1 Lab 5: Multi-robot systems coordination using market-based and threshold-based algorithms

This laboratory requires the following equipment:

- C programming tools (gcc, make)

Office hours

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

Information

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

1. The notation $S_x$ means that the question can be solved using only additional simulation.

2. 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.

3. The notation $I_x$ means that the problem has to be solved by implementing a piece of code and performing a simulation.

4. The notation $B_x$ means that the question is optional and should be answered if you have enough time at your disposal.

To prepare yourself for the exam and to allow you for better time planning during the exercise session, we show an indicative number of points for each exercise between parentheses. The combined total number of points for the laboratory or homework exercises is 100.

Division of Labor

This lab is intended to be a simple introduction to two main families of control mechanisms used for coordination and division of labor in multi-agent systems: Threshold-based and Market-based algorithms.

As presented in lecture and in the readings, both of these techniques can be used to perform task allocation in a distributed system. In this lab, we will try to identify and understand some of the subtleties of these methods, their strengths and weaknesses, the trade-offs inherent in their utilization, and what types of situations or problems they are best suited for.

2 Threshold-based Task Allocation Algorithms

A threshold-based stimulus-to-response model is defined by two basic input and output variables as well as by one parameter on which it is built:

1. **Stimulus** $(s)$: A quantity which can be measured locally by an agent, corresponding to a (possibly noisy) perception of the current demand or urgency of a given task.

2. **Threshold** $(θ)$: A parameter internal to an agent, influencing the decision whether or not to act on a given stimulus.

This model, originally proposed for explaining division of labor mechanism in natural societies, has inspired a new family of task allocation algorithms, which we will call threshold-based allocation algorithms. The response to a given stimulus can be deterministic or probabilistic.
When using a probabilistic response, upon encountering or measuring a stimulus \( s \), an agent will decide whether or not to act based on the following formula describing a sigmoid function for the probability of response:

\[
T_{\theta}^{\text{prob}}(s) = \frac{s^n}{s^n + \theta^n}
\]

The exponent \( n \) controls the severity of the change from very unlikely to very likely. In the limit \( n \to \infty \), the response becomes deterministic (the probability density is a step function).

These thresholds \( \theta \) can be either fixed or adaptive. Fixed thresholds, as their name implies, are never changed, while adaptive thresholds allow for a simplistic type of learning in the system. We also have a choice in how the initial thresholds are assigned; we can give all the agents the same threshold value (homogeneous), or we can assign different thresholds to different agents (heterogeneous), either at random, according to prior knowledge of the task to be performed, or because the agents become different (“specialized”) after a learning process.

**Experimental Setup**

We will consider a scenario with the following properties: tasks appear randomly in the environment with a certain arrival rate that must be handled by the agents. The number of open tasks that have not yet been handled serves as the stimulus. The stimulus may be measured by the individual agents in various ways – during the exercises we will explore two of the more obvious possibilities. Each agent has a threshold determining what level of stimulus is necessary before it will respond. In our case study, a responding robot executes a random walk and handles events that it encounters; robots whose thresholds are not met will stop or remain stationary.

In the exercises, we will investigate different ways of distributing and maintaining these thresholds and we will study the impact on the group behavior.

**2.1 Homogeneous vs. Heterogeneous Threshold Distributions**

**Perfect Perception**

First, we’re going to look at a simplified threshold-based task allocation method in a distributed system. In this setting, the stimulus level is announced to the agents via a global broadcast.

You need to download the lab package from the course webpage on Moodle, following these steps:

- Download the `lab05.tar.gz` file from the WEEK 6 section of the course's Moodle site, to the directory of your choice.
• Extract the file with this command: `tar xzvf lab05.tar.gz`
• Launch Webots from a terminal by entering this command: `webots &`

**S1** : Load `lab5thresholds.wbt` in Webots. Compile the provided controller `lab5thresholds.cc` and the physics library `lab5physics.c`. At the top of `lab5thresholds.cc` there are several `#define` statements which control the type of simulation being run – study these briefly before continuing.

