ALMA MATER STUDIORUM – UNIVERSITY OF BOLOGNA
CAMPUS OF CESENA

Engineering and Architecture Faculty
Master's Degree in Engineering and IT

ROBOT EXPLORATION

Subject
INTELLIGENT ROBOTIC SYSTEMS

Presented by
GABRIELE GUERRINI

Accademic Year 2019 – 2020

Table of Contents

Chapter 1

Project goal

Few landmarks are scattered around the arena, and the main goal is to explore them all using swarm robotics techniques (non-deterministic transition between executed tasks, local sensing, low computational power of each robot etc.). Each landmark should be explored exactly once.
All robots are initially located in a base placed somewhere in the arena. The base is identified by a black spot on the ground, and a light source is placed above it.
A robot randomly starts exploring the arena and, when a landmark is detected, it may stop in order to create/join a cluster around such landmark. The probability to start exploration depends upon the time spent resting.
When a cluster is complete, the robots return to the base by executing phototaxis after a short phase of recognition (that would be the phase where the robots should actually explore and map the landmark surrounding area, but in this project it will be mocked for simplicity). The probability of the robot to join the cluster depends upon two parameters, i.e. the number of the robots that already joined the cluster and the time spent exploring.
A robot may leave a joined cluster randomly before it is completed depending on the two same aforementioned parameters.
A robot may quit exploration task randomly and return to the base. The probability to end such task is proportional to the spent exploration time.
In the end, few obstacles may be distributed in the arena, and so an obstacle avoidance activity must be executed along all the robot lifetime. Such task includes avoidance of any obstacle (walls and boxes as well as other robots and landmarks).

Chapter 2

State of the art

An overall of four different well-known tasks can be detected in the problem, namely:

- **Arena exploration**: the robots must explore the arena to find the landmarks.
- **Clustering**: the robots must create cluster near any landmark so it can be considered explored.
- **Phototaxis**: used to allow the robots to easily come back to the base.
- **Obstacle avoidance**: general behaviour to be executed to prevent any collision (between robots as well as between robots and obstacles/walls).

It follows a brief description of the state of the art for each task.

2.1 Arena exploration

The exploration of the area is a well-known task in the literature, and lots of algorithms and methodologies have been studied. However, when talking about swarm robotics, the main adopted method is the *random walk* approach. This is mainly due to the usage of off-the-shelf robots that own limited individual abilities (low processing power, local sensing etc.). Nevertheless, the usage of a swarm of robots for exploration tasks is still widely applied thanks to the advantages it entails respect to the usage of a single robot (flexibility, robustness, scalability).

There are few different common sense implementations of a random walk task [1], namely:

- Brownian motion
- Lèvy Flight
- Lèvy Taxis
- Correlated radom walk
- Ballistic motion

The main idea is to randomly choose a direction and go straight till a new one is selected.
All the aforementioned models share the same underlying mathematical model, and the different tuning of the parameters provides different behaviours and exploration capabilities. Only one exception can be identified, that is the ballistic motion, where the two parameters *step length* (μ) and *turning angle* (ρ) are not provided and/or limited.

Further from the above basic approaches, new solution have been probed to overcome the limits they impose. Indeed, there are two main problem that emerge, that is:

- Execution of repeated exploration of the same point.
- Exploration does not scale with arena widening. Ineed, just zona near the starting position will be quite well explored.

New approaches have been studied to overcome the aboe issues. The example reported in [2] improves the random walk by considering relative distribution of robots among the arena so that each zone can be equally explored by a even quantity of robots.

Once the robots are spread over the arena, different algorithm can be applied to map it (e.g., *GMapping* that produces a two-dimensional occupancy grid of the environment).

2.2 Clustering

Clustering is one of the main tasks identified in most of the proposed taxonomies for swarm robotics. The goal is to gather objects (the robot themselves or tokens to be moved) in one or more points of the arena. The task can be used for multiple purposes: foraging activity (group food supplies), divide robots upon classes that execute different tasks, etc..
All the proposed clustering algorithms create different solutions upon one of three common approaches [10], that is:

- Force-oriented solutions (similar to motor schema principles).
- Probabilistic approach by using non-deterministic transition among states and tasks. Indeed, as stated in [13] [9], randomness resulted to be a crucial aspect for emerging behaviours in swarm robotics in general.
- Artificial evolution through genetic algorithms.

