Distributed Intelligent Systems
Course Project

General Information

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

In the lab session of Week 7 (October 30), 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, the report, and further implementation material will have to be submitted. Additional details about the project presentations will be communicated in a timely fashion. The students will be asked to submit their preferences on groups and project topics via Moodle. During the course project period, the last hour of each lab session will be dedicated also to project assistance. In the lab session of Week 13 (December 11), we will provide assistance to the course project for three hours. No further office hours will be made available.

Each group must submit a report. The report is at most four pages long including bibliography and annex. The report should be based on the provided latex template. The final project presentations will be carried out as a team and will take place during the last week of the semester. The presentations should last at most 6 minutes with equal speaking time among team members. 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 14, 23:59.**

Deliverables

Your deliverables include three components:

1. Submit a .pdf file named `DIS_project_report_group_X.ppt` where X is the group number, including:
    a. Clear division of labor and contributions of individuals
    b. Problem statement, formulation, and motivation
    c. Quantitative and qualitative results outlined for each project
    d. Citation of the most relevant methods related to your approach
2. 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. Quantitative and qualitative results outlined for each project
    c. An informative video showcasing your solution in action within the simulation environment
3. 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

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 solution	40 %	Team
Quality of the report	20 %	Team
Quality of the presentation	20 %	Ind.
Quality of the answers delivered during the oral discussion	20%	Ind.

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*

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 metaheuristic optimization technique inspired by the behavior of real ants, which can be used to solve hard 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 TSP.

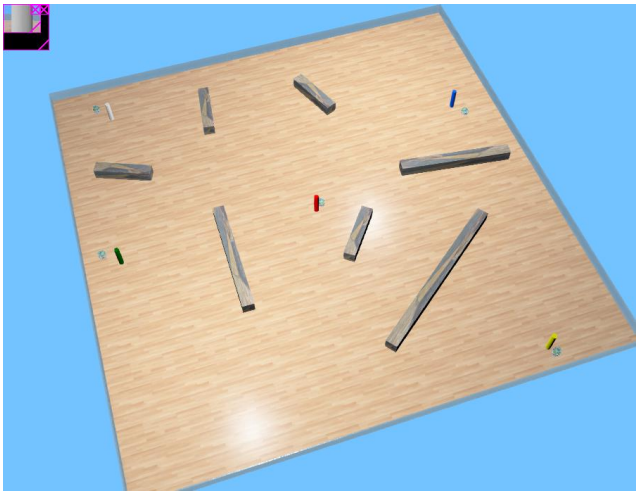

Figure 1. *Provided Webots world with five patrol points, five robots, and obstacles*

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.

Tasks

The project consists of three main components:

1. *Centralized ACO:* In the first part of the project, you will implement and analyze a centralized version of ACO in Webots.
    a. *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**.

    b. *Implementation of Ant Colony Optimization:* The robots should build the 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. Implement the algorithm with the help of a supervisor node in a centralized fashion. **Note that any offline calculation of the shortest path is prohibited.**

   c. *Performance in Complex Environment:* Analyze the robustness of your algorithm in a more complex environment with more patrol points, robots and obstacles.

2. *Distributed ACO:* Implement ACO using a distributed approach, i.e. without the help of a supervisor node. Compare the results to the centralized version, both for the simpler and more complex environments.

3. *Multi-level-Modelling:* Define a microscopic and a macroscopic model which can be used to capture the time that robots spend in obstacle avoidance. Discuss how this model could be used to improve the results of the previous tasks.

Metrics & Results

You are expected to provide the following results:

- Provide an overview of your system architecture for the centralized and decentralized ACO implementations
- 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 the results for the centralized and decentralized search strategy on ACO's accuracy and convergence time.
- The performance of the algorithm with the chosen configuration on a custom environment consisting of at least 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.
- Provide an overview of the microscopic and macroscopic models that you designed including its states, transition rules and parameters.

Provided Material

Provided material:

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

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

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 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.

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.

