Performance of Cooperative Coevolution in Different Scenarios with Heterogeneous Robot Teams

Katharina Spitzley
katharina.spitzley@student.uni-luebeck.de
Seminars CPS and BIR (SS 2022)
Service Robotics Group
Institute of Computer Engineering, University of Lübeck

June 21, 2022

Abstract

Heterogeneous robot teams have the potential to solve a great variety of tasks. Cooperative Coevolutionary algorithms are used to evolve the behaviour of robots and their abilities to cooperate with each other. The NeuroEvolution of Augmenting Topologies algorithm used here is able to evolve the controllers of several robots simultaneously with more flexibility than other evolutionary algorithms. In this paper, two morphologically heterogeneous robots have to solve a token collection task in different scenarios. A strong aerial robot has to lead a ground robot with fewer sensor capabilities to collect the token. Their behaviour and learning speed is analyzed in variants with increasing difficulty. In contrast to a similar experiment from Gomes et al. [2016] the robots in this paper do not learn to cooperate. Instead the ground robot learns a strategy to collect some tokens independent of the aerial robot.

1 Introduction

Evolutionary algorithms are often used in complex environments where the behaviour of the robots is difficult to implement manually. Often the robots are controlled by a neural network, which is evolved with these algorithms. The neural network is mutated and recombined in each generation and only the best regarding a specified fitness function survive. Cooperative coevolutionary algorithms (CCEAs) are used when multiple robots have to be trained in the same setting and need to communicate with each other to solve their task. This can be done with morphologically homogeneous robots (Nitschke et al. [2012] and Potter et al. [2001]), where all robots have the same abilities, or with morphologically heterogeneous robots like in this paper. This kind of heterogeneity results in the ability to solve a much greater variety of tasks, which could not be achieved by robots of the same type. Synchronized Learning [Uchibe et al., 1998] is a key element of the evolution of such populations. That means, the

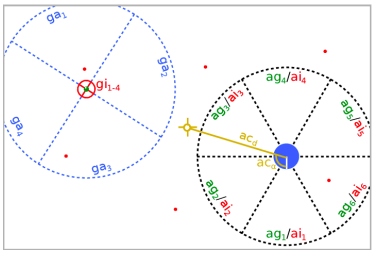


Figure 1: Illustration of the robots' sensors. See Tab. 1 for the sensors description. (Image from Gomes et al. [2016]).

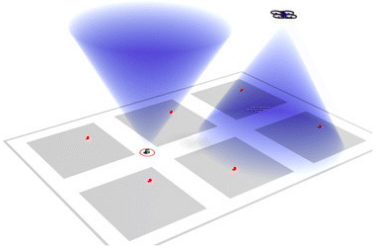


Figure 2: Simulation environment with both robots, tokens (red spheres) and areas (grey zones). (Image from Gomes et al. [2016]).

controllers of all individuals learn simultaneously and dependent on the other populations.
In this paper, two very different robots (a ground and an aerial one) (see Sec. 3.1) have to solve a collection task. The paper analyzes the influence of different simulation environments to the performance of the robots. The influence of random and fixed initial positions, random and fixed token positions, and a bounded and open map is part of this research.

2 Related work

This paper is based on the work from Gomes et al. [2016]. They study how different evolutionary algorithms perform in a collection task of two morphologically heterogeneous robots in variants of different complexity. A weak ground robot, which is able to collect token but has a small vision range, and an aerial robot with a high vision range are used. Their research shows that CCEAs can successfully evolve controllers for these robots but also have difficulties with increasing complexity of the task. They successfully use incremental evolution and novelty-driven evolution to improve the effectiveness of the CCEA.
Like Gomes et al. [2016], this paper uses the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [Stanley and Miikkulainen, 2002] for the evolution of the robot controllers. In addition to the weights of the neural network this algorithms develops the amount of nodes and connections instead of having a fixed architecture. Every neural network consists of a list of node genes and connection genes. In the beginning, there are only the sensor inputs and outputs as nodes and during the evolution more nodes are added. One notable feature about this algorithm is the ability to mutate and recombine genomes with different structure (amount of nodes and connections).