**Q2 (5):** The constants defined near the top of `lab5thresholds.cc` are all set to zero. Now set them for **fixed homogeneously distributed thresholds** with a **global stimulus** announcement. This means that every agent will have the same threshold value, and it will not change during the course of the experiment.

The green cylinders represent the robots, and the red ones represent the events. The simulation runs until 15 events have been handled, and then resets after displaying some simple statistics. Run the simulation a couple of times and at the end of each simulation, take note of the statistics printed in the console which give insight to the time-to-completion and percent of average active time. All robots always decide to stop and move at the same moment. Why?

**Q3 (5):** Make the necessary changes in the code and repeat the same scenario with **fixed heterogeneously distributed thresholds** and **global stimulus** announcement. Now, the agents are assigned thresholds randomly, but they continue to use the same value unaltered for the entire duration of the experiment. Again, note what happens, and compare it to the previous simulation.

**Q4 (6):** Briefly state a few of the drawbacks you see in these two approaches, given that we are working on the problem of efficient **task allocation**; that is, we want to get as much as possible done as quickly as possible with as little ‘effort’ (resource expenditure) as possible. (Hint: When does each agent activate? Is the work distributed fairly among the agents?)

**Noisy Perception of the stimulus**

Now consider a different method for stimulus calculation, similar to that used by Agassounon & Martinoli [1]; each agent must estimate locally (independently) what they believe the stimulus to be based only on the information that they have available to them.

**S5:** Run the simulation with **fixed homogeneously distributed thresholds** and **local stimulus estimation**. Here, instead of receiving the number of unhandled events from a supervisor, robots are only able to count the events near them (within a radius of 0.65 times the arena width). Again, notice what happens in comparison to the previous cases.

**Q6 (6):** Which of the previous 2 cases (i.e. Q2 and Q3) does it seem most similar to? Is this what you expected? Explain why. (Hint: If you’re confused, check [1] or the lecture slides.)

### 2.2 Fixed vs. Adaptive Thresholds / Adaptation Rules

As you may have guessed while working with the previous simulations, depending on the particular situation being considered, a certain choice of threshold value(s) may yield better performance, faster completion/convergence, or a fair/unfair distribution of the workload. But what if we don’t know a priori all the necessary details for determining the “best” values, or if we prefer the controller to be flexible, portable, or more broadly applicable? Perhaps we can modify it slightly, so as to endow it with the capability to adapt to the environment with which it is presented.

Of course there are entire laboratories devoted to the study and design of learning algorithms—any of which could certainly be used here, but that’s not our primary focus right now, so try to stick with simple arithmetic functions, such as addition, multiplication, exponents, etc. Also, note that in this particular example we are considering a single task and a single “caste” of agents. During the lecture you have seen how threshold-adaptation rules can be used to generate
specialization in different tasks amongst different “castes” of agents. However, here we are only concerned with adapting a single threshold in order to improve the global performance.

Q7 (6): Here are a few pairs of equations for simple threshold-adaptation rules, considering the conditions which cause them to be executed:

A. \[ \theta = \begin{cases} \theta + k, & \text{condition B} \\ \theta - k, & \text{condition A} \end{cases} \]

B. \[ \theta = \begin{cases} (1 + k_1) \times \theta, & \text{condition B} \\ (1 - k_2) \times \theta, & \text{condition A} \end{cases} \]

C. \[ \theta = \begin{cases} \theta^{(1+k_3)}, & \text{condition B} \\ \theta^{(1-k_2)}, & \text{condition A} \end{cases} \text{ for } \theta > 1 \]

Explain how each one of these equations may change the threshold based on two conditions: What happens when condition A is true? What happens if B is true? What are events A & B?

