Distributed Intelligent Systems
Course Project

1 General Information

Distributed Intelligent Systems involves a 50-hour (per student) course project (this includes reading, implementation, and oral defense of the project). This edition includes four project topics, which will be elaborated on later. Projects will be carried out in groups of three (default) or two students belonging as much as possible to different teaching sections or programs. The teams will be organized based on the preferences expressed by the students during Weeks 6 and 7.

In the lab session of Week 7 (October 31), there will be a kickoff session of 1-hour, during which the details of the project topic and organization will be presented while the material will be made previously available on Moodle. This session serves as the official start of the project period until the end of the semester, when the project presentations, together with further implementation material (e.g., code), will have to be submitted. Additional details about the project presentations will be communicated in a timely fashion. After the kick-off session, the students will be asked to submit their preferences on project topics via Moodle. During the course project period, the last hour of each lab session will be dedicated to project supervision (as mentioned in class, questions about technical understanding and implementation choices will be possible but no debugging requests will be accepted). In the lab session of Week 13 (December 12), we will provide assistance to the course project for three hours. No further office hours will be made available.

The final project presentations will be carried out as a team and will take place during the last week of the semester within your group slot. Your presentations should last at most 9 minutes with equal speaking time among team members and should outline your particular contribution to the team effort. The follow-up Q&A session, which is 6 minutes, will cover the whole project. We expect equal contributions from students in this session. Presentation slides and material will be due ahead of time, namely by Sunday, December 15, 23:59.

2 Deliverables

Your deliverables include two main components:

1. Submit a .ppt file named `DIS_project_presentation_group_X.ppt`, where X is the group number, including:
    a. Problem statement, formulation, and motivation
    b. Clear division of labor and contributions of individuals
    c. A flow chart or architectural diagram or pseudocode of the main algorithm/approach/methodology
    d. Quantitatively and qualitatively analyzed results highlighting main outcomes
    e. An informative video showcasing your solution in action within the simulation environment (Webots)
2. Submit a .zip file named `DIS_project_material_group_X.zip`, where X is the group number, including:
    a. Webots environments and controllers
    b. Any additional code or resources used

3 Grading

As previously mentioned in the syllabus, the course project performance will weigh 50% of the total grade of the course, 30% based on the team's overall performance, and 20% based on the part of the individual performance focusing on the presentation and discussion of your contribution to the course project. Therefore, the project grade (scaled to 100%) will be evaluated using the following criteria:

Quality of the proposed overall solution	40 %	Team
Quantitative performance on distributed metrics or evaluation criteria	20 %	Team
Quality of the presentation	20 %	Ind.
Quality of the answers delivered during the oral discussion	20%	Ind.

4 References

Some useful references can be found below:

- Reference manual for Webots 2022a: https://cyberbotics.com/doc/reference/index?version=R2022a
- User guide for Webots 2022a: https://cyberbotics.com/doc/guide/index?version=R2022a
- User guide for e-puck: https://cyberbotics.com/doc/guide/epuck?version=R2022a
- User guide for all sensors: https://cyberbotics.com/doc/guide/sensors?version=R2022a
- Reference for supervisor: https://cyberbotics.com/doc/reference/supervisor?version=R2022a
- DIS Lecture Notes and Labs 1-6

Topic 1: *Solving Traveling Salesman Problem by Ant Colony Optimization with a team of robots*

1 Introduction

In a large facility populated with obstacles, you are tasked with providing security checks in critical locations using autonomous robots. You have already determined critical patrol points to be visited by the robots in this facility; however, you have to calculate the shortest possible routes between these patrol points. This problem is known as the Traveling Salesman Problem (TSP). It is a classical NP-hard optimization problem in which the objective is to find the shortest possible route that visits a set of cities (patrol points) and returns to the starting point. Ant Colony Optimization (ACO) is a probabilistic optimization technique inspired by the behavior of real ants, which can be used to solve combinatorial problems like the TSP. This project will involve implementing ACO in a simulated Webots environment consisting of e-puck robots and patrol points to solve the aforementioned problem.