3 Simulation environment

This paper is about a token collection task with two robots with different morphologies and properties (see Sec. 3.1). As in the work from Gomes et al. [2016] the robots have to work together to collect the tokens. The aerial robot has a higher sensor range and should learn to lead the ground robot, which is able to

Ground robot	Aerial robot
Sensory inputs	
gi_{1-4}: 4 binary inputs indicate the presence of token in the respective sector (range = 14cm)	ai_{1-4}: 6 real-valued inputs that give the distance to the closest token in the respective sector (range = 114cm)
ga_{1-4}: 4 binary inputs indicate the presence of the aerial robot in the respective sector (range = 114cm)	ag_{1-4}: 6 real-valued inputs that give the distance to the ground robot in the respective sector (range = 114cm)
	$ac_{d,a}$: 2 inputs that give the distance and relative angle to the centre of the arena
Actuators	
Linear speed (15 cm/s)	Front-back thrust (100 cm/s)
	Left-right thrust (100 cm/s)
Turning speed (180°/s)	Yaw rotation (90°/s)

Table 1: Configuration of the sensors and actuators of both robots (similar to Gomes et al. [2016]).

collect the token but has fewer sensor capabilities. The ground robot respectively has to learn to follow the aerial robot.
At the beginning of each simulation the robots are placed in a 550 × 350 cm^2 map. The starting position is (40,40) with a rotation of $\frac{\pi}{4}$ for variants with fixed initial positions and in a random corner of the map for variants with random initial positions. A corner position means 40 cm away in both directions from the wall and facing in a random direction. The aerial robot starts always at the same position as the ground robot. There are six tokens placed on the map, each one in an area of 150 × 150 cm^2 (see Fig. 2). Dependent on the task variant the tokens are either placed at the centre or a random position inside the area. A simulation ends when all six token are collected or after 400 time steps have been executed. Two time steps are equal to one second.

3.1 Robots

In the simulation two robots with different abilities exist, a ground robot with only a few sensor capabilities and an aerial robot. They can not communicate explicitly but have sensors which can detect the other robot in a specified sector. One sector is covering the front, one the back, and one or two each side of the robot (see Fig. 1). The ground robot only gets a feedback in which sector the aerial robot is detected,while the aerial robot also knows about the distance. The same behaviour is shown when detecting token (see Tab. 1). Like in the task variant Fix-Tog from Gomes et al. [2016] the aerial robot is only able to move at one altitude and with a fixed sensor range. The ground robot collects the token by driving over them (collection range = 4 cm). Both robots are controlled by their specific neural network which will be evolved. The neural network takes normalized sensor readings as input and outputs the commands for moving the robots. The robots are able to move freely in the map, but their position is held inside the boundaries of the map. There are no disadvantages if they touch the boundary.

acronym	*fiftc*	*riftc*	*firtc*	*rirtc*	*fifto*	*rifto*
inital position	fix	random	fix	random	fix	random
token position	fix	fix	random	random	fix	fix
map	closed	closed	closed	closed	open	open

Table 2: Variants of the simulation environment and their parameter settings.

Parameter	Value	Parameter	Value
Population size	100	Probability add/remove node	0.03
Generations	700	Probability add/remove connection	0.05
Elitism	5	Survival threshold	0.4
Max stagnation	15	Compatibility threshold	1.0

Table 3: Parameters of the NEAT-Algorithm used for both robots.

3.2 Task variants

There are six different task variants to differentiate (see Tab. 2). The first variant *fiftc* represents the ground truth. The robots start at a fixed initial position and the token have fixed positions, too. Each of the next variants has a higher difficulty level. In the second variant *riftc* the ground robot starts in a random corner of the map, the aerial robot above it, both facing a random direction. In the variant *firtc* the robots start at fixed positions again but the token are randomly distributed in their areas. Variant *rirtc* combines both and simulates random initial positions as well as random token positions. At last, the boundaries of the map are removed. The variant *fifto* has again fixed initial and token positions. Variant *rifto* has random initial positions but fixed token positions.

