1 Lab 9: PSO for Multi-robot Systems

This laboratory requires the following:

- C development tools (gcc, make, etc.)
- Webots simulation software
- Webots User Guide
- Webots Reference Manual

1.1 Office hours

Additional assistance outside the lab period (office hours) can be requested using the dis-ta@groupes.epfl.ch mailing list.

1.2 Information

In the following text you will find several exercises and questions.

- The notation $S_x$ means that the question can be solved using only additional simulation.
- The notation $Q_x$ means that the question can be answered theoretically, without any simulation; if you decide to write a report, your answers to these questions should be submitted in your report. The length of answers should be approximately two sentences unless otherwise noted.
- The notation $I_x$ means that the problem has to be solved by implementing a piece of code and performing a simulation.
- The notation $B_x$ means that the question is optional and should be answered if you have enough time at your disposal.

The combined total number of points for this laboratory exercise is 100. The laboratory is not graded.

2 Multi-robot PSO for obstacle avoidance

Using PSO on a single robot, it can take a long time to automatically design high-performing controllers. This is because the robot will have to test every member of the swarm (i.e., pool of candidate solutions) one after another for each iteration of the algorithm. For example, doing 100 iterations of PSO with 20 particles with each evaluation taking 1 minute means we need 2000 minutes or over 33 hours to complete!

One way to speed up this process is to use multiple robots. In this way, we can distribute the particles among the robots and evaluate their performance in parallel, thus saving a lot of time. Going back to our previous example, if we have 20 robots each evaluating a different particle, it will only take 1 minute to evaluate 20 particles, and the total adaptation time goes down to about 1.7 hours.

We will now test the shaping of an obstacle avoidance behavior using multi-robot PSO. In the following questions, you will be asked to run simulations that will take about 2-3 hours of simulated time, and several minutes of real time to complete. In order to save time, you may want to look ahead to the next questions to see if there is something which you can work on while you are waiting.
Start by downloading lab09.tar.gz from Moodle and decompressing it:

```
$ tar xvfz lab09.tar.gz
```

This will create a `lab09` folder containing the worlds and controllers for this lab. After loading the Webots worlds, make sure you compile all the relevant controllers.

**S1:** Load the `psoso_` world. Run the world and have a look at the speed-up factor in at the top of the Webots window, while running in fast mode. As in lab 8, the simulation will do 10 separate optimization runs and print the final fitness for each, as well as the average fitness at the end. When the runs are finished, observe the performance of the robots (they will be running the best found controller).

**Q2 (10):** In lab 8 you were using a single robot to test all particles sequentially. This took ~20h of simulated time and the speed-up factor while running in fast mode was around 350. What is the improvement in simulation time? What is the improvement in wall-clock time speed-up, when moving from single- to multi-robot simulations? How do you explain these results?

**Q3 (5):** What kind of fitness values do you get?

**I4 (10):** Open the pso.c file in the `pso_obs_sup` directory and implement the noise-resistant version of the algorithm presented in lecture, which reevaluates the personal best and averages with the previous one. Recompile. (Hint: set NOISY=1; the part of the code you need to change is marked with TODO)

**Q5 (5):** What fitness values do you obtain in this case?

**Q6 (5):** What do you observe in terms of performance increase for the noise resistant PSO when comparing to the single robot scenario in the previous lab? Explain why this happens.

**Q7 (5):** Have a closer look at the supervisor code (in particular the `calc_fitness()` function). How are the particles divided among the robots? Is this a homogeneous or heterogeneous team learning approach?

**Q8 (5):** Explain the differences between private and public policies in solution sharing.

**Q9 (5):** Why is a group performance evaluation, public solution sharing and heterogeneous learning approach not scalable with the increasing number of robots?

### 3 PSO for collaborative tasks

We will now move on to a more complex task that requires cooperation between robots. Collaborative tasks are a greater challenge because it is hard for robots, in case
of fully distributed control and especially with limited explicit information sharing
about their mutual intentions, to determine whether an observed event is the result of
their own actions or the actions of other individuals, a situation characterized by a
canonical credit assignment problem in multi-robot systems.

In this lab, we will use PSO to learn the coordinated motion task, in which robots
must cover as much distance as possible while remaining in each other’s sensor range.

S16: Open the pso_coop.wbt world in Webots and run the simulation; it takes
about 3 hours of simulated time. While the simulation runs, answer the
following questions.

Q11 (10): Have a look at the calc_fitness() function in the supervisor code. What is the
fitness function being optimized?

Q12 (10): Is this an individual or a group fitness evaluation? What hardware would be
required to implement it with real robots?

Q13 (10): If your answer to the previous question was individual, can you suggest a
group function for global evaluation? Or, if it was group, can you suggest an
individual one? (Hint: remember what you are trying to optimize; any
function should involve the distance travelled and the proximity)

Q14 (5): The optimization process should be done. What fitness values do you obtain?

Q15 (5): Observe the behavior of the final solution. Is the performance consistent
across runs? Knowing that the sensor noise is the same as the one used for
the obstacle avoidance task, what do you think is the main cause for the
variations in the evaluations?

I16 (10): Copy over the noise-resistant code from I4 and run the simulation with
NOISY=1. What fitness values do you obtain in this case? What do you
observe in terms of performance difference between noise-resistant and
standard PSO?

4 References

swarm optimization, Swarm Intelligence 3(3), 203–222.

Self-organized coordinated motion in groups of physically connected robots. IEEE
Transactions on Systems, Man and Cybernetics—Part B: Cybernetics, 37(1), 224–
239.