Learning a ‘following and chain-building’ behavior using PSO

Alice Concordel[1], Christoph Körner[2] and Etienne Thalmann[3]

***Abstract*— Using PSO, we are optimizing a robot-following and chain-creation behavior for the e-puck platform. The goal is to create a long and stavle chain of robots in an unknown movement. First, in Webots, we develop a Braitenberg type algorithm and implement a learning strategy. Then, we implement a optimized behaviour on the real platform using 4 e-puck robots (1 leader, 3 followers). We will discuss the strengths and weaknesses of this approach and identify possible changes to the heuristics which might improve the transition from the simulated e-puck to the real one.**

I. Introduction

The goal of this project is to implement a Braitenberg-type controller for movement in formation on e-pucks. For this, a PSO algorithm was implemented, using the weights of the Braitenberg-type controller as a search space. The formation consists of 3 follower robots and one leader that moves with a predefined trajectory, unknown at the time of development. The PSO is run on the Webots platform, which is capable of simulating the behaviour of the e-pucks in a realistic way. This makes it possible to run the algorithm and get results much faster than on real e-pucks.

A. Design of the PSO

In order to design a PSO algorithm for a collaborative task, it is necessary to consider three axes: the diversity of the team, the performance evaluation and the solution sharing. It was decided to use a heterogeneous approach with individual performance evaluation and group solution sharing.

The idea is to evaluate different solutions on the three followers (heterogeneous approach) in order to save computation time. For each robot, the previous one acts as leader thus the problem is decomposed into three similar parts. Another option would have been to run the simulations with only one robot following a leader, but the results would have been similar to the used approach. The difference would be, that if following a follower with a controller, which causes a very erratic movement and abrupt acceleration changes, which are propagated and increased over the chain. Then the evaluation would not be quite fair as opposed to that on a robot following the leader with a more reasonable trajectory.

The pool of solutions is shared (group solution) because the situations are similar, with leader-follower couples and thus testing a particle in each position of the chain will give similar results.

Finally, since the performance of each follower is what is aimed,with its own particle and not in the performance of the entire chain it is logical to use an individual performance evaluation.

B. Design of Fitness Function

The idea of the fitness function is to evaluate the performance of a controller in being able to follow a leader and stay in chain formation. The design of the fitness function is of primary importance because it defines towards which solution the PSO will converge [**?**]. Four characteristics were found to be important to evaluate following and chain-building behavior:

- The range d, which is the distance between the follower and the leader, has to be minimized.
- The relative heading of leader and follower $\Delta\phi$. The heading is defined as the angular orientation of the robot in a fixed frame of reference. The relative heading is zero, when both robots are facing the same way. This angle has to be minimized in order to have nice following.
- The bearing θ, which is the angular offset of the leader’s position with respect to the heading of the follower. This quantity has to be minimized so that the follower always stays behind the leader.
- The relative speed between leader and follower Δv can be minimized in order to have a smoother following behavior. This was part of our original design but has later been proven unnecessary.

The fitness function has to be maximized. We define it as follows:

$$F = \frac{1}{A \cdot d + B \cdot \theta + C \cdot \Delta\phi + D \cdot \Delta v} \qquad (1)$$

Where A,B,C,D are coefficients of importance of the different components that are initially set to one and have to be determined.

II. Methodology

A. Implementation on Webots

1) Environment: The Webots environment used to simulate the following and chain building consists of one leader e-puck and three followers on a smooth ground free of walls or any obstacles. They followers are initially placed in line behind the leader with a distance of 10 cm between their centres and facing the same way. A supervisor, which is a special kind of robot with more functions, is used to run the

This work is a project of the course Distributed intelligent systems by Prof. Alcherio Martinoli

A. Concordel is with Faculty of Mechanical Engineering, School of engineering, Ecole Polytechnique fdrale de Lausanne

C. Körner is with the Faculty of Electrical Engineering, Vienna University of Technology

E. Thalmann is with Faculty of Mechanical Engineering, School of engineering, Ecole Polytechnique fdrale de Lausanne

PSO algorithm, track the position of the robots, communicate with them and reset their position.

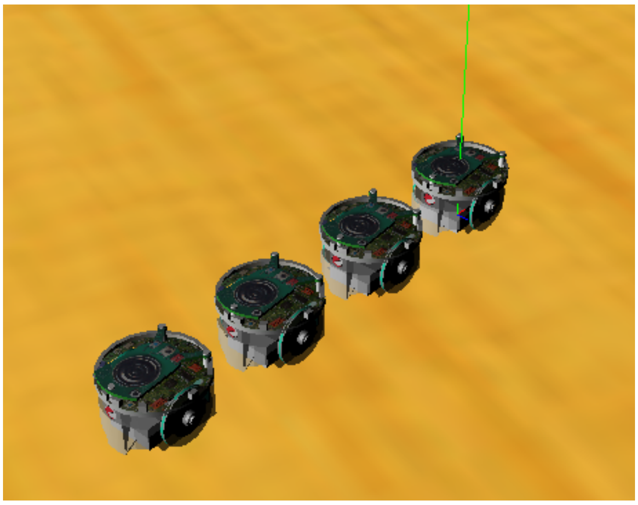

Fig. 1. Initial chain positions of the robots in the Webots environment.