3.3 Evolutionary setup

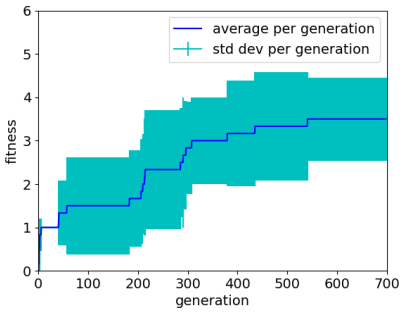
There are two populations. One contains the ground robots and the other the aerial robots. Each population evolves the controllers of their robots. The NEAT-Algorithm (see Sec. 2) evolves a neural network for every individual. Because of the different characteristics, the size of the ground and aerial robot networks differ. The ground robot has 8 sensors and 2 actuators (see Tab. 1) so the neural network of the ground robots has 8 input and 2 output nodes. The aerial robot has more sensors and actuators so the network has 14 input and 3 output nodes. As intended by the NEAT-Algorithm [Stanley and Miikkulainen, 2002], the networks have no hidden neurons in the beginning but evolve more nodes and connections over time. Except for the amount of input and output nodes of the genomes, all parameters are the same for both populations (see Tab. 3).
Each generation is split into two phases. First the ground robots are evaluated with the best of the aerial robots from the previous generation. After that the aerial robots are evaluated with the best of the ground robots.
Pairing each individual from one population with the best one of the other population is a common approach in coevolutionary algorithms (Nitschke et al. [2012], Potter et al. [2001]) because it has a good cost-performance ratio as it does not need to evaluate all individuals from one population with each from

the other [Gomes et al., 2017].
The best individual is identified through the best fitness in the previous generation. In the case of the first generation a random individual is chosen. Each pair evaluates the fitness ten times in independent simulations. The fitness is calculated by the amount of collected token and is only granted to the individual of the population which is currently evaluated, not to both robots of the pair. The maximum fitness that can be achieved is 6.0 (all six token are collected in every independent simulation). The best team of each phase is re-evaluated 25 times again and only these results go into the plots in Sec. 4.
The simulation, evolution and evaluation is made with python3. The implementation of the NEAT-Algorithm is based on the NEAT-Python library [1]. The source code of all experiments including the parameters and scripts used for the evaluation are available under `https://github.com/k31415/Cooperative_Coevolution`.