Independently from the ad hoc approach, the main idea is to apply *swarm intelligence* principles to guide robots behaviour, and so to create a result inspired to various animal species (insects, birds, fishes) or natural laws (e.g., the settling process of liquids of different densities, [11]).
Beyond this, few more articulated experiments have been executed trying to add further capabilities to robots (e.g., *spatial awareness* and exchange of *virtual tokens*, [12]).
In the end, notice that swarm intelligence principles can be applied to other domains as well. Indeed, lots of experiments have been made about the clustering of data (instead of robots) in order to overcome traditional clustering algorithms such as *K-means* by blending swarm intelligence with other domains (e.g., game theory, [9]).

2.3 Obstacle avoidance

The obstacle avoidance task has been widely studied, and so few different algorithms have been proposed. The input values can be retrieved with distance sensors (e.g., sonar, proximity sensors) and/or visual sensors. In this case, we are going to focus on techniques that exploit the first ones solely.

The most known technique is the *Artificial Potential Field (APF)* that is based upon the motor schema idea by creating two potential fields: the first is attractive and generated by the goal point so that the robot can aim toward it, the second one is repulsive and it is generated from obstacles. Obviously, even though this method suffers all the issues of the motor schema approach such as local minima, it finds shorter paths than other well-know algorithm such as the Bug one [3] [4].
Other solutions, even though rely on the same underlying mathematical model, contemplate different types of potential fields: tangential fields to circumnavigate obstacles, Gaussian potential fields [8].
In addition, we notice the *Vector Field Histogram* algorithm that produces a vector force as output (length, angle), but differently from others it exploits peculiar tools as histograms to represent the distribution of obstacles around

the robot. The algorithm provides for three stages so that the initial 2D histogram is collapsed to a polar one.
Moreover, as well as the classical approaches, novel solutions have been explored by combining genetic algorithms and neural networks (as reported in [6]).
In the end, complete navigation with obstacle avoidance have been experimented by using the *Particle Swarm Optimization* algorithm [4].

Chapter 3

Solution development

3.1 Behaviour design

3.1.1 NFA

The behavior of each robot is modeled using a non-deterministic finite state automata (*NFA*)(see figure 3.1). Such an architecture has been chosen for the robot since its behavior is not trivial and more tasks must be executed at different times. By using a FSA, it is easy to explicitly express activities, conditions and non-deterministic transitions. Moreover, each state can be developed as a separated module, and the most suitable solution can be implemented case by case.
The robot starts in the "resting" state. A non-deterministic transition makes the robot leave the base and start the exploration with a probability $P1$.
When exploring, the robot looks for any close landmark and, if it is the case, it joins the cluster around such landmark with a probability $P2$ and enters in "waiting for cluster to complete" state. Moreover, the robot may quit the exploration task at any moment and return to the base with a probability $P3$. In the latter casuistry, the robot enters in the "returning to base" state.
When a cluster is completed by reaching the required number of robots N, the robot enters in the "reconnaissance" state. Such a state conceptually correspond to the phase during which the robots in the cluster should explore the area near the landmark. Such a task is not executed in this project for simplicity, but few different algorithms have been created to let a set of robots to explore and map a given area.
The robot ends the reconnaissance task after t_1 seconds, and it returns to the base.
When returning to the base, the robot executes a phototaxis task. The robot enters in the "resting" state once it is detected that it successfully returned to

the base.
In the end, the robot may leave the cluster before it is completed with a probability $P4$. In this case, it enters in the "biased exploration" state, that is a state where the robot executes the regular exploration tasks but with inhibited stimuli from landmarks. After t_2 seconds it returns in the normal exploration behaviour. Such a intermediate state is required so that the robot may step away from the landmark (otherwise, even though it is a probabilistic transition, it would join the cluster again in most of the cases). TODO: required?