2) Supervisor: The role of the supervisor is to set the initial settings and then to run the PSO. In short, it sends the particles to the followers for their evaluation, gets their fitness back, evolves the swarm and restarts an iteration. After running one full PSO, it uses the best results for the next one and so on until results are satisfying.
The number of particles in the swarm is set higher than the number of followers. Since the approach is heterogeneous, the supervisor evaluates them in batches of three by sending one particle (i.e. one set of weights for their Braitenberg controller) to each follower. Then comes the evaluation period during which the followers evaluate the fitness of the controller. To evaluate the fitness of its run according to equation **??**, the follower needs to know its coordinates and those of the robot it is following. Thus, while the followers are doing their evaluation run, the supervisor is sending the spatial coordinates (translation and rotation) of the respective leader in the follower's coordinate system to each follower. Once the fitness of each particles is evaluated, it is returned to the supervisor and stored. The supervisor resets the followers and the leader in their initial position and replicates the previous procedure with the following batch of particles. Once all the particles of the swarm have been evaluated, they evolve according to the PSO algorithm using the fitness values obtained for each particle. The new swarm obtained is evaluated and evolved in the same way as the previous swarm. The number of these evolution iterations is set beforehand.
When the last iteration is over, the weights of the particle that returned the best fitness during this run of the PSO are tested by sending them to the followers a chosen number of times and measuring the average fitness of the solution. If the results are better than the previous best, the solution is stored in a text file and set as initial weights for the next PSO run. The entire PSO algorithm is run again like the previous one, each particle being initialized with the best weights so far. This higher level of iterations is run as long as needed to get satisfying results. The weights that had the best fitness evaluation can be extracted from the text file and implemented on the real e-pucks.
The supervisor also communicates in a unilateral way with the leader, sending an interruption message every time the positions are reset. The purpose of this is that the leader restarts its trajectory, so as to have always the same trajectory in the case of predefined ones. This was needed at the beginning of the evolutions in order to have all the particles evaluated on a trajectory having the same level of difficulty. This way, weights that are worse than others cannot have a better fitness evaluation just because the trajectory was easier, which would be counter productive.

3) Follower: There are three followers having the same code. The role of a follower is to evaluate the fitness of the particle that is sent by the supervisor. The weights received are used for their Braitenberg-type controller and the fitness is evaluated using **??** before being sent back to the supervisor.

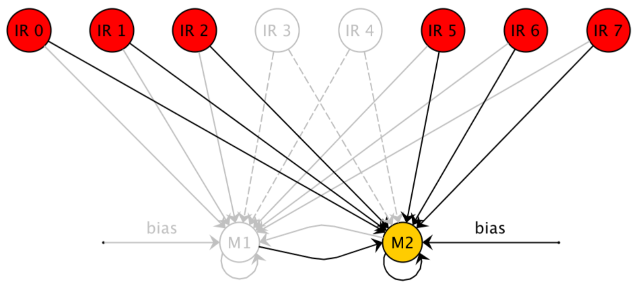

Fig. 2. The simplified neural network (the nodes and links that are greyed are not included on the PSO).

The Braitenberg-type controller is a simple neural network consisting of two neurons corresponding to the wheel motors of the e-puck. As input nodes, the neural network uses the infra-red sensors of the e-puck and a bias. The outputs are the motor commands of the two wheels. The outputs are also cross-coupled and fed back to their neuron. The complexity of the controller is reduced using symmetry. Indeed, the same weights can be used for the bias, cross-coupling and auto-coupling of both nodes and the weights corresponding to the opposite sensors can be used for the neuron corresponding to the opposite wheel. Moreover, the back sensors can be ignored as they do not play a role in a following behavior. Thus, the number of weights of the controller is reduced to nine: six for the infra-red sensors, one for the bias, one for the cross coupling and one for the auto-coupling.This can be seen in figure **??**
The infra-red sensor values measured in Webots were calibrated in order to be close those obtained on the real e-pucks under lighting conditions similar to the testing environment. This can be done by updating the lookup tables in Webots, which then maps distance to sensor values by linear extrapolation between the inserted data. In order to improve the Braitenberg controller, it was decided to

linearise the input by mapping the sensor values back to distance before using them. In order to do this, the inverse function of an exponential regression through the data was used, as seen in figure **??**. This improves the quality of the controller because it avoids the exponential increase of the stimulus when the follower gets close to the robot, which makes the response smoother, and it increases the stimulus with distance instead of decreasing it, which makes following easier.