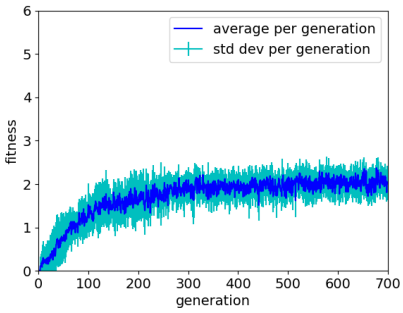
4 Results

For each variant the evolution is executed six times independently. The plots in Fig. 3 show the mean fitness of the best pairs of each generation averaged over all six runs. Additionally, the best fitness value from the results of the best pairs are given. It is observable that the *fiftc* variant with a bounded map and no random initial or token position performs best. But one can see that even the robots in this variant are not able to collect all token. The *fifto* variant with an open map and the *riftc* variant with a bounded map but random initial positions have similar best fitness values. The last three variants all have an average value of lower than one, which means that in every of the six independent simulations the robots collect at most one single token. Overall, all plots show, that the robots are learning at least in the first 20 generations but then they often stay at the same fitness level for the remaining generations.
Fig. 4 shows exemplary paths of the robots. For each variant only one path of one run is shown. All plots show clearly that there is no direct communication between the robots. Often the aerial robot does not move at all or moves in one direction until the map boundary is reached. In Fig. 4d the aerial robot moves in large random looking circles.
At first sight the path of the ground robot looks more goal-oriented, but at a closer look this robot only moves in circles and changes direction when being in range of a token or the map boundary. As seen in Fig. 4c where the path of the ground robot nearly touches the token position, the robot often sees the token only for one or two time steps because of its constricted view. That is sufficient for changing its direction but not for collecting the token.
In Fig. 4f the map-unawareness of the ground robot is shown. Only the aerial robot knows the position of the map center and, because this robot does not move, the ground robot does not know in which direction the tokens are placed.
In Fig. 5 the path of the same robot pair of the *riftc*-variant is shown but with different initial positions. The paths and especially the amount of collected tokens differ, but the general behaviour is similar.
In Tab. 4 the velocities of the robots are displayed. These values are collected during the plotting of the path of the best pairs of each run and averaged over all

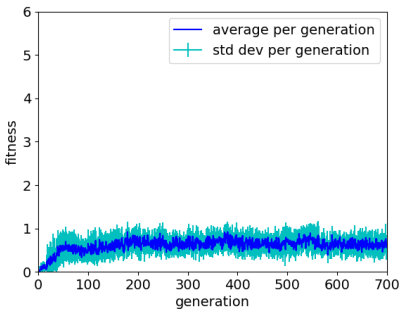
[1] Implementation of NEAT Algorithm in Python3: https://neat-python.readthedocs.io/



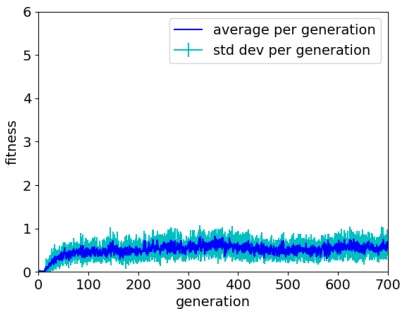
(a) *fiftc*: maximum fitness = 3.5



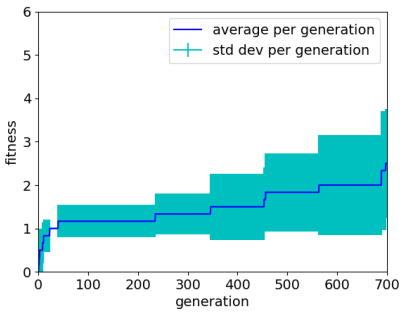
(b) *riftc*: maximum fitness = 2.37



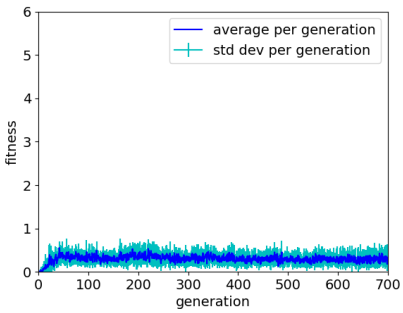
(c) *firtc*: maximum fitness = 0.92



(d) *rirtc*: maximum fitness = 0.82

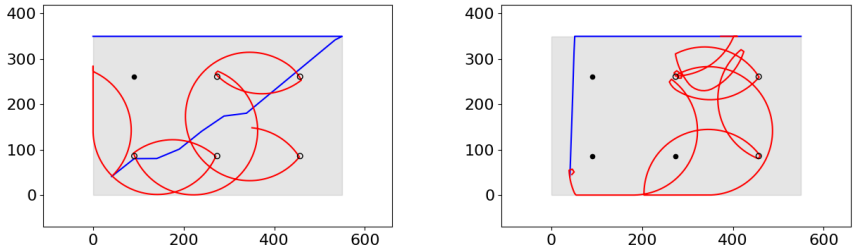


(e) *fifto*: maximum fitness = 2.5



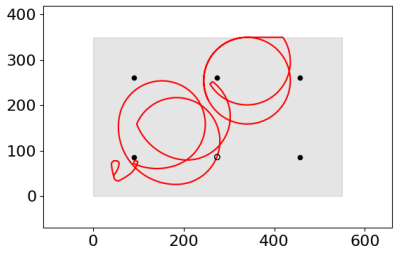
(f) *rifto*: maximum fitness = 0.55

Figure 3: Fitness of robot pairs per generation. Each setting is run six times independently. The results of the best pair of each generation are averaged over all six independent runs.

