# Master 2
# UE NI402 - ARA
# (Homework)

Franck Petit
franck.petit@lip6.fr

**Assignment.** This homework can be done alone or in pairs. Possible languages are French and English.

Deadline: **November 29th, 2013** by email addressed to me. Please make sure to prefix the subject of your email with [ARA 2013]. Thank you.

## 1 General Presentation

We consider autonomous robots that are endowed with motion actuators and visibility sensors, but that are otherwise unable to communicate. Those robots must collaborate to solve a collective task, here the *terminating exploration* (*exploration* for short), despite being limited with respect to inputs from the environment, asymmetry, memory, *etc.* In particular, the robots we consider are anonymous, uniform, yet they can sense their environment, and take decisions according to their own ego-centered view. In addition, they are oblivious, *i.e.*, they do not remember their past actions.

## 2 Model

**Distributed Systems.** We consider systems of autonomous mobile entities called *robots* evolving in a *simple unoriented graph* $G = (V, E)$, where $V$ is a finite set of $n$ nodes (*i.e.*, $|V| = n$) and $E$ a finite set of edges, *i.e.*, unordered pairs of distinct nodes. Nodes represent locations that robots can take and edges represent the possibility for a robot to move from one location to another. Two nodes $u$ and $v$ are *neighbors* in $G$ if and only if $\{u, v\} \in E$.

We assume that $G$ is a *Square O-Grid* (also referred to as a *Square Cylinder*), *i.e.,* there exists a positive integer called *side*, $S$, such that $G$ satisfies the following two conditions: (*i*) $n = S^2$ and (*ii*) there exists an order $v_1, \ldots, v_n$ on the nodes of $V$ such that $\forall i \in [1..n]$:

- If $i + S \leq n$, then $\{i, i + S\} \in E$, else $\{i, (i + S) \bmod n\} \in E$.
- If $i \bmod S \neq 0$, then $\{i, i + 1\} \in E$.

In this homework, we assume that (*i*) $S$ is odd, and (*ii*) $7 \leq S$. Given the previous order $v_1, \ldots, v_n$, for every $i \in [1..S]$, the sequence $r_i = v_i, v_{i+S}, v_{i+2\times S}, \ldots,$

$v_{i+(S-1)\times S}$ is called a *ring*. Similarly, $\forall i \in [1..S]$, $c_i = v_i, v_{i+1}, \ldots, v_{i+(S-1)}$ is called a *chain*. Note that there is no way for a robot to orient a ring or a chain. For every $i \in [1..S]$, the chains $c_i$ and $c_{i \bmod S+1}$ are said to be neighbors. Similarly, for every $i \in [1..(S-1)]$, the rings $c_i$ and $c_{i+1}$ are neighbors. The unique ring $r_i$ such that $i = \lfloor \frac{S}{2} \rfloor$ is called the *middle* ring.

A square O-Grid of side $S$ can be seen as a $(S,S)-$*grid* such that nodes of two opposite borders are linked together to form $S$ rings. Two other representations of the square O-Grid of side $S = 11$ are given in Figure 1.

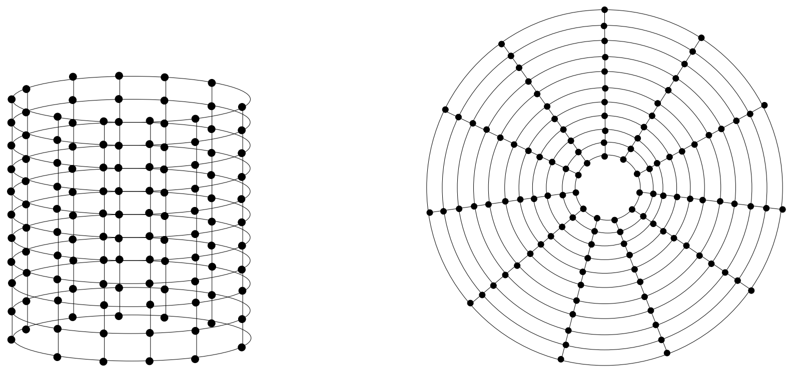
Figure 1: An example of a square O-Grid ($S = 11$).