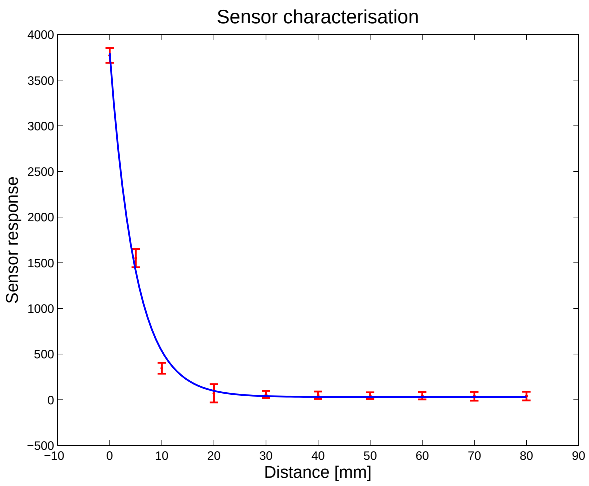

Fig. 3. The measured IR sensor values and the curve fitted to the data.

a) Braitenberg-type controller: The output values computed by the Braitenberg controller are transformed into coherent speed values before being sent to the wheels. This is done by mapping the values between -1 and 1 before multiplying them by the maximum speed that the e-pucks can reach. The threshold upon which the speed is set to maximum speed (or minimum speed by taking the negative of the threshold) is set arbitrarily by looking at the range of values returned by the controller. One last step is done before sending the speed commands to the wheel motors: verifying that the acceleration does not exceed a maximum value. This is done looking at the difference between the new speed and the one of the previous time step. If the difference is larger than the defined maximum acceleration for the e-pucks, the speed corresponding to this maximum value is set.

b) Fitness evaluation: Every time a particle is sent to the followers, its fitness has to be evaluated. To do this, Webots simulates the behavior of the followers with the given controller and initial chain formation during a certain amount of time steps. The components of the fitness function are evaluated at each time step and averaged at the end. These components are calculated using the position information sent by the supervisor. Each follower calculates at each time step the range, bearing and relative heading between the leader and itself using the received coordinates. At the end of the run, these values are averaged and entered into the fitness function (equation **??**). The calculated fitness is sent back to the supervisor and the follower waits until it receives the next particle to start the process again.

4) Leader: The formation performance is strongly dependant on the leader behaviour. It is therefore critical to control this to ensure that the solution obtained is robust. For example, if the leader goes straight forward, then a controller with strong bias and almost no coupling from the sensors will have a good fitness. However, if this controller is then tested with a leader that turns, then the followers will be lost.

Thus it is important to test the controller on a varied trajectory, containing both left and right turns. A random trajectory will be good for this, but will require a longer evaluation time to ensure that a sufficient variety is obtained. For this reason, the first runs of the PSO (to test its performances and tune its parameters) were run with a predefined trajectory, including a soft turn to the right, then the corresponding turn to the left which brought the robot's heading back to its initial value. The benefit is that all of the parameters acting in the fitness function appear in a short time span, making runs faster. This turned out to be very effective for debugging: the trajectory being the same, the consistency of the fitness values returned was much easier to check. This was also a good way to quickly get good weights with pretty short iterations. However, this could not be used to evolve the final solution as it would be limited to similar movements of the leader.

So a random, longer trajectory was preferred. At first, random speeds were assigned to the wheels for given time intervals. However, this will give a rather erratic movement. It was optimized such that the new speed of wheel should be 0.4 times the old speed plus 0.6 times a new random speed value (within the wheel command values interval). This gives the leader a smoother and more realistic trajectory.

B. *PSO characteristics and Optimization*

To get started with the PSO and to understand the behaviour of the robots in Webots as well as the fitness function, we initially started the noiseless PSO with 10 iterations and 30 fitness steps. This way, the overall simulation time was kept short in the beginning. The weights of attraction to personal best and neighbourhood's best solution were set equal. The leader motion was deterministic and consisted in a single curve. Because of this short simulation interval and the leader turning only one way, the first optimization results lead to useless controller weights. The only goal in this first step was to examine the quality of the designed fitness function and to test the design of the symmetric controller weights.

