and speed is a difficult task. The work presented here is significant due to the combination of tools that it uses (the Gazebo simulator and a proprietary message-passing framework called NUClear) and the challenge that it posed. The lessons learned in terms of software engineering and how to effectively make the two systems communicate are

S. Chalup et al. (Eds.): RoboCup 2019, LNAI 11531, pp. 466–475, 2019.
https://doi.org/10.1007/978-3-030-35699-6_38

important contributions. From the perspective of optimization, the contribution is the demonstration of how NSGA-II [4] manages to obtain very good solutions in short periods of time and requiring few generations, which is an important aspect when the evaluation of individual solutions is a costly task.

Our work follows on the footsteps of other multi-objective optimization studies conducted on legged robots. Among those we cite [8], who used NSGA-II to optimize a walk controller for a quadruped robot. More recently, we have the work of Juang and Yeh (2017) [6], who also used NSGA-II in the optimization of a walk controller for a biped NAO robot. The results presented here indicate that the approach is suitable for the optimization of simpler movements (e.g. a kick script), as well as a considerably more complex walk engine.

2 Kick Script and Walk Engine Optimization

This work was developed with the NUgus robotic platform in mind [2]. The original *igus* platform was developed by the University of Bonn as an open platform in 2015 [1] and the NUbots team made a number of modifications to the original model, naming it NUgus afterwards. A simulation model of the NUgus was also derived from the original igus model, and used in our tests.

Two optimization problems are addressed. The first one is the tuning of the kick script used by the NUgus to kick the ball. The script consists of a sequence of six target positions for each of the 20 motors in the robot, i.e. they correspond to a sequence of poses. The first values in the sequence refer to the initial positions of each motor when the kick is triggered, and the last values refer to their final position after the kick is concluded. The four intermediate positions are strategic poses, e.g. the pose when the kick leg is brought backwards; the pose when the kick leg is at its maximum extension after ball contact, etc. The optimization goal is to find the positions of the motors in each of the four intermediate poses so that the ball's travelled distance is maximized, and maximum torso sway is minimized (to reduce the chances of the robot falling over while the kick is being executed). In order to provide information about how a solution for the problem is encoded, the values for motor positions are integers in the interval [0, 1024]. Therefore, there are 80 integer variables to be optimized, all of them within the interval above.

The second optimization problem refers to the walk engine, and the goal is to optimize 46 parameters used by the controller responsible for moving the robot forward. The goal in this case is to maximize forward speed, while minimizing maximum torso sway. The parameters are represented by real numbers in different ranges for each parameter. Again, to give an indication of the search space size for this problem, the largest interval among the controller's parameters is [0, 100].

3 Methodology

In this section we will explain four elements of this study: (1) the NUbots software system, (2) the Gazebo simulator, (3) the communication middleware and (4) the multi-objective genetic algorithm.

3.1 Nubots Software System - NUClear

In recent years, the NUbots developed a proprietary software architecture named NUClear [5]. It is a framework designed to aid in the development of real time modular systems and is built from a set of C++ template meta-programs that control the flow of information through the system. These meta-programs reduce the cost of routing messages between modules, resulting in faster communication times. Since NUClear utilises a co-messaging system, it allows for simple event callback functions through an expressive domain specific language (DSL). The DSL is highly extensible and provides several attachment points to develop new DSL keywords as required. NUClear has been successfully applied in several projects for robotics and virtual reality, and also in the NUgus Humanoid Platform [2]. A detailed description of NUClear is given in [5], which also provides a comparison with the Robot Operating System (ROS)[1] in terms of features and communication performance.

3.2 Gazebo Simulator

Gazebo is an open-source, physics-based robotics simulator. It is very popular with the simulation-league of RoboCup, being utilised by all teams to run their robots in simulations. Considerable support and information are available online to assist the integration of any ROS compliant robotic system with the Gazebo simulator. However, this is not the case for proprietary software, such as NUClear. Since Gazebo has strong connections with RoboCup, and resources such as 3D models and data for the igus platform are readily available, it would be a logical decision to choose Gazebo to integrate with the team's software architecture and NUClear. Integrating ROS into the NUbots' software architecture has been opposed due to the time required to re-factor the entire code, and also because NUClear is arguably more flexible than ROS, as described in [5].