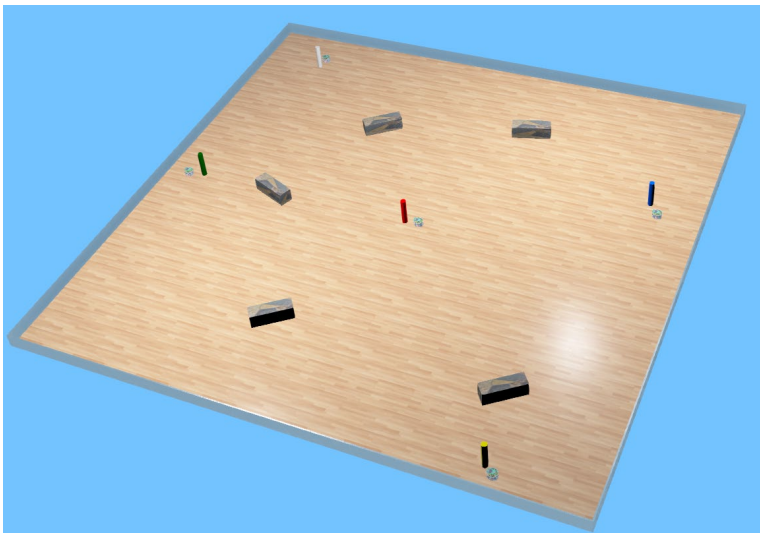

Figure 1. *Provided Webots world with five patrol points, five robots, and obstacles*

2 Project description

The primary objective of the project is to find an optimal solution to the TSP by leveraging ACO for the given Webots world `topic1.wbt`, where five patrol points with distinctive colors and five e-puck robots are already present in the environment. Here, while the patrol points represent nodes in the graph, the robots represent the artificial ants in the ACO algorithm. The environment also includes some obstacles that robots should avoid while reaching the next patrol point. **Note that all patrol point locations with their distinctive colors are assumed to be known by all robots.** The project consists of three main components:

1. *Implementation of robot navigation, localization, and patrol point detection in Webots:* The robots should be able to go from one patrol point to the next without colliding with obstacles and other robots. They should also be able to detect distinct patrol points through their sensors in order to complete the visit. **All sensors on the e-puck robot and GNSS can be used for localization and detection purposes**.

2. *Implementation of Ant Colony Optimization:* The robots should build the shortest route solution step by step and communicate via simulated (virtual) pheromone trails. The pheromone level initialization, tour construction, pheromone update, evaporation and any additional functions should be implemented. This algorithm can be implemented with the help of a supervisor node in a centralized fashion or by using only the robot controllers in a distributed fashion. In the latter case, the robots should use inter-robot communication to share the necessary information between robot controllers without any supervisor node. **Note that any offline calculation of the shortest path is prohibited.**

3. *Analysis of parameter sensitivity, algorithm variants, and environmental factors:* The effect of ACO parameters such as α, β, ρ, and Q, as well as the addition of local search (2-opt) on the ACO's performance, should be investigated. The algorithm's robustness should be validated in a more complicated environment, i.e., with more patrol points, robots, and obstacles.

You are expected to present the following results in your presentation:

- Provide a visualization showing the evolution of the virtual pheromone heat map over the iteration of the ACO and clearly indicate the final TSP solution. You can use any visualization tool in Matlab, Python, etc.
- Quantitative analysis of pheromone dynamics, i.e., ACO parameters and 2-opt local search strategy on ACO's accuracy (path length) and computational efficiency (total computation time for the same number of iterations).
- The performance of the algorithm with the chosen configuration on a custom environment consisting of ten patrol points and ten robots with densely populated obstacles. After generating this world, name it `topic1_10_custom.wbt` and include it in your material.