**Remark 1** *Since a square O-Grid is a simple graph, then* $3 \leq S$ *(*$n = S^2 \geq 9$*).*

Nodes are assumed to be anonymous (they have no access to identifiers or other symmetry breaking capabilities). Moreover, given two neighboring nodes $u$ and $v$, we assume that there is no explicit or implicit labeling allowing the robots to determine whether $u$ is either on the left, on the right, above, or below $v$. However, for the purpose of proof description, we may use indices for nodes or robots.

**Robots and Computation.** Operating on $G$ are $k$ robots. The robots do not communicate in an explicit way; however they see the position of all other robots in their ego-centered coordinate system and can acquire knowledge from this information.

Each robot operates according to its (local) *program*. We call *protocol* a collection of $k$ *programs*, each one operating on a single robot. Here we assume that robots are *uniform* and *anonymous*, *i.e.*, they all have the same program using no parameter allowing to differentiate them.

The program of a robot consists in executing *Look-Compute-Move cycles* infinitely many times. That is, the robot first observes its environment (Look phase). During the Compute phase, based on its observation, a robot then decides to move or stay idle (Compute phase). The decision between moving or staying idle is **either** *deterministic* **or** *probabilistic*. In the former case, it a robot decides to move, then it (deterministically) moves toward its destination. In the latter case, the robot decides between moving (according to its observation) and staying idle using some fixed probability $p$, with $0 < p < 1$. We say that the robot *tries to move*.

We assume that robots cannot remember any previous observation nor computation performed in any previous cycle. Such robots are called *oblivious*.

We consider the semi-synchronous model. In this model, time is represented by an infinite sequence of instants 0, 1, 2, ... No robot has access to this global time. At each instant, a non-empty subset of robots is *activated.* Every robot that is activated at instant $t$ *atomically* executes a full cycle between $t$ and $t+1$. Activations are determined by an *adversary.*

Note that in this model, any robot performing a Look operation sees all other robots on nodes and not on edges.

**Multiplicity.** We assume that during the Look phase, every robot can perceive whether several robots are located on the same node. This ability is called *multiplicity detection.* We shall indicate by $d_i(t)$ the multiplicity of robots present in node $v_i$ at instant $t$. We consider two versions of multiplicity detection: the *strong* and *weak* multiplicity detections. Under the *weak* multiplicity detection, for every node $v_i$, $d_i$ is a function $\mathbb{N} \mapsto \{\circ, \perp, \top\}$ defined as follows: $d_i(t)$ is equal to either $\circ$, $\perp$, or $\top$ according to $v_i$ contains none, one or several robots at instant $t$. If $d_i(t) = \circ$, then we say that $v_i$ is *free* at instant $t$, otherwise $v_i$ is *occupied* at instant $t$. If $d_i(t) = \top$, then we say that $v_i$ contains a *tower* at instant $t$. Under the *strong* multiplicity detection, for every node $v_i$, $d_i$ is a function $\mathbb{N} \mapsto \mathbb{N}$ where $d_i(t) = x$ indicates that there are $x$ robots in node $v_i$ at instant $t$. If $d_i(t) = 0$, then we say that $v_i$ is *free* at instant $t$, otherwise $v_i$ is *occupied* at instant $t$. If $d_i(t) > 1$, then we say that $v_i$ contains a *tower (of $d_i(t)$ robots)* at instant $t$.

**Configurations, Views and Executions.** An *isomorphism* of graphs $G$ and $H$ is a bijection $f$ between the vertex sets of $G$ and $H$ such that any two nodes $u$ and $v$ of $G$ are neighbors in $G$ if and only if $f(u)$ and $f(v)$ are neighbors in $H$. When $G$ and $H$ are one and the same graph, $f$ is called an *automorphism* of $G$.