Gazebo depends on several packages but the most significant for this work is the Ignition Transport library[2], which provides Gazebo with an intra- and inter-process co-messaging communication protocol for robot simulation (similar to NUClear). It is an open source communication library that allows sharing data between nodes that could be running within the same process in the same machine or in machines located on the same network.

[1] http://www.ros.org/.
[2] http://ignitionrobotics.org.

3.3 Communication Middleware

This section provides an overview of the communications configuration for this study. An online repository[3] has been created, where developers can download all necessary files and use them. The NUbots codebase repository[4] contains all the required source code for setting up a virtual machine (VM) that runs the NUbots software system under the NUClear framework. A diagram of the configuration for communications between Gazebo and the VM is shown in Fig. 1 and described below.

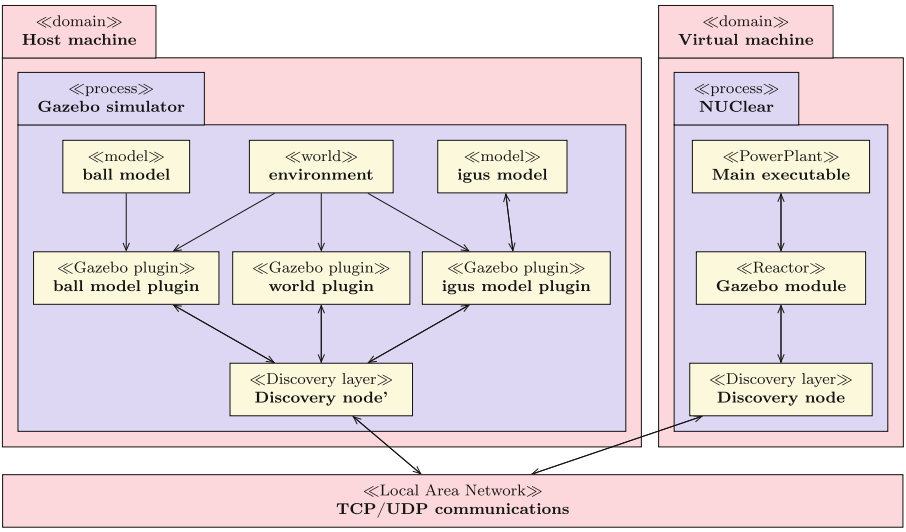

Fig. 1. The configuration for communications between the NUbots software system and Gazebo. On the left, we have the host machine, which runs the Gazebo simulator. On the right, we have a virtual machine that runs the NUbots software system. Two communication nodes (the Discovery nodes) were created using the Ignition Transport library – one for each module. Those nodes manage all the communication through the network and operate in a two-way, publish-subscribe fashion. In each communication cycle, the NUbots software in the virtual machine sends data to the Gazebo simulator about where to move each of the 20 motors, and the simulator sends back data about the position of each limb, as well as gyroscope and accelerometer data. The communication is in real time, via a common TCP/UDP communication layer.

The main contribution in terms of communication infrastructure was the creation of two communication modules. The first one sits within the Gazebo simulator in the host machine and is named *Discovery node'*. The goal of this module is to receive servo commands from the NUbots software that are then

[3] http://github.com/NUbots/Gazebo.
[4] http://github.com/NUbots/NUbots.

applied to the robot model in the simulation. In addition, it sends the position of each joint of the robot model, as well as gyro and accelerometer data back to the Nubots software, so it can be used in the walk engine. The second communication module sits within the NUbots software side (namely *Discovery node*). It manages the publishing of the data that its Gazebo counterpart subscribes to, and subscribes to the data published by the Gazebo's *Discovery node'*. The two modules communicate via a common TCP/UDP communication layer.

Integrating Gazebo with the NUClear framework required several iterations, until a configuration that was able to compile using the NUbots' toolchain was found. Several hurdles had to be crossed in that process, from compilation issues due to package conflicts (which required Gazebo to be built from source), to synchronization problems between the two environments (which was solved with the implementation of a custom clock function). For a thorough analysis of all the issues that occurred during the integration process and their corresponding solutions, we refer the reader to reference [10]. This reference will be valuable to developers who want to use Gazebo with non-ROS compliant applications.