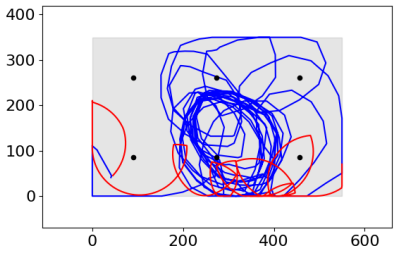


(a) *fiftc*: collected token = 5

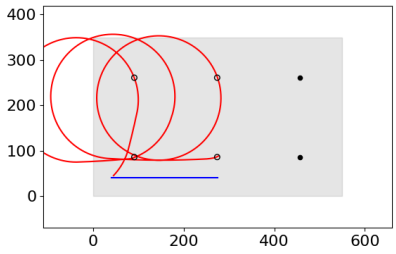
(b) *riftc*: collected token = 3



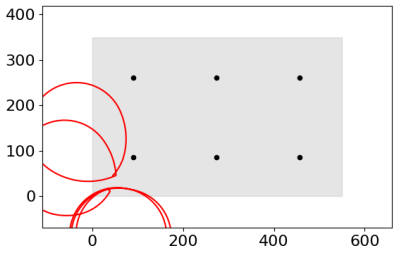
(c) *firtc*: collected token = 1



(d) *rirtc*: collected token = 0

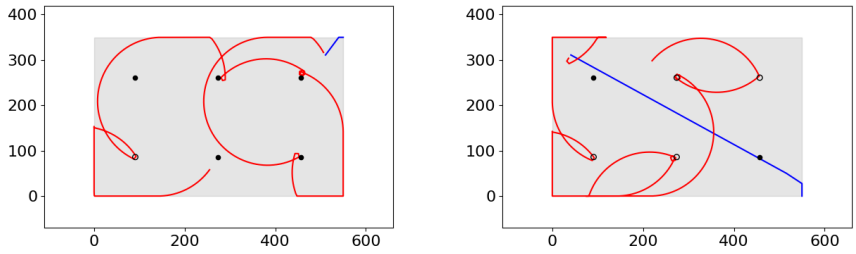


(e) *fifto*: collected token = 4



(f) *rifto*: collected token = 0

Figure 4: Example trajectories of both robots. Path of the ground robot (red line), path of the aerial robot (blue line), uncollected token (black filled circles), collected token (black empty circles), map (grey area).



(a) Robots start in upper right corner, collected token = 1

(b) Robots start in upper left corner, collected token = 4

(c) Robots start in upper right corner, collected token = 4

(d) Robots start in lower right corner, collected token = 0

Figure 5: Same robot pair from *riftc*-variant with different initial positions. Path of the ground robot (red line), path of the aerial robot (blue line), uncollected token (black filled circles), collected token (black empty circles), map (grey area).

	Ground robot				Aerial robot			
variant	average	stddev	max	min	average	stddev	max	min
fiftc	13.44	2.38	15.0	7.5	53.04	52.94	127.08	0.0
riftc	14.9	0.1	15.0	12.36	88.62	41.66	137.18	0.0
firtc	14.46	0.38	15.0	0.1	7.78	17.42	100.0	0.0
rirtc	14.78	0.24	15.0	7.44	63.86	45.48	125.9	0.0
fifto	14.74	0.2	15.0	3.02	40.78	44.46	107.6	0.0
rifto	13.0	4.12	15.0	0.04	0.1	0.24	9.4	0.0

Table 4: Velocities of the robots in [cm/s]. The path velocities of the best results per run were averaged for each variant.

runs of a variant. The ground robot has an average near to the maximum possible velocity of 15 cm/s. Also the standard deviation is small. In contrast the average velocities of the aerial robot are much lower than the possible 140 cm/s and also the standard deviation is very high. This is because in some runs the aerial robot does not move at all and in some it moves very fast.