To get useful results we extended the number of fitness evaluation steps per iteration to 65. The trajectory of the leader was still deterministic, but it consisted of two opposite curves and a straight part in between. The velocity of the leader was constant. After evolving the PSO several times with random initial particles and velocities we observed a big variance in the quality of the results. So we tried to improve the quality of the PSO by implementing bounded constraints for the PSO [**?**]. With initially starting an iteration of the optimization with the last best solution we could achieve better results in less number of iterations compared to random initialization of particles. Increasing the number of PSO evolution iterations to 40 extended the overall optimization time, but lead to useful results after each optimization. After the implementation of a function that stored the best controller weights to a file, if they were better than the previous best, we could run the optimization with good initial particles in an infinite loop over the night. Applying the optimization results to the followers in Webots, the robots could follow the leader perfectly under the given (optimal) constraints.

In the next step, we extended the follower model to have a more realistic behaviour. Therefore we added noisy nonlinear sensor values for the proximity sensors of the e-pucks. We configured the PSO to be noise-resistant, which consists in re-evaluating the best performance of a PSO iteration. This results in doubling the number of iterations per optimization step. We also complicated the leaders behaviour, such that the trajectory is random and 180 simulation steps long. We seeded the random number generator with the actual time stamp to get different random values for each optimization but with giving up reproducibility of the solutions. For the overnight simulation, we started one PSO with good initial weights and random velocities and another one with random particles and random velocities. The performance for the optimization with initial particles was nearly twice as good as the one with initial random particles.

The obtained results could not be improved by using a random local neighbourhood and increasing the weight of attraction towards the neighbourhood's best solution.

C. Fitness function Optimization

After the first results from the PSO, we could tune the fitness function (equation **??**) and the different weights of the parameters. First the minimization of the relative speed between leader and follower was dropped from the design, because this would need to compare the changing of the positions over time. To keep the fitness function simple and to not compare differences in positions over time we decided to just stick to minimization of range, relative heading and bearing between leader and follower. With the tuning of these parameters gave to good results of the PSO in the simulation. With the greatest importance on minimizing the range, the following final fitness function was developed.

$$F = \frac{1}{10 \cdot d + 1 \cdot \theta + 6 \cdot \Delta\phi} \tag{2}$$

III. Results

A. Best solution

The best weights in Webots were obtained running a noisy PSO on a random leader trajectory with random acceleration (180 iterations). These weights are $[-24.45, 4.96, 1.69, -0.43, 5.70, 100.01, 56.63, -71.26, 17.65]$, where the first 6 correspond to the proximity sensors, 7 to the bias and 8 and 9 to the recursive connections. Applying these weights to the e-puck immediately gave a good chain following behaviour.

To test the weights obtained, the same Braitenberg controller as the one used in Webots was implemented on the e-pucks. Then the weights that returned the best performance in the PSO were given to the controller. For the leader, a random movement coupled with obstacle avoidance was loaded onto the leader in order to test the solution on a test table without the robot getting stuck or wandering too far.

B. Limitations

The solution found has some drawbacks. Due to the nature of the neural network, the robots do not stop at defined distance from the leader. Instead they move toward it until they touch it and begin to push gently. Just one single follower cannot push hard enough to make the leader move, however, the sum of forces from the chain of three robots can cause the leader to slip forward, thus affecting the leader's trajectory.

To correct this, an additionally rule based algorithm could be implemented, which uses a threshold on the smallest distance measured by all of the sensors and causing it to stop, if it is below the defined distance. This was tested in the webots simulation, but it creates an oscillating jerky movement of the followers, that is caused by the abrupt changes in acceleration and propagated with a delay through the chain of followers.

A solution to remove the oscillations would be an implementation of a hysteresis on the threshold, or a PID controller on the detected distance (in the test $1cm$). A simple weighted PID controller, that minimized the error to the defined distance was tested without success. After correcting the range in the fitness function with the defined distance the PSO evolved a weight near to zero for the PID controller, because the controller was breaking the chain following behaviour, when it tried to keep the defined distance between leader and follower. While setting a higher weight manually to the PID controller we observed, that the distance was kept, but the chain broke easily. By setting a distance smaller than $1cm$ or implementing multiple PID controllers for each sensor, the controller could be used in this application.

IV. Conclusions

V. Acknowledgements

The authors would like to thank Alcherio Martinoli for an exciting and challenging course. Special thanks also to Milos Vasic, for his precious advice and support, as well as all the other teaching assistants of the course for their work throughout the semester.

References

[1] R. Poli, J. Kennedy,T. Blackwell, ”Particle swarm optimization, An overview” in Swarm Intell (2007) 1, Springer Science + Busines Media, 2007, pp 33-57

[2] E.F. Campana, M. Diez, G. Fasano and D. Peri, ”Initial Particles Position for PSO, in Bound Constrained Optimization” in Advances in Swarm Intelligence, *4th International Conference, ICSI 2013; Harbin, China, June 2013; Proceedings, Part I*, Springer-Verlag Berlin Heidelberg, 2013, pp 112-119.