To define the notion of *configuration*, we need to use an arbitrary order $\prec$ on nodes. The system being anonymous, robots do not know this order. Let $v_1, \ldots, v_n$ be the list of the nodes in $G$ ordered by $\prec$. The configuration at instant $t$ is $d_1(t), \ldots, d_n(t)$. We denote by *initial configurations* the configurations from which the system can start at instant 0. Every configuration from which no robot moves or tries to move if activated is said to be *terminal.*

The *view* of robot $\mathcal{R}$ at instant $t$ is a labeled graph isomorphic to $G$, where every node $v_i$ is labeled by $d_i(t)$, except the node where $\mathcal{R}$ is currently located, this latter node $v_j$ is labeled by $d_j(t), *$. (Indeed, any robot knows where it is located.) Hence, from its view, a robot can compute the view of each other robot, and decide whether some other robots have the same view as its own. The views $\mathcal{V}$ and $\mathcal{V}'$ are *identical* if and only if there exists an isomorphism $f$ of $\mathcal{V}$ and $\mathcal{V}'$ such that every node $v$ of $\mathcal{V}$ has the same label in $\mathcal{V}$ than $f(v)$ in $\mathcal{V}'$.

Every decision to move is based on the view obtained during the last Look action. However, it may happen that some edges incident to a node $v$ currently occupied by the deciding robot look identical in its view, *i.e.*, $v$ lies on a symmetric axis of its view. In this case, if the robot decides to take one of these edges, it may take any of them. We assume the worst-case decision in such cases, *i.e.*, the actual edge among the identically looking ones is chosen by the adversary.

Two configurations $d_1, \ldots, d_n$ and $d'_1, \ldots, d'_n$ are *indistinguishable* (resp., *distinguishable* otherwise) if and only if there exists an automorphism on $G$, $f : V \mapsto V$ such that $\forall i \in \{1, \ldots, n\}$, $d_i = d'_j$ where $v_j = f(v_i)$.

Consider two indistinguishable configurations $\gamma$ and $\gamma'$. Assume that some robot $\mathcal{R}$ is located at node $v_i$ in $\gamma$. Then, by definition, there is some robot $\mathcal{R}'$ located at node $f(v_i)$ in $\gamma'$. Moreover, the view of $\mathcal{R}$ in $\gamma$ is the same as the view of $\mathcal{R}'$ in $\gamma'$. So, if $\mathcal{R}$ is activated in $\gamma$ and may decide to move to some neighboring node $v_j$, then it is the same for $\mathcal{R}'$: if $\mathcal{R}'$ is activated in $\gamma'$, it may decide to move to $f(v_j)$.

A *scheduling* is a list of activation's choices that can be made by the adversary, *i.e.*, a scheduling is any infinite list of non-empty subset of robots $\sigma_0, \sigma_1, \ldots$, where $\forall i \geq 0$, $\sigma_i$ is the set of robots activated at instant $i$. An infinite list of configurations $\gamma_0, \gamma_1, \ldots$ can be *generated* from the scheduling $\sigma_0, \sigma_1, \ldots$ if and only if $\forall i \geq 0$, $\gamma_{i+1}$ can be obtained from $\gamma_i$ after each robot in $\sigma_i$ is activated at instant $i$ to atomically perform a cycle (in this case, $\gamma_i \gamma_{i+1}$ is *step*).

We call *execution* any infinite list of configurations $\gamma_0, \gamma_1, \ldots$ that can be *generated* from an arbitrary scheduling and such that $\gamma_0$ is a possible initial configuration. An execution $e$ *terminates* if $e$ contains a terminal configuration.