Provided material:

- `controllers/`
  - `e-puck/e-puck.c`
  - `supervisor/supervisor.c`
- `worlds/`
  - `topic1.wbt` (the world file for this topic)

Note: *The provided material serves as a starting template for your coding. You are encouraged to edit any permitted files or add additional code as needed.*

Topic 2: *Efficient localization and navigation in cluttered environments with a team of robots*

1 Introduction

In a post-disaster scenario, a team of rescue robots must navigate through the debris of collapsed buildings to locate and assist survivors. The environment is filled with rubble, and the GNSS signal is weak and unreliable due to interference. The robots need to work together to find a signal tower that

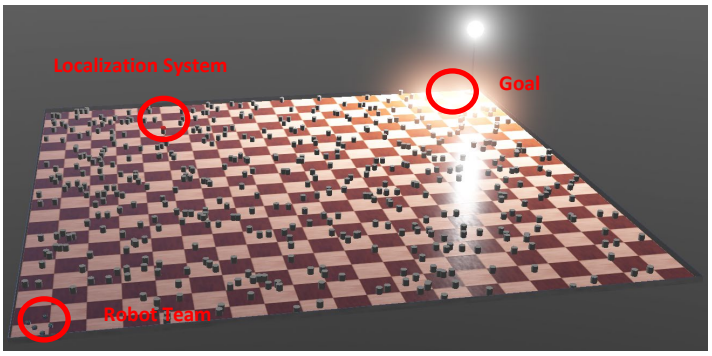

Figure 2: Cluttered arena with light beacon and localization system with a team of five E-Pucks.

provides absolute localization information within the area to accurately localize themselves before continuing their search for survivors at the far end of the area. However, they must stay close together while navigating through a cluttered and dangerous environment. Your task is to ensure the robots can successfully find their way to the survivors despite these challenges.

In this project, you will tackle the localization and navigation problem with a team of five e-pucks. The robots must traverse a cluttered environment and reach a light pole on the opposite side of the area, where the survivors are present.

2 Project description

The main task is to have $R = 5$ robots traverse the arena while maintaining a separation distance $S = 0.15$ m between each pair of closest robots and is deemed complete once all robots are within 1.0 m from a light pole located in the corner of the arena. The pole's coordinates are **unknown to the robots**. Additionally, an absolute localization system is at a location **unknown to the robots**, and its accuracy degrades the further the robots are from it. Thus, the intermediate task is for the robot team to locate the localization system and get as close as possible to it before resuming their course toward the light pole. The localization system broadcasts the robots' position over the radio channel n°1. You can set its update frequency in the controller arguments in the world: `Robot "localization_system" / controllerArgs / <frequency>`. The goal here is to use the **lowest possible rate** to carry out both tasks. The robots **do not know** their initial pose and must estimate it from the available data they can gather from their sensors. Communication between robots and from the localization system can only occur using their emitter/receiver. Communication is omnidirectional and is not hindered by obstacles (emulates Bluetooth communication).

The **e-pucks can only use their default sensors**, specifically wheel encoders, accelerometer and gyroscope, proximity sensors, light sensors, emitter and receiver (Hint: the receiver can return the received signal intensity)

To assess the performance of your solution, you will need to add a supervisor to the world that will track the ground truth position and heading of the robots, as well as declare the mission complete when all robots reach the light pole. To know the location of the localization system or the light pole representing the goal for evaluating the performance of your solution, you can look at the scene tree of the provided Webots world directly.

The ground truth values that you should obtain from the supervisor are marked with a "$*$" below, and the ones you estimate with a "^". The discrete simulation time is denoted t. Positions are denoted by $\boldsymbol{p} \triangleq [x \quad y]^{\mathrm{T}}$ and headings by θ.

You are expected to present the following results:

- Navigation and localization methods, management of both primary and secondary tasks, how you declare the secondary task completed, and theoretical tools employed.

- Localization error metrics in position p_t and in orientation o_t for the robot team over one representative run as a function of the time t:

$$p_t \triangleq \frac{1}{R}\sum_{r=1}^{R} ||\hat{\boldsymbol{p}}_{r,t} - \boldsymbol{p}^*_{r,t}|| \quad \text{and} \quad o_t \triangleq \frac{1}{R}\sum_{r=1}^{R} |\hat{\theta}_{r,t} - \theta^*_{r,t}|$$

- Report the uncertainty in position and heading throughout the run as a function of time for each robot.