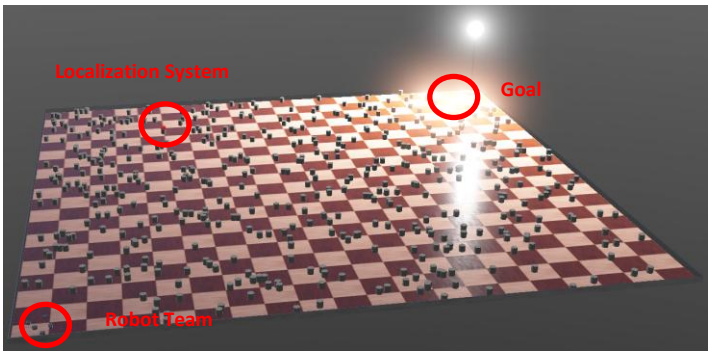

Figure 2: Cluttered arena with light beacon and localization system with a team of five e-pucks.

Description

The main task is to have $R = 5$ robots traverse the 10x10m 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 opposite corner of the arena. The robots do not know their initial position and orientation. The pole's coordinates are **unknown to the robots**, but it can be assumed the robots and the pole are located in opposite corners in the arena. 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 **first 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 at 1 Hz. 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, gyroscope, proximity sensors, light sensors, emitter and receiver (Hint: the receiver can return the signal intensity equal to the inverse of the distance squared; you are allowed to add emitters and receivers to the robots if you need to operate on different channels).

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 are within 1.0 m from the light pole. To know the ground truth 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.

Tasks, Metrics & Results

The ground truth values that you need to 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:

- Method
  - Report on the navigation (finite state machine, controller(s) employed, etc.) and localization methods you implemented (sensor fusion design with EKF, initialization, etc.). The following questions should be clearly addressed:
    - How are inter-robot communications handled (e.g., what information is exchanged and when, etc.)?
    - Could you get robots to help each other localize? How or why not?
    - How did you manage both primary (reaching the light pole) and secondary (reaching the localization system) tasks, and how did you declare the secondary task completed before proceeding with the primary task?
- Localization
  - Report a graph of the robot team trajectory in the arena for a representative run (x, y coordinates sufficient).
  - Report a graph of the localization error metrics in position p_t and in orientation o_t (pay attention how the angle differences are computed) averaged over the robot team over one representative run as a function of the discrete simulation 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 a graph of the uncertainty in position and heading throughout the run as a function of time for each robot.
  - Report a graph of the localization system's estimated position error (as returned by the robot team) as a function of time (same formulation as for p_t).
- Navigation
  - Report a graph of the cohesion metric c_t describing how the team stayed grouped while traversing the arena over time throughout the mission:

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

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

  - Use Particle Swarm Optimization (PSO) to optimize the gains, parameters, thresholds, etc. used by your navigation pipeline and report the performance changes on relevant metrics (at least on total mission duration and cohesion, but feel free to add any other relevant metrics) before (i.e., with hand-tuned parameters) and after optimization (note you can choose whether to use hand-tuned or optimized parameters to report on the previous tasks mentioned above, but state it clearly!). You must report the before/after metrics in a table, including their standard deviation (think about how many runs to use to obtain representative statistics). Report how you implemented the optimization process and clearly list all optimized parameters.

Provided Material

You are provided with the following material to help you get started with the project:

- `controllers/`
  - `localization_system/localization_system.cpp` (DO NOT EDIT)
  - `robot/robot.cpp` (modify and continue implementing this controller)
- `worlds/`
  - `topic2.wbt` (the world file for this topic)

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

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.

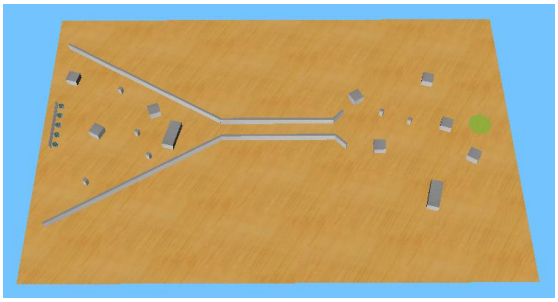

Figure 3. Environment contains obstacles and a corridor with a team of five e-pucks. E-pucks start from the left side of the arena and their goal (green circle) is on the right side at the coordinates {4.2m, 1.7m}.

Description

In the given Webots world `topic3.wbt`, the swarm should be able to first navigate towards the entrance of the corridor using **Reynolds flocking** behavior. Once at the entrance of the corridor, they should switch to a **Laplacian-based formation** to maintain a line formation to navigate through it. After exiting the corridor, the swarm should revert to the flocking behavior to reach their destination: {4.2m, 1.7m}. The transition between flocking and formation behaviors should be as smooth and natural as possible. **Obstacle avoidance** must also be implemented so the robots avoid collisions while navigating.

