Praktikum: Virtual Neurorobotic in the Human Brain Project

Marie Bommersheim, Rafael Kübler, and Simon Reinkemeier
FZI Forschungszentrum Informatik
Karlsruhe

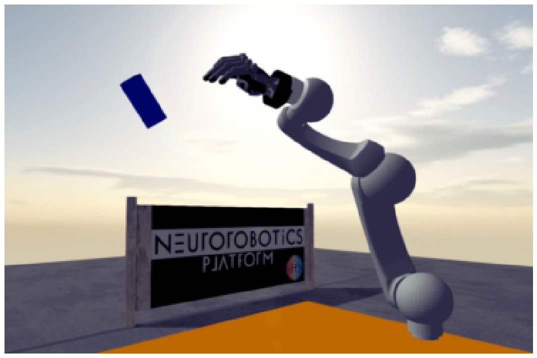


Figure 1. The robot arm throwing the cylinder away from the table.

Abstract

1. Introduction

The aim of our Praktikum is to let the robot arm learn how to throw a cylinder as far away from the table as possible. The robot arm and the cylinder are shown in figure 1.

Sections 2 and 3 describe algorithms for solving optimization problems in general. Sections 4, 5, and 6 describe our approaches for solving the Praktikum challenge.

2. Evolutionary Algorithms

Evolutionary algorithms are population-based metaheuristics [1]. Metaheuristics are methods for solving optimization problems by iteratively improving candidate solutions in relation to a given quality measure. Population-based metaheuristics use populations of multiple solutions and iteratively change populations to find a good solution. An evolutionary algorithm is based on a population P of possible solutions $e_i \in P$ for the optimization problem or heuristic to be solved. A solution is also referred to as an individual in the context of population-based metaheuristics.

In general, evolutionary algorithms consist of the following steps [1]:

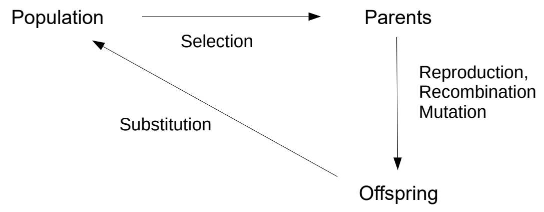


Figure 2. Evolutionary algorithms, adapted from [1]

1. Initialization: The initial population of individuals is generated randomly.

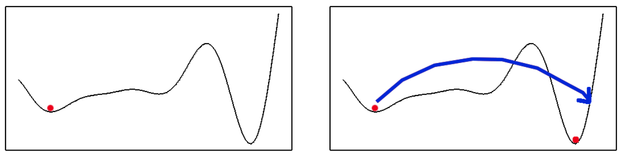
2. Evolutionary process: The individuals are evaluated by calculating their fitness. Therefore a fitness function is needed. According to their fitness, some of the individuals are selected to be the parents of the following generation. Then a process with reproduction, recombination, and mutation starts - similar to biological processes. The parents are being recombinated and the resulting individuals are mutated to form the offspring of the current generation. These steps are being repeated until a new population of individuals is generated. The parents are also added to the new population. This new population substitutes the old population and the process is repeated until an abort criterion is fulfilled (see figure 2).

3. The best individual is selected.

3. Monte Carlo Algorithms

Monte Carlo algorithms are a class of computational algorithms [2]. They rely on repeated random sampling and provide generally aproximate solutions. Monte Carlo algorithms are used in cases where analytical or numerical solutions do not exist or are too difficult to implement. In general, Monte Carlo algorithms consist of the following steps [2]:

1. Determine the statistical properties of possible inputs



(a) A numerical solution found a local (not the global) minimum.

(b) Monte Carlo permits a random exit from local minimum.

Figure 3. Monte Carlo algorithms are a possible method for solving optimization problems [2].

2. Generate many sets of possible inputs which follows the above properties
3. Perform a deterministic calculation with these sets
4. Analyze statistically the results

Optimization problems are a possible application of Monte Carlo algorithms. Numerical solutions to optimization problems have the risk of getting stuck in local optima. The Monte Carlo approach can alleviate the problem by permitting random exit from the local optimum and find another, hopefully better optimum (see figure 3) [2].

4. First Approach

The first approach is to let the robot arm hit the cylinder away from the table. Therefore, the movement is divided into three phases: reset, prepare, and hit. The phases are implemented as states using the state machine editor, the movements are hard-coded.

- Reset: Moves the arm in an upright position and puts the cylinder on the top right corner of the table (as seen from the robot arm).
- Prepare: Moves the arm in an horizontal position pointing to 10 o'clock.
- Hit: With a high acceleration, the arm moves towards the cylinder and knocks it off the table.

5. Second Approach

The second approach uses grasp and throw phases to throw the cylinder away from the table. The reset phase is taken from the first approach, the prepare phase from the first approach is adapted, the hit phase is replaced by the grasp and throw phases.

- Reset: Moves the arm in an upright position and puts the cylinder on the top right corner of the table (as seen from the robot arm).
- Prepare: Moves the arm next to the cylinder (on the left as seen from the robot arm).
- Grasp: Moves the fingers of the robot hand around the cylinder in order to hold it tight.
- Throw: With high acceleration, the arm moves backwards beyond the vertical position and the hand opens to throw the cylinder away.

These movements are also hard-coded.

6. Third Approach

This section describes two different ideas to optimize the throw movement. In both approaches the aim is to let the robot learn which movement is best to throw the cylinder as far away from the table as possible.

6.1. Evolutionary Approach

This approach uses the phases from the second approach. The reset, prepare, and grasp are still hard-coded as in the second approach. The throw phase is trained using an evolutionary algorithm.

6.2. Monte Carlo Approach

This approach also uses the phases from the second approach. The reset, prepare, and grasp are still hard-coded as in the second approach. The throw phase is trained using a Monte Carlo algorithm.

References

[1] P. Flick. Evolutionäre Algorithmen. `http://parco.iti.kit.edu/henningm/Seminar-AT/seminar-arbeiten/Flick_final.pdf`, accessed 03.02.2019, 2012.

[2] S. Paltani. Monte carlo methods. `https://www.unige.ch/sciences/astro/files/2713/8971/4086/3_Paltani_MonteCarlo.pdf`, accessed 03.02.2019, 2010.