- Cohesion metric c_t describing how the team stayed grouped while traversing the arena:

$$c_{r,t} \triangleq \min_{n \in R \setminus r} \left| \left\| \boldsymbol{p}^*_{r,t} - \boldsymbol{p}^*_{n,t} \right\|_2 - S \right| \quad \text{(single robot cohesion)}$$

$$c_t \triangleq \frac{1}{R}\sum_{r=1}^{R} c_{r,t} \quad \text{(group cohesion)}$$

- Localization system's position estimate (as returned by the robot team) and the error as a function of time (same formulation as for p_t).

- Study the impact of the localization system rate on the robots' localization error and cohesion metric and report the lowest rate achieved.

Provided material:

- `controllers/`
  - `localization_system/localization_system.cpp` (DO NOT EDIT)
  - `robot/robot.c` (continue implementing this controller or create a new one)
- `worlds/`
  - `topic2.wbt` (the world file for this topic)

Note: *The provided material serves as a starting template for your coding. You are encouraged to edit any permitted files or add additional code as needed.*

Topic 3: *Optimized flocking and formation control for a team of robots in obstacle-rich environments*

1 Introduction

A team of autonomous vehicles is tasked with navigating a dense forest to deliver essential supplies to a remote research station. The vehicles must stay close to each other to ensure they reach the station together, maintaining communication and coordination. However, they must also navigate through narrow paths and avoid obstacles such as trees, boulders, and uneven terrain. Your mission is to optimize the coordination between these vehicles by using formation and flocking strategies to complete the journey as quickly and safely as possible.

In this topic, you will implement collective movement algorithms in Webots with several e-pucks and develop a formation-flocking controller for a swarm of five robots traversing the arena.

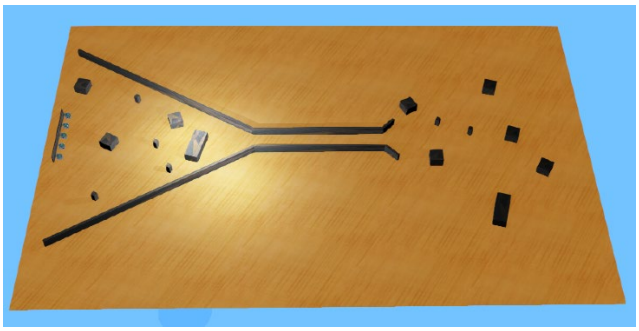

Figure 3. Environment contains obstacles and a corridor with a team of five E-Pucks.

2 Project description

The primary objective of the project is to have the robot swarm navigate to the other side of the arena (with the destination located at {4.2m, 1.7m}) by leveraging formation and flocking control for the given Webots world `topic3.wbt`. The project consists of two main components:

1. *Implementation of Reynolds flocking and Laplacian-based formation controller:* The swarm should be able to first navigate towards the entrance of the corridor using a Reynolds flocking controller. Once at the entrance of the corridor, they should switch to the Laplacian-based formation controller to maintain a line formation to navigate through it. After exiting the corridor, the swarm should revert to the flocking behavior to complete the mission. This transition should be as smooth and natural as possible. The swarm should be able to avoid obstacles while navigating. **The sharing of global positions among robots is prohibited**. **You should only use infrared emitters and receivers to acquire the relative range and bearing measurements** between the robots to formulate flocking and formation. The messages transmitted between robots also contain robot IDs, allowing each robot to be uniquely identified.

2. *Optimization of the flocking and formation behaviors*: You should implement an optimization algorithm (such as Particle Swarm Optimization (PSO), which you will learn in week 11, or other methods you find in the literature) to find the optimal parameters (such as the inter-robot distance, the weight of each rule, etc.) for the flocking and formation controller you designed.

For this task, you must use e-puck robots and apply realistic physical constraints (robot size, maximum robot speed, axis length, number of distance sensors, wheel radius). You should define a series of migration goals to guide the swarm to the desired waypoints. You are free to determine any reasonable inter-robot distance for the Reynolds rule and formation.