Note the following points:

- The e-pucks can only use their default sensors, specifically wheel encoders, accelerometer and gyroscope, GNSS, proximity sensors, emitter, and receiver.
- You are free to determine any reasonable inter-robot distance for the flocking and formation.
- For the decentralized architecture: ⚠ **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 can only contain robot IDs**, allowing each robot to be uniquely identified.

Tasks

The project consists of 3 main steps:

A) Implement this task with a *centralized control architecture.*

B) Implement this task with a *decentralized control architecture.*

C) Optimize your flocking and formation parameters (such as the inter-robot distance, the weight of each rule, etc) by implementing *Particle Swarm Optimization* (PSO), which will be presented in class in week 11. The metrics you have to optimize are given in part **2.3** of this assignment, and you can run the PSO either with the centralized or decentralized architecture.

Note : You can also include your local obstacle avoidance parameters in the PSO, but it is not an obligation (indeed the more parameters you try to optimize the longer the optimization algorithm will need to run, which we understand can be difficult to manage).

Metrics & Results

You are expected to present the following results:

- The path of the swarm to complete the mission (you can use the supervisor to get access and export the ground truth robot positions).

- Evaluation metric for flocking based solution:

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

The overall metric is the average of all M_{fl} values for each time stamp t.

-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:

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

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. *C = 20.0* is a coefficient we add to encourage flocking. Indeed given the size of the robots and terrain the error in meters is quite small so we want to give it more importance by rescaling it.

-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 metric for formation-based solution:

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

The overall metric is the average of all M_{fo} values for each time stamp t.

- $d[t]$ is computed as:

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

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. Same as for the flocking, we add the coefficient *C = 20.0* to give more importance to formation for the optimization. Note that this metric represents the error of the robots with respect to epuck0.

- $v[t]$ 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 Materials

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

Tips

- Write your controllers in C or C++, and use Python to implement PSO: Write the optimization algorithm in a Python script that automatically opens Webots.
  This is just a tip, feel free to do it your own way!
- If you want an extra challenge, you can simulate noise to make your simulation more realistic (packet loss during communications, GPS accuracy, …)
- Reminder about centralized and decentralized/distributed architectures: see *Figure 1*.

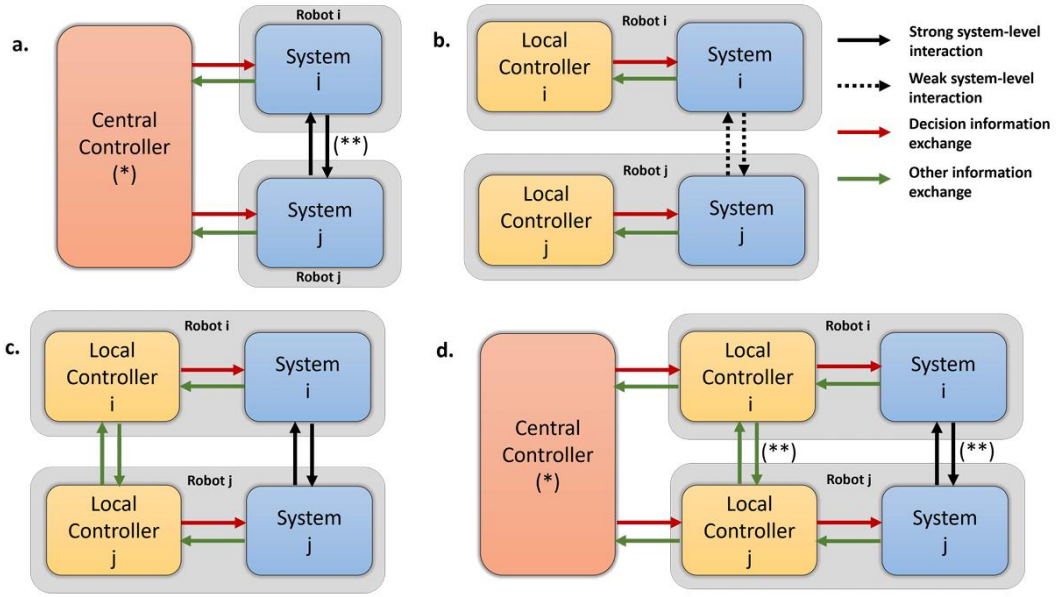

Figure 1. Various control architectures: a. Centralized, b. Decentralized, c. Distributed, d. Hybrid. Note that the green arrows indicate information exchange by communication, the solid red arrows display the flow of decision variables, and the solid and dashed black arrows show strong and weak system-level interaction, such as explicit or implicit communication, respectively. (*) The centralized controller can also be implemented inside another robot. (**) In the centralized and hybrid architectures, strong interactions are shown; however, weak interactions are also possible.
Adapted from "Multi-Robot System Architectures" by C.Baumann, I.K.Erünsal, A.Martinoli.