I8 (7): Select one of the exemplar rulesets above which you feel is the most promising, and write the bodies of the two functions in lab5thresholds_adapt.cc:

adapt_threshold_on_event() (called after an event is handled)
adapt_threshold_on_no_event() (called when no event is handled)

Note: as the no_event version is called quite frequently, you will want it to make a very small change (e.g., by \( k = 0.001 \)). Don’t forget to change ADAPTIVE to 1 in lab5thresholds.cc.

S9: Run a few simulations with your adaptation rule, and note its (quantitative) performance, once with local and another time with global stimulus.

Q10 (6): How does it compare to the non-adaptive version? How is the performance similar to or different from your expectations? Try to think of another change/improvement in the robots’ behavior that may increase the performance of the system.

3 Market-based Task Allocation Algorithms

Market-based approaches are based on economic principles of supply and demand. In market-based multi-robot systems, robots are designed as self-interested agents that operate in a virtual economy. Both the tasks that must be completed and the available resources have quantitative value, and can be traded. In this way, tasks can be assigned to robots via market mechanisms—such as an auction. When a robot completes a task, it receives “payment” (of virtual currency) for providing a service to the team. However, the robot must also pay for the value of the resources it consumed in completing the task. The essence of market-based approaches is that, in a well-designed system, the process of robots trading tasks and resources with one another to maximize individual profit simultaneously fulfills the goal and good team performance.

To illustrate this more concretely, consider a team of robots performing a distributed sensing mission on Mars. Suppose the robots must gather data from specific sites of interest to scientists while consuming the least amount of energy. One important aspect of completing the mission is to determine which robot should visit which site. We can solve this problem with a market-based approach in which robots compete in auctions for the job of visiting a site; the robot with the best bid is awarded a contract for the task. Bids typically correspond to the estimated costs a robot will incur based on the resources it expects to consume to complete the task.

So suppose we offer a maximum reward of $50 and, since the resource of concern is the consumed energy, robots incur a cost of $2 for each meter of travel. This $50 defines a reserve price that essentially says that the task should only be attempted if the site can be reached by some
Distributed Intelligent Systems and Algorithms Laboratory

path of length less than 25 meters. Furthermore suppose that a robot $A$ is only 5 meters from a site $S$; since $A$ would have to spend $10 to complete the task, it bids $12, which includes a 20\% mark-up of its cost that it will keep as profit. Meanwhile, a robot $B$ that is 10 meters from the site would have to spend $20 so it will bid $24. $A$ is awarded the contract because it can perform the task most efficiently and for less than the reserve price.

There are a few main components in a market-based approach:

1. A **global objective (utility) function** that we use to measure the quality of a task allocation. In our example, this objective was to minimize the total energy consumed by the team.

2. A **local objective (utility) function** specified for each robot that quantifies the costs and profits of its activities. In our example, this objective function is the robot’s profit: task reward - $2/meter use of energy.

3. A **mapping** between the global objective function and the local objective function that determines how individual activities contribute to the overall solution. In our example, the sum of the individual energy consumption.

4. A **market mechanism** that we use to redistribute tasks. Here, we used an auction.

The Market: Trade-goods and Objective Functions

Consider the following multi-robot problem: there are a set of hazardous spills sites scattered in a nuclear power plant that need to be cleaned up. Each spill must be cleaned by exactly one robot, and spills require different amounts of time to be cleaned up. We want to distribute these spill sites to the robots so that we minimize the time taken to clean the spills.

**Q11 (4):** What are some commodities (things being bought & sold) in this setup? What are the robots trading? (Hint: there are two main ones.)

For the next two questions, you may use the following type of notation:

- $\text{Distance}(r, s_j)$ the distance between a robot and a spill site
- $\text{TaskList}(r_i)$ the list of spills a robot currently plans on managing

You may use whatever information you need, but please explain it. And remember that for the local objective function, it must be feasible for the robot to have access to the information you use in the function.

**Q12 (4):** “Minimizing the time taken to clean all the spills” (our **global objective function**) can mean a number of different things. Describe at least two ways we can interpret this goal. (Express them as simple equations, using $t_i$ to represent the time at which the cleaning of spill $i$ is completed.)