5 Discussion

In this token collection task even the simplest variant does not achieve satisfying results with only an average of 3.5 out of 6 token collected. In all variants the robots do not communicate or cooperate with each other. The ground robot moves in circles and the aerial moves in a straight line or does not move at all. Both variants without random positions (*fiftc* and *fifto*) get the best results because the robot can learn where the token are placed and which size the circle should have that the robot moves in. The robots of the *riftc*-variant with the random initial position can also learn the size of the circle and collect some tokens. But when the token positions are randomized this behaviour does not work. Because of the restricted view of the ground robot, it can only move randomly and attempts to collect token it encounters by chance. Sometimes the ground robot even moves close to a token but is not able to collect it. Overall I was not able to reproduce the results of Gomes et al. [2016]. They get an average of over five collected tokens within less than 200 generations for a variant similar to *rirtc* but with an open map, which is even more complex than all the settings tested. The parameters used here are as identical as possible with respect to the different implementations. Even a test run of the *rirtc*-variant with 1400 generations performed worse. One possible reason could be that they used other parameters I am not able to implement in the python code. I tested more parameter settings than stated here, such as more generations, different node and connection probabilities and population sizes, but the results did not improve.

6 Conclusion

This paper is based on and attempts to reproduce the results of Gomes et al. [2016] and tries to reproduce their results. Two very different robots, a ground robot with restricted sensor capabilities, but with the ability to collect tokens,

and an aerial robot with more sensor capabilities, have to solve a token collection task by cooperating with each other. A NEAT-algorithm, which is able to evolve the weights of a neural network as well as its architecture, evolves the controllers of the robots. This paper analyzes the learning process and evaluates the behaviour of the two robots in simulation environments with different difficulties. The variants range from having a fixed initial position and fixed token positions to random positions in an closed or open map. The results show that the robots do not learn to cooperate with each other in neither of the variants. The ground robot moves always in large circles which is successful to some degree and the aerial robot does not move most of the time or moves in a straight line in one direction.
Overall the results of Gomes et al. [2016] could not be reproduced, which could be because of the different implementation although similar parameters are used. To solve the problem of lacking cooperation one could change the fitness function from the amount of collected token to a more specialized fitness function such as including the time both robots spend in sensing range to each other, or their average distance to the map center.

References

Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Novelty-Driven Cooperative Coevolution. *Evolutionary Computation*, 25(2):275–307, 06 2017. doi: 10.1162/EVCO_a_00173.

Jorge C. Gomes, Pedro Mariano, and Anders Lyhne Christensen. Challenges in cooperative coevolution of physically heterogeneous robot teams. *Natural Computing*, 18:29–46, 2016. doi: 10.1007/s11047-016-9582-1.

Geoff S. Nitschke, Martijn C. Schut, and Agoston E. Eiben. Evolving behavioral specialization in robot teams to solve a collective construction task. *Swarm Evol. Comput.*, 2:25–38, 2012. doi: 10.1016/j.swevo.2011.08.002.

Mitchell A. Potter, Lisa Meeden, and Alan C. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *IJCAI'01: Proceedings of the 17th international joint conference on Artificial intelligence*, volume 2, 2001. ISBN 1558608125.

Kenneth O. Stanley and Risto Miikkulainen. Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1757–1762 vol.2, 2002. doi: 10.1109/CEC.2002.1004508.

Eiji Uchibe, Masateru Nakamura, and Minoru Asada. Co-evolution for cooperative behavior acquisition in a multiple mobile robot environment. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:425–430 vol.1, 1998. doi: 10.1109/IROS.1998.724656.