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

Introduction

In a post-disaster scenario, several victims are located in certain areas: 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. In this project you will tackle a market-based task allocation problem in a multi-robot system. 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.

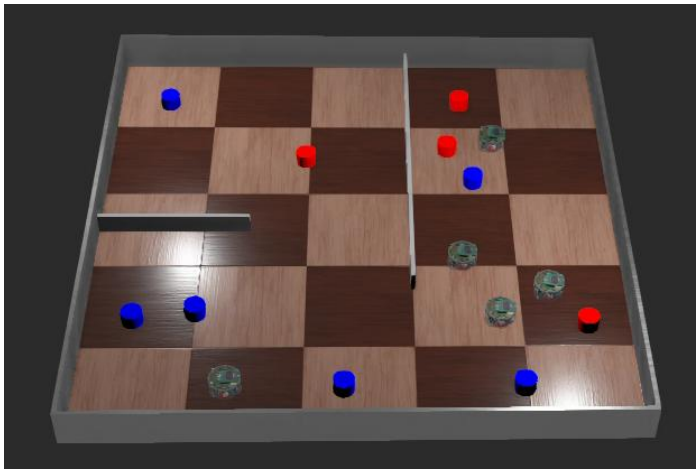

Figure 4. The provided Webots world with 5 robots and 10 tasks of type A and B.

Description

The main task of the project is to reach the victims as quickly as possible, i.e., to provide help to as many victims as possible. For this use the provided world `topic4.wbt`, where the environment is separated in three regions with walls whose coordinates are known a priori. The environment contains custom e-puck robots with a receiver and an emitter having a limited communication range of 0.3m. At the beginning of the simulation, all robots are placed randomly and equally distributed in the environment.

There are 5 agents who have the following specializations:

- 2 robots specialized in task A,
- 3 robots specialized in task B.

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 tasks should be distributed randomly and uniformly in the environment, such that at each moment there are 10 tasks present. 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.

The robots may use GNSS for absolute localization and are assumed to know the position of all tasks at any given moment. The task is considered as reached if a robot is closer than 0.1 m to the task center location. Note that each robot may move at a maximum speed of 0.5 m/s and has a limited energy budget:

it can move or complete a task only for 2 minutes. After a robot runs out of time, it has to shut down and remain static in the world. The robots must avoid collisions at any time. A collision is defined as a robot being closer than 0.05 m to a wall or another robot.

Tasks

The project consists of three main tasks:

1. *Implementation of a centralized algorithm*
   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. You may use Lab 5 as a template.
   a. Implement a single-step planning, where each agent should be assigned a single task at a time.
   b. Implement a multiple-step planning, where each agent plans up to 3 tasks in the future.
2. *Implementation of a distributed algorithm*
   Use 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 0.3m. Note that the centralized and distributed algorithms should be comparable.
   a. Implement a single-step planning, where each agent should be assigned a single task at a time.
   b. Implement a multiple-step planning, where each agent plans up to 3 tasks in the future.
3. *Multi-level modelling*
   Model the number of robots being active (time moving or completing a task) for a centralized market-based strategy with a macroscopic and a microscopic model. You may assume that there is only one type of task and two types of robots that require different times to complete the task.

Metrics & Results

Report always the average metric over 5 runs.

- Tasks 1 and 2:
  - Total number of completed tasks.
  - Average time of the robots doing collision avoidance as a percentage (average time closer than 0.05 m to an obstacle or another robot over average active time).
- Tasks 1, 2 and 3:
  - Average number of active robots (robots either moving to a task or completing it).

You are expected to present the following results:

- Give an overview of how you solve the task allocation problem for the four methods (centralized/distributed & one-step/multi-step). Cite the most relevant publications related to your approach.
- Compare the four methods quantitatively (metrics) and qualitatively (videos, illustrations, …).
- Present the multi-level modeling and what assumptions you are making. Compare the macroscopic and microscopic models to the sub-microscopic model.

Provided Material

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