**Q13 (4):** Write an equation for the **local objective function** used on the individual robots (e.g. what a robot $r_i$ bids to cleanup site $s_j$). Assume that each robot has complete and correct information about itself and about the spills (e.g., their locations and whether they are finished or not) but no information about its teammates.

3.1 Experimenting with different auction and bidding strategies

The simulation experiments for this portion of the lab use the files **lab5markets.cc** and **lab5markets_bidding.cc**, which can be compiled to run in Webots (as with the threshold simulations).

These questions will require you to change the bidding function in the file **lab5markets_bidding.cc** (**nothing needs to be changed in lab5markets.cc**). Details about the working of the algorithm can be found in the comments contained in the code. Essentially,
there is a list of events that must be assigned to the robots. The central auctioneer requests one bid from each of the robots and, based on these bids, assigns an event to a robot. In the default setup provided, robots construct tours of events incrementally; that is, a robot only considers adding an event to the end of its to-do list (there is no mechanism provided for changing the portion of the schedule which has already been assigned to a robot—implementing such a mechanism is not part of this lab).

**S14:** Run the simulation several times as it is, taking note of the performance.

**I15 (7):** Currently, there is a rather simple default bidding strategy in the function `CollectBid()`. We implemented a slightly better bidding strategy for you (commented-out in the function `CollectBid()` under `// === improved bidding strategy ===`)—uncomment this code, recompile and run the simulation.

Why is this a better strategy?

**Q16 (4):** Quantitatively compare the performances of the provided and improved bidding strategies.

**Q17 (6):** Consider the setup below, using the slightly modified bidding strategy from I15. We have two robots and two spill sites and the distances between them are shown (not to scale). By inspection, what is the optimal assignment? Now, suppose that we auction S1 and then S2. What is the resulting solution? Explain this problem and give an auction strategy that will solve it. (Hint: notice that the `Bid` specifies which `Event` is being bid on.)

I18 (10): Implement this auction strategy in your code and compare the results with the version you implemented previously. Hint: you need to do your modifications in the `Robot::CollectBid()` function in `lab5markets_bidding.cc` file. Read and follow the directions provided under the line with “// === even better strategy ===”.

### 3.2 Heterogeneity in market-based systems

Now consider a slightly more complex problem. We again have the hazardous spills domain. However, this time spills are either of chemical A or of chemical B. Again, we are interested in minimizing the time taken to handle spills.

**Q19 (5):** Suppose that we also have two types of robots. A-robots can only handle A-spills, and B-robots can only handle B-spills. Will we have to change the global objective function, and if so, how? Will we have to change the local objective function, and if so, how?

**Q20 (5):** Suppose that we have A-Robots and B-Robots again, but now A-robots are better (e.g., faster) at A-spills than B-spills, and the opposite is true for B-robots. Will we have to change the global objective function and, if so, how? Will we have to change the local objective function from the previous question and, if so, how?
4 Comparing market-based and threshold-based approaches for task allocation

Q21 (5): For what types of scenarios are market-based approaches most well suited? What about for threshold-based approaches? Consider the task information available to the team, the (different) physical capabilities of the team members, availability of communication and/or computational power, and the requirements/measures of success. Try to come up with a couple of general indicators that would cause you to select one over the other.

B22 (5): Consider the following scenario: You have been hired to modernize a production line by using a fleet of mobile robotic agents. The production flow is defined by a series of interdependent tasks (the output of one task serves as input for the next). The robots are generalists capable of handling any of the tasks by switching among their tools (with a certain time penalty). Which of the two division of labor methods presented in this lab would you implement to obtain a system that is adaptive to variations in the raw material supply and robust to robot brake-down? Justify your choice and explain the fundamental elements (e.g., stimulus, threshold for threshold-based, and commodities, local/global objective functions for market-based) of the respective method in the presented context.

References