We restrict the power of the adversary by assuming that schedulings are *fair*: a scheduling $\sigma_0, \sigma_1, \ldots$ is *fair* if and only if for every robot $\mathcal{R}$, for every instant $i$, there exists an instant $j \geq i$ such that $\mathcal{R} \in \sigma_j$. An execution $e$ is *fair* if and only if $e$ can be generated by a fair scheduling.

A particular case of fair scheduling is the *sequential* fair scheduling: a scheduling $\sigma_0, \sigma_1, \ldots$ that is fair and such that $\forall i \geq 0$, $|\sigma_i| = 1$. An execution $e$ is *sequential fair* if it can be generated from a sequential fair scheduling.

**Exploration.** We consider the *exploration* problem, where $k$ robots, initially placed at different nodes of a graph $G = (V, E)$, collectively explore $G$ before stopping moving forever. By "collectively" we mean that every node of $G$ is eventually visited by at least one robot. More formally, we say that a protocol $\mathcal{P}$ *deterministically* (resp., *probabilistically*) solves the exploration problem assuming a fair scheduling if and only if every fair execution $e$ of $\mathcal{P}$ starting from a *towerless* configuration satisfies: *(1)* $e$ reaches a terminal configuration *in finite time* (resp., *with probability one*), and *(2)* every node is visited by at least one robot during $e$. Note that the previous definition implies that every initial configuration are *towerless*. Note also that in case of probabilistic exploration, termination is not certain, however the overall probability of non-terminating executions is 0. Observe that the exploration problem is not defined for $k > n$ and is straightforward for $k = n$. (In this latter case, the exploration is already accomplished in the initial towerless configuration.)

## 3 Algorithm

Algorithm 1 shows a probabilistic protocol to explore any square O-Grid of **odd side greater than or equal to 7**.

It requires **exactly $k = 4$** robots with weak multiplicity detection.

We fist need the following definition. A configuration $C$ is said to be a $Z$-configuration if and only if the three following conditions are satisfied:

1. $C$ contains no tower,
2. Exactly 2 robots are located on the middle ring in $C$, and
3. Two robots are located on 2 neighboring nodes of the middle ring so that each of them have exactly two robots in their neighborhood.

**Question 1** *Draw a picture showing a $Z$-configuration on a square O-Grid of side 7. By convenience, design your picture using a grid representation (see for instance Figures 2 and 3).*

The algorithm works in three distinct successive phases: **Phase SetUp**, **Phase Tower**, and **Phase Exploration**. Starting from any towerless configuration, the aim of the **Phase SetUp** is to arrange the robots in such a way that they eventually form a $Z$-configuration, without creating any tower during the process. This phase is probabilistic.

Once a $Z$-configuration is built (by Phase Setup), **Phase Tower** begins. This phase is also probabilistic and consists in creating a tower on the middle ring using the two robots that have exactly two robots in their neighborhood. As the scheduler is fair, both of them are activated regularly until a tower is created. So, this latter event eventually occurs with probability one.

When the tower is created, the location of robots give an explicit orientation to the O-Grid. Then, the last phase, **Phase Exploration**, begins. This phase is deterministic. The two single robots collaborate together to deterministically explore the O-Grid and eventually stop.

**Algorithm 1** Main Program.

```
1.01  if the configuration contains no tower and is not a $Z$-configuration
1.02  then  Execute Procedure Phase SetUp
1.03  else  if the configuration is a $Z$-configuration
1.04            then  Execute Procedure Phase Tower
1.05            else  Execute Procedure Phase Exploration
```

We first study Phases Tower and Exploration. We will consider Phase Setup thereafter.