3.4 Multi-Objective Genetic Algorithm (MOGA)

As mentioned before, this work focused on two optimization tasks: (a) the kick script parameters and (b) the walk engine parameters for forward walking. The Multi-Objective Genetic Algorithm (MOGA) implemented uses the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [4]. NSGA-II has several advantages over other well-known MOEAs, such as the Strength Pareto Evolutionary Algorithm (SPEA) [11]. Among those advantages, we cite the concept of minimal distance between sets of solutions, which results in less fitness evaluations until the population converges, and most notably, maintains a strong population diversity throughout the evolutionary process. Now, we describe how MOGA was used in each of the optimization problems.

3.5 Optimizing the Kick Script

In order to optimize the kick script parameters using MOGA, firstly a suitable representation for a solution must be defined. The kick script is defined by a sequence of target positions for each motor (i.e. script frames). In out implementation, the first and last frames are fixed, and the MOGA can change the values for the intermediate frames, only. A valid solution contains 80 genes (20 motors and 4 intermediate poses), encoded as integers in the interval [0, 1024].

Objective Function. Two objective functions were chosen to evaluate the quality of the kick. Obj_1 was designed to measure the robot's stability. By using the IMU sensor in the torso, the accelerometer data is recorded during a kick and a maximum field plane sway $||\overrightarrow{fps}_{max}||$ value is calculated. The field plane sway $||\overrightarrow{fps}||$ is the 2D vector magnitude of the x and y components of the accelerometer

sensor, calculated as:

$$||\overrightarrow{fps}|| = \sqrt{\overrightarrow{x}^2_{accel} + \overrightarrow{y}^2_{accel}} \quad \left(\mathrm{ms}^{-2}\right) \tag{1}$$

$||\overrightarrow{fps}||$ is calculated each time a new sensor message arrives, which is 90 times per second, and Obj_1 is to minimize $||\overrightarrow{fps}_{max}||$.

Obj_2 is the maximization of the ball distance from the kick location. Since a minimization function is required by the MOGA framework, Obj_2 is defined as:

$$Obj_2 = min\left(\frac{1}{\text{ball distance}}\right) \quad \left(\mathrm{m}^{-1}\right) \tag{2}$$

3.6 Optimizing Walk Engine Parameters

This optimization involved the tuning of 46 parameters that describe how the walk engine generates waypoints for the joints during a walk cycle [9].

The representation of a solution consisted of 46 real values, and each of them was given minimum and maximum possible values, set at 50% of the original value in each direction. For example, if a parameter is originally set at 0.4, the minimum and maximum values then become 0.2 and 0.6, respectively. Notice that the original values correspond to the current hand-tuned walk engine, which already produces a relatively stable and fast walk. Thus, we expect the parameters of the optimized walk to be relatively close.

The two objective functions for walk evaluation were the minimization of the time required to walk a distance of 2.5 m, starting from a standing pose; and the minimization of the maximum 3D vector magnitude of the three components of the accelerometer, $||\overrightarrow{accel}_{max}||$, where:

$$||\overrightarrow{accel}|| = \sqrt{\overrightarrow{x}^2_{accel} + \overrightarrow{y}^2_{accel} + \overrightarrow{z}^2_{accel}} \quad \left(\mathrm{ms}^{-2}\right) \tag{3}$$

4 Results

4.1 Results for the Kick Script Optimization

The MOGA framework ran for 300 generations with a population size of 40 individuals. Additional parameters are listed in Table 1 and the results are shown in Fig. 2. The parameters were determined after a series of exploratory tests and empirical observation of the evolution of the population of solutions.