The e-pucks **can only use their default sensors**, specifically wheel encoders, accelerometer and gyroscope, GNSS, proximity sensors, emitter, and receiver.

You are expected to present the following results:

- The path of the swarm to complete the mission (you can use supervisor to get access and export the ground truth robot positions).
- Evaluation matrix for flocking based solution:

$$M_{fl}\,[t] = o[t] \cdot d[t] \cdot v[t], \qquad with\ \ o, d, t \in [0,1]$$

The orientation alignment of the robots $o[t]$ is measured as:

$$o[t] = \frac{1}{N}\left|\sum_{k=1}^{N} e^{i\psi_k[t]}\right|$$

where ψ_k represents the absolute heading of robot k and N the number of robots in the flock. The distance between robots $d[t]$ is measured as:

$$d[t] = \left(1 + \frac{1}{N}\sum_{k=1}^{N} |dist(x_k[t], \bar{x}[t]) \ - \ rule1_{thres}|\right)^{-1}$$

where x generally denotes an x, y location, $\bar{x}$ denotes the flock's average location and $rule1_{thres}$ is the distance threshold for the rule 1 (cohesion) of Reynolds' flocking. The velocity of the team towards the migration goal, v[t], is computed as:

$$v[t] = \frac{1}{v_{max}} max(proj_m\left(\frac{\bar{x}[t] - \bar{x}[t-1]}{\Delta T}\right), 0)$$

where $proj_m$ denotes the projection of the flock's velocity onto the vector linking the flock's center of mass to the migration goal, ΔT the sampling interval and v_{max} the maximal speed a robot can achieve.

- Evaluation matrix for formation-based solution:

$$M_{fo}[t] = d[t] \cdot v[t], \qquad with\ d, v \in [0,1]$$

The overall metric is the average of all M_{fo} values for each time stamp t. Moreover, $d[t]$ is computed as:

$$d[t] = \ (1 \ + \frac{1}{N}\sum_{k=1}^{N} ||x_k \ - \ g_k||)^{-1}$$

where N is the number of robots, x_k is the position of robot k and g_k is the target position of robot k in the formation. Note that this metric represents the error of the robots with respect to the central robot (epuck0).

Finally, $v[t]$ is computed as:

$$v[t] \ = \frac{||x[t] - x[t-1]||}{D_{max}}$$

where D_{max} is the maximal distance possible per timestep, given the robot maximum speed v_{max} .

- The formulation of the optimization framework for the controller and comparison of initial and optimized performance.

Provided material:

- `controllers/`
  - `flocking_formation_controller/flocking_formation_controller.c`
  - `super/super.c`
- `worlds/`
  - `topic3.wbt` (The world file for this topic)

Note: *The provided material serves as a starting template for your coding. You are encouraged to edit any permitted files or add additional code as needed.*

Topic 4: *Market-based task allocation with heterogeneous team of robots*

1 Introduction

This project is about implementing a market-based task allocation algorithm in a multi-robot system. Imagine that after a natural disaster, several victims are located in a certain area: Some victims are badly injured and need medical treatment (task A), and others are in shock and only need psychological support (task B). A fleet of robots should reach the victims as quickly as possible and provide the necessary assistance in a limited amount of time. Although autonomous robots can provide both medical and psychological treatment (task A or B), they are specialized in one or the other task, which means they can perform specific tasks faster. As in any search and rescue scenario, the robots have a limited battery life. Therefore, the multi-robot system is assumed to be heterogeneous and energy-constrained. First, you will be asked to implement a centralized strategy. Then, you will implement a distributed strategy. Finally, you will extend the task allocation to a multi-step horizon optimization.