## 3.1 Phase Tower

**Question 2** *Give an algorithm for Phase Tower.*

**Question 3** *Prove that the algorithm provided in Question 2 creates a tower with probability 1.*

## 3.2 Phase Exploration

Again, we need some definitions. Given two nodes $u$ and $v$, let $R_{uv}$ be the *smallest enclosing rectangle* that includes both $u$ and $v$, refer to Figure 2. Let $\alpha_{uv}$ ($\beta_{uv}$) be the length in terms of hops of one of the smallest (resp., greatest) side of $R_{uv}$, *e.g.*, $\alpha_{uv} = 2$ and $\beta_{uv} = 4$ in Figure 2. $R_{uv}$ is an $(\alpha_{uv}, \beta_{uv})$-rectangle. The (Manhattan) *distance* between two nodes $u$ and $v$, denoted by $d_{uv}$, is equal to $\alpha_{uv} + \beta_{uv}$, *e.g.*, $d_{uv} = 6$ in Figure 2. We define a total order on distances as follows: given four nodes $u$, $v$, $u'$, and $v'$, $d_{uv} \leq d_{u'v'}$ *iff* either $d_{uv} < d_{u'v'}$ or $d_{uv} = d_{u'v'}$ and $\beta_{uv} \leq \beta_{u'v'}$.

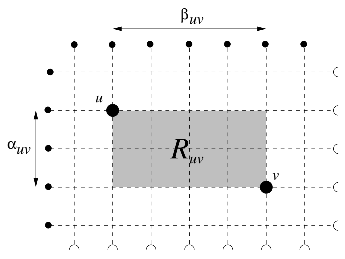

Figure 2: Smallest Enclosing Rectangle.

The deterministic exploration starts assuming that two robots form a tower. Denote the node holding the tower by `T`. In the sequel, '`o`' (respectively, '`*`') denotes the *nearest* (respectively, *farthest*) single node (or robot) from `T`, *i.e.,* $d_{\mathtt{o}T} < d_{*T}$.

**Question 4** *Given a node $u$, if $\alpha_{Tu} < \beta_{Tu}$ and ($\alpha_{Tu} \neq \beta_{Tu} \neq \lfloor \frac{S}{2} \rfloor$, then there exists an orientation of O-Grid such that $u = (\alpha_{Tu}, \beta_{Tu})$.*

In the sequel, when it is possible, we build a coordinate system over $R_{Tu}$ by setting the $x$-axis (the $y$-axis) as the axis (ring or chain) that is parallel to the smallest (resp., greatest) side of $R_{T,*}$ and by orienting both axis so that the coordinates of $u$ are positive.

The main idea of Phase Exploration is the following: both robots that are not part of the tower collaborate together in order to explore the whole O-Grid. They alternate between two roles: *Explorer* and *Leader*. Leader $L$ allows to build a coordinate system $S^L$ over $R_{\mathtt{T}L}$. The explorer is in charge of deterministically exploring the O-Grid over $S^{\mathrm{L}}$. The exploration works in two phases, executed one after another.

**Phase Exploration-1.** Figure 3 illustrates this phase. Robot `*` (*i.e.,* the farthest robot of `T`) plays the Leader role. Starting from the configuration built by Phase Tower, Robot `*` first built a $(1, 2)$-rectangle with `T` by moving in the opposite direction to `o`—refer to Moving #1 in Figure 3. $R_{\mathtt{T}*}$ allows to build a coordinate system $S^*$, where Robot `*` occupies Node $(1, 2)$ *w.r.t.* $S^*$. Then, Robot `o` initiates a spiral-shaped exploration. It visits the nodes that form the first surrounding square around `T` and stops at node $(-1, -1)$—Moving #2 in Figure 3.

Next, Robot `*` moves to node $(2, 3)$ passing through node $(1, 3)$—Movings $3'$ and $3''$. Then, Robot `o` visits the nodes that form the second surrounding square around `T` and stops at $(-2, -2)$—Moving #4. Finally, Robot `*` moves back to $(1, 3)$, followed by Robot `o` that moves back to $(-2, -1)$—Movings #5 and #6. This ends the first phase.