In our tests, we opted for a high crossover probability, aiming at a fast evolution of the population, and a low mutation probability, since diversity would be handled by the minimum distance between solutions, which is part of NSGA-II. In Fig. 2, we highlight four solutions. Ind_O is the original, hand-tuned kick script (distance of 0.6 m; acceleration of 35.8 ms^{-2}) and becomes dominated already by generation 10. After 300 generations, several trade-off solutions were found, with a good concentration around the 3.3 m distance and instantaneous acceleration

Table 1. NSGA-II parameters used in the kick script and walk engine optimizations. For detailed information about those parameters, please refer to [4].

Test	popSize	gens	mutProb	crossProb	etaC	etaM
Kick script	40	300	0.025	0.8	15	40
Walk engine	40	50	0.025	0.8	13	20

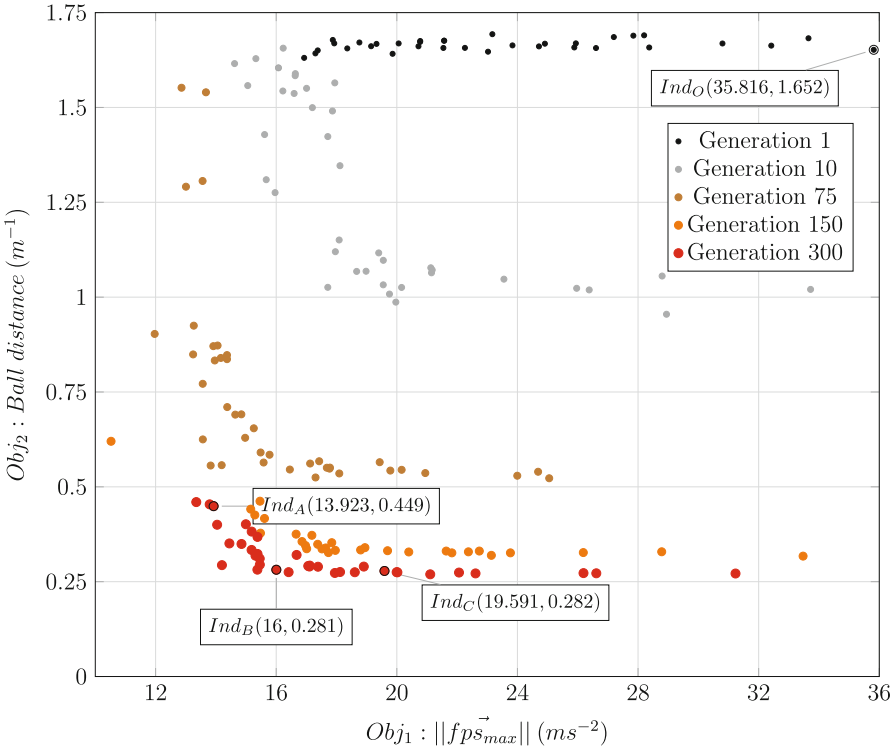

Fig. 2. Populations of solutions found during the evolutionary process for the optimization of the kick script on a simulated grass surface. Notice how the population converges to high quality solutions in 300 generations, with lower torso sway values and higher ball distances (in Obj_2 the values are inverted, so lower is better). Four individual solutions are highlighted for illustrative purposes.

of $15\,\mathrm{ms}^{-2}$. In that group of solutions, we highlighted three, with Ind_B offering a good trade-off between the two objectives. That solution has a maximum torso acceleration of $16.0\,\mathrm{ms}^{-2}$ and a resulting ball distance of 3.5 m. That represents an improvement of almost 6-fold for the distance travelled by the ball, while the maximum lateral acceleration was reduced to less than half of the original value.

Porting the Kick Script to the NUgus Robot: After completing the kick script optimization, we tested three candidate solutions on the robot itself, namely Ind_A, Ind_B and Ind_C. This test was fundamental to determine if the simulation settings were a correct representation of the real world. The so-called *reality gap* between simulation and real world is a major obstacle for the actual use of simulation results in real robotic platforms. For more information about this topic, we refer the reader to a recent survey [7]. Among the three solutions tested, the only one that resulted on the robot still standing at the end of the run was Ind_A. The other two led to the robot falling over at the end of the kick. The most probable reasons for that result might be the incorrect simulation of the friction between the studs on the robot's feet and the artificial grass surface; as well as the internal behaviour of the individual servos not being accurately simulated. This topic will require additional investigation in the future.

