Distributed Intelligent Systems
Lab 10 Tutorial

Wanting Jin
Lab Structure

1. Multi-robot PSO for obstacle avoidance:
   – Same fitness function as single-robot PSO
   – Differences in performance and evaluation time

2. Budget allocation:
   – Comparing standard PSO, PSO-pbest, PSO OCBA

3. Multi-robot PSO for collaborative tasks:
   – Coordinated motion: move as far as possible while staying together
**Code Structure**

**Pso_sup.c**
- **Main()**
  - Initialize world
  - Best = pso()
  - Evaluate best

- **Calc_fitness()**
  - Reposition robots randomly
  - Send candidate solutions to robots
  - Evaluate fitness
  - Return fitness

**Pso.c**
- **Pso()**
  - Initialize swarm
  - For each iteration
    - Move particles
    - Evaluate particles
  - Return best particle

**Obs_con.c**
- **Main()**
  - Initialize robot
  - Receive weights from supervisor
  - Run controller with weights
  - Send sensor data to supervisor
Noise-resistant PSO

• Setting NOISY=1 triggers two changes
  – Half the number of iterations
  – Reevaluate performance for lbest (with flag EVOLVE_AVG)

• You need to implement the behavior for EVOLVE_AVG
  – Modified moving average (MMA) with age as the number of periods.
  – Remember to increase age.
PSO-OCBA

- Optimal allocation of computation budget
- Very effective in the presence of noise
PSO for a collaborative task

- Fitness value to be calculated in the supervisor
- Copy-paste the noise resistant PSO
- More difficult task than obstacle avoidance
- New sources of uncertainties
Notes and Clarifications

• Simulations take longer with complex tasks, set rendering off, read ahead and answer questions while the simulation runs.

• Performance evaluations have a high variance, you may need additional runs to establish clear trends

• Remember to fill the feedback form