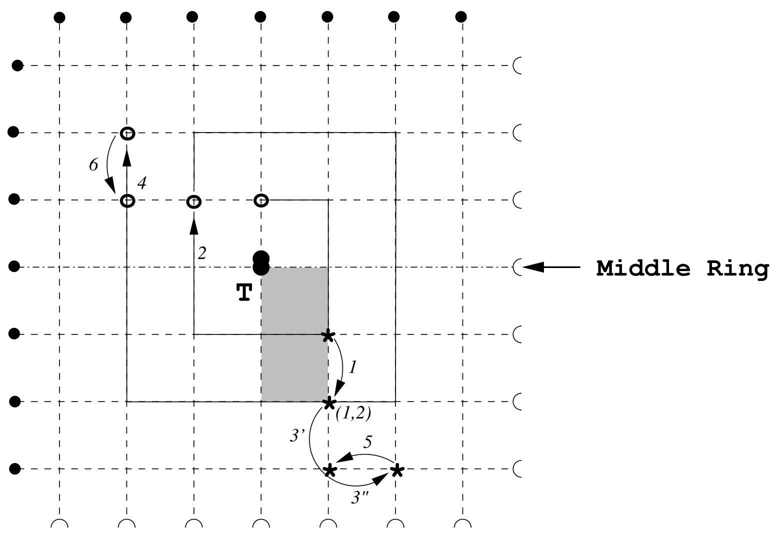

Figure 3: First phase of exploration.

**Question 5** *Give a formal algorithm performing Phase 1.*

Note that our method requires the three following conditions:

1. Robot `*` must be able to move at least three lines away from the tower,
2. Robot `o` must be able to visit the two squares centered on the tower, and
3. The orientation built by Robot `*` must be unambiguous.

**Question 6** *Show that the three above conditions together require $S \geq 7$.*

**Phase Exploration-2.** In this phase, Robot `o` now plays the part of the leader. $R_{\mathrm{T}o}$ provides a coordinate system $S^\circ$, where Robot `o` is located at $(1,2)$. Phase 2 is managed by Procedures *ExploP*21 and *ExploP*22 (refer to Algorithms 2 and 3).

**Algorithm 2** Procedure *ExploP*21 executed by Robot `*`.

```
2.01  Let $S^\circ$ be the coordinate system built over $R_{\mathrm{T}o}$ // $d_{\mathrm{T}o} = 3$
2.02  Let $(x, y)$ be my current coordinates over $S^\circ$
2.03  Let $phase = \max\{|x|, |y|\}$
2.04  if $(x, y) = (-phase, -phase)$
2.05  then if $phase = \lfloor \frac{S}{2} \rfloor$
2.06       then Exploration Done!
2.07       else Move to Position $(x - 1, y)$
2.08  else Execute $ExploP22(phase)$
2.09  endif
```

**Algorithm 3** Procedure *ExploP*22.

```
Procedure $ExploP22(phase)$
    Let $(x, y)$ be my current coordinates over $S^\circ$ // i.e., over $R_{\mathrm{To}}$
    if $y = -phase$
    then Move to Position $(x - 1, y)$
    else if $y = phase$
         then if $x = phase$
              then Move to Position $(x, y - 1)$
              else Move to Position $(x + 1, y)$
         else if $x = -phase$
              then Move to Position $(x, y + 1)$
              else Move to Position $(x, y - 1)$
              endif
         endif
    endif
end
```

**Question 7** *Similarly as in Figure 3, sketch the behavior of Phase 2 in an O-Grid. Choose* $S = 15$.

## 3.3 Phase Setup

**Question 8** *Try to provide either a formal algorithm or a sketch of Phase Setup. Include explanations.*

## 3.4 Lower Bound

In this section, discuss the minimality in terms of number of robots.

**Question 9** *Do you think that it could be possible to achieve a deterministic algorithm with only* 4 *robots? If yes, then sketch your algorithm. If not, what about* 5 *robots? Would it be possible to achieve a probabilistic or deterministic algorithm with* 3 *robots?*

**Question 10** *Would it be possible to achieve a probabilistic or deterministic algorithm with* 3 *robots?*