Figure 3 shows the time-lapse sequences for the kicks represented by Ind_A (top row) and Ind_B (bottom row), tested on the real robot. The top sequence shows the ball travelling a bit over 2 m from the middle of the field towards the goal (which is very similar to the simulated result), and the robot still standing on the last frame. The bottom sequence shows the ball travelling all the way to the goal (i.e. more than 3 m) but the robot falls over at the end.

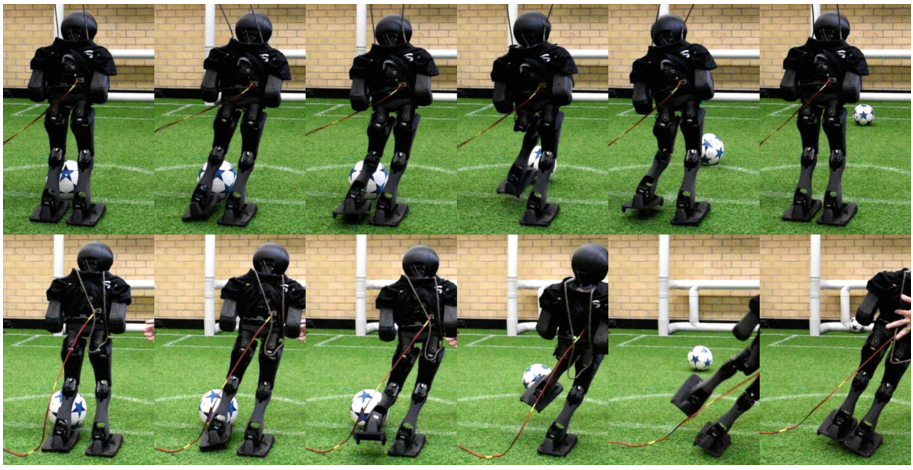

Fig. 3. Time-lapse of two of the kick scripts highlighted in Fig. 2 – Ind_A (top) and Ind_B (bottom) – running on the real NUgus robot. In both cases, the distance travelled by the ball is very similar to the simulated results. However, only for solution Ind_A the robot still stands at the end, while Ind_B leads to the robot falling over – in the simulation, both scripts resulted in the robot standing at the end.

4.2 Results for Walk Engine

The framework was set up with the input parameters listed in Table 1 and was run for 50 generations. The low number of generations was chosen due to time constraints, as each evaluation of the walk required ≈1 min of simulation. The results are shown in Fig. 4. The evolutionary process was again successful. The time required for the robot to walk 2.5 m dropped from 32.5 s to 21.5 s (see solution Ind_A in Fig. 4, compared to the original, hand-tuned solution Ind_O), i.e. an increase of 50% in forward speed. The maximum torso acceleration was reduced from 70.7 ms^{-2} to 43.2 ms^{-2}.