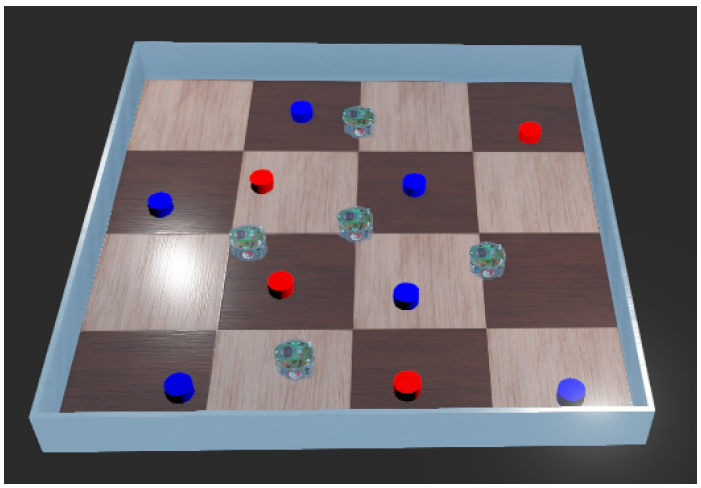

Figure 4. The provided Webots world with 5 robots and 10 tasks of type A and B.

2 Project description

Environment:

- Use the provided world `topic4.wbt`, for this project. It contains custom e-puck robots with a receiver and emitter and a limited communication range of 0.3 m.
- The tasks should appear randomly and uniformly distributed over the environment. There are 10 tasks in total. After a robot completes a task, it should disappear from the environment, and a new one should be created. The probability of a new task being of kind A is 1/3, and of kind B is 2/3. **These probabilities are unknown to the robots a priori**.
- For visualization, color task A in red and the task B in blue.
- The rescue mission lasts for 3 minutes.

Robots:

- At the beginning of the simulation, all robots are placed randomly and uniformly distributed in the environment.
- Each robot may move at a maximum speed of 0.5 m/s.
- There are 5 agents having the following specialization:
    - 2 robots being specialized in task A
    - 3 robots being specialized in task B
- If a robot is closer than 0.1 m to a task center location, it is assumed to complete this task.
- During the task completion, the robot is not allowed to move for a given amount of time:
    - Robot specialized in task A, completing task A: 3s
    - Robot specialized in task B, completing task B: 1s
    - Robot specialized in task A, completing task B: 5s
    - Robot specialized in task B, completing task A: 9s

- The robots are assumed to know the position of all tasks at any given moment.
- The robots have a limited energy budget: They can move or wait for a task completion only for 2 minutes. After they run out of time, they have to shut down and remain static in the world.
- The robots may use GNSS for absolute localization.
- The robots must avoid collisions at any time. A collision is defined as a robot being closer than 0.01 m to a wall or another robot.

Method 1: Centralized Algorithm
- Implement the scenario as described above. You may use Lab 5 as a template. Use a centralized market-based strategy to maximize the total number of completed tasks. The centralized unit can be implemented as part of a supervisor controller. Each agent should be assigned a single task at a time.

Method 2: Distributed Algorithm
- Implement a market-based strategy in a distributed fashion, i.e., without a centralized unit to organize the task allocation. The robots are allowed to broadcast information to their neighbor robots using the emitter and receiver, which have a limited range of 0.3 m.

Method 3: Multiple-Step Planning
- Extend the centralized and distributed algorithms such that the agents plan up to 3 tasks in the future.

Metrics:
- The following metrics have to be reported for every method:
    - Total number of completed tasks
    - Average activation time of the robots as a percentage (average time moving or completing a task over total time)
    - Total number of collisions

Report always the average metric over 5 runs.

You are expected to present the following results in your presentation:

- Give an overview of how you solve the task allocation problem for the three methods.
- Report the metrics and discuss the performance of the methods.
- Compare the three methods implemented quantitatively (metrics) and qualitatively (videos, illustrations, …).

Provided material:

- `worlds/`
    - `topic4.wbt`
- `protos/`
    - `epuck_range.proto`
- `controllers/`
    - `supervisor/`
        - `supervisor.c`
    - `e-puck/`
        - `e-puck.c`

Note: *The provided material serves as a starting template for your coding. You are encouraged to edit any permitted files or add additional code as needed.*