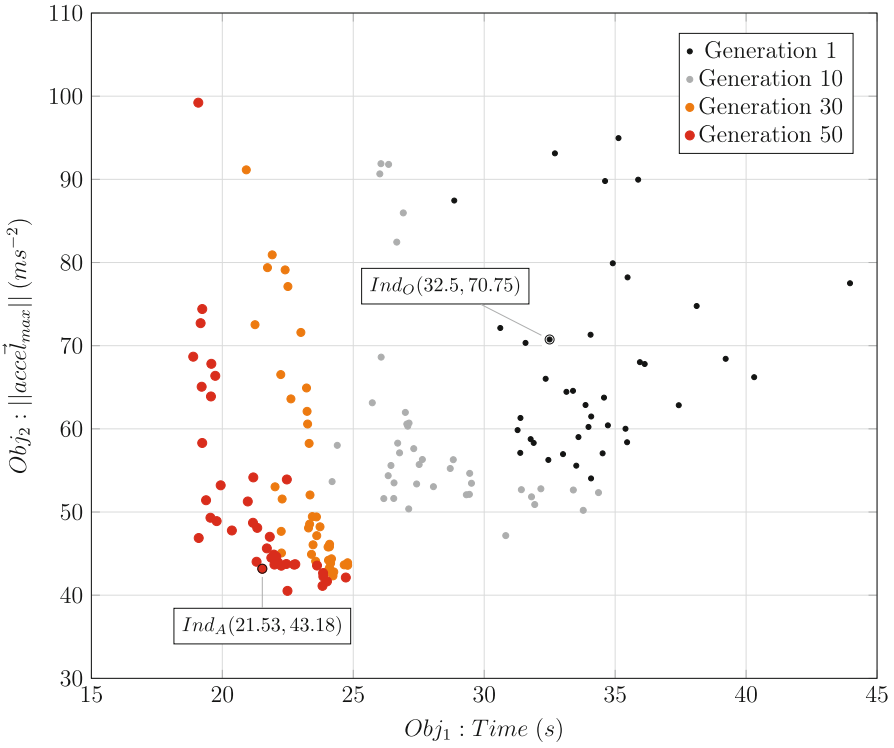

Fig. 4. Evolution of the populations during the walk engine parameters optimization. Notice how the population converges to high quality solutions in 50 generations, with lower times required to walk the 2.5 m distance, and smaller torso sway values. Two solutions are highlighted for illustrative purposes.

5 Conclusion

This work presented a multi-objective genetic algorithm approach for the optimization of two robot movements: kick and forward walk. Moreover, it also

described how to setup a communication bus between the Gazebo simulator and a non-ROS compliant application. The results of the simulation were very good, with the optimized kick making the ball travel 6 times farther. For the walk simulation the results were also very good, with forward speed increasing by 50%. In both cases, the stability of the robot, represented by the maximum acceleration of the torso section, was also reduced, compared to the original, hand-tuned values. Future work will concentrate on bridging the reality gap between the results in simulation and the real robot. At this stage, kick scripts can be directly ported to the robotic platform, but the reality gap remains an issue for simulated kicks with a travelled distance greater than 2.5 m. The walk engine parameters will also require additional testing on the NUgus platform before porting is successfully achieved.

References

1. Allgeuer, P., Farazi, H., Schreiber, M., Behnke, S.: Child-sized 3d printed IGUS humanoid open platform. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids), pp. 1–8 (2015)
2. Biddulph, A., Houliston, T., Mendes, A., Chalup, S.K.: Comparing computing platforms for deep learning on a humanoid robot. In: Cheng, L., Leung, A.C.S., Ozawa, S. (eds.) ICONIP 2018. LNCS, vol. 11307, pp. 120–131. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04239-4_11
3. Chalup, S., Murch, C., Quinlan, M.: Machine learning with AIBO robots in the four-legged league of robocup. IEEE Trans. Syst. Man Cybernet. Part C **37**(3), 297–310 (2007)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
5. Houliston, T., et al.: Nuclear: a loosely coupled software architecture for humanoid robot systems. Front. Robot. AI **3**, 20 (2016)
6. Juang, C.F., Yeh, Y.T.: Multiobjective evolution of biped robot gaits using advanced continuous ant-colony optimized recurrent neural networks. IEEE Trans. Cybern. **48**(6), 1910–1922 (2017)
7. Mouret, J.B., Chatzilygeroudis, K.: 20 years of reality gap: a few thoughts about simulators in evolutionary robotics. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1121–1124. GECCO 2017. ACM, New York (2017). https://doi.org/10.1145/3067695.3082052
8. Nygaard, T., Torresen, J., Glette, K.: Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform. In: IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–8 (2016)
9. Yi, S., Hong, D., Lee, D.: A hybrid walk controller for resource-constrained humanoid robots. In: 13th IEEE-RAS International Conference on Humanoid Robots, pp. 88–93 (2013)
10. Zahn, B.: Optimisation of a walk engine for a humanoid robot. Technical report, School of Electrical Engineering and Computing, The University of Newcastle, Australia (2018). http://github.com/NUbots/Gazebo
11. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm. Technical report, Department of Electrical Engineering, Swiss Federal Institute of Technology, Zurich, Switzerland (2001). https://www.research-collection.ethz.ch/handle/20.500.11850/145755