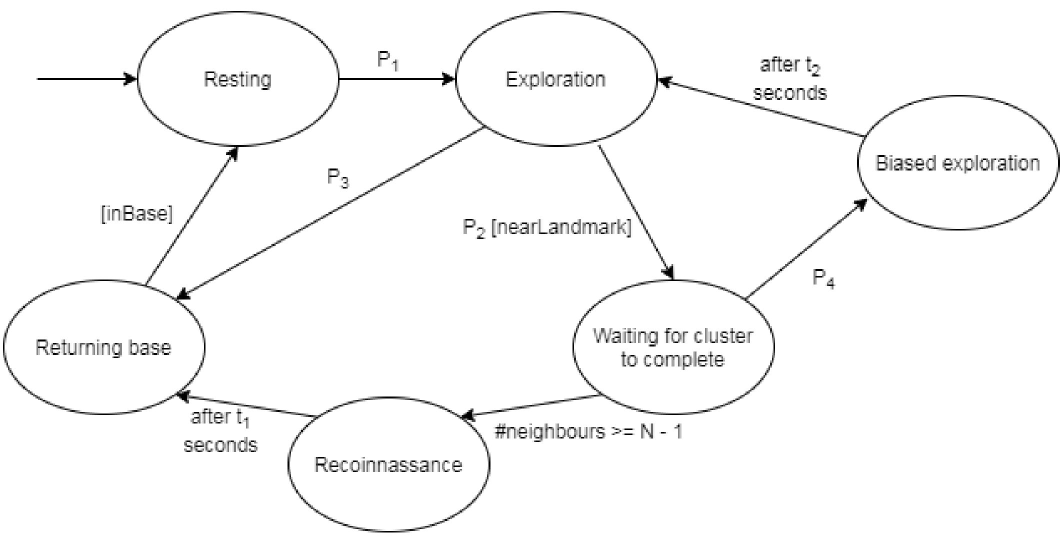

Figure 3.1: *NFA representing the behaviour of the robot. P_i represent a non-deterministic transition that happens with such probability.*

The implementation of the FSA is executed by using a "states" table to store the code be executed in each state. The variable "current_state" represents the current active state, while transition are executed by checking simple"if-then-else" conditions.
In the end, a variable "t" stores the time spent by the robot in the current state.

3.1.2 Motor schema

When a state requires motion, its design has been developed using a motor schema approach and, once the total force suffered by the robot has been calculated, the following formula is used to switch from the translational-angular model to differential one (i.e., find the velocity of each wheel out from the direction and length of the total suffered force):

$$\begin{bmatrix} v_l \\ v_r \end{bmatrix} = \begin{bmatrix} 1 & -L/2 \\ 1 & L/2 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.0}$$

where:
v_l = velocity of the left wheel
v_r = velocity of the right wheel
L = distance between the two wheels
v = translational velocity
ω = angular velocity

3.2 Resting

The robot just checks if a transition to the exploration state should occur, and so no particular architectural style has been adopted for this module. The following function has been used to model the probability:

$$p = tanh((t - Shift)/Patience) + 1 \tag{3.1}$$

The idea is to create a probability that is directly proportional to the time spent in the current state using a non linear relation. Moreover, the probability should grow very slowly at the beginning so that the robot will remain in the resting state for a while.
The function $tanh(x)$ is the basic function used to represent such relation[1] (figure 3.2[2]). The $Shift$ value is constant (500) so we can take the slice of the function having up concavity while dealing with positive time values. The $Patience$ value is a parameter used to tone all values down. In the end, the $+1$ factor let the function have positive values.
Table 3.1 reports few reference values of the function.

[1]A reasonable alternative would be the exponential function, but it requires the tuning of few parameters so that the "tail" of the function take reasonable values

[2]The figure illustrates the regular *tanh* function instead of the one exposed in formula 3.1 since the latter one, despite having a "similar shape", can hardly be graphically shown.

t	p
0	0.013
10	0.015
50	0.022
100	0.036
500	1

Table 3.1: *Few example values of the function used to model dependency with elapsed time using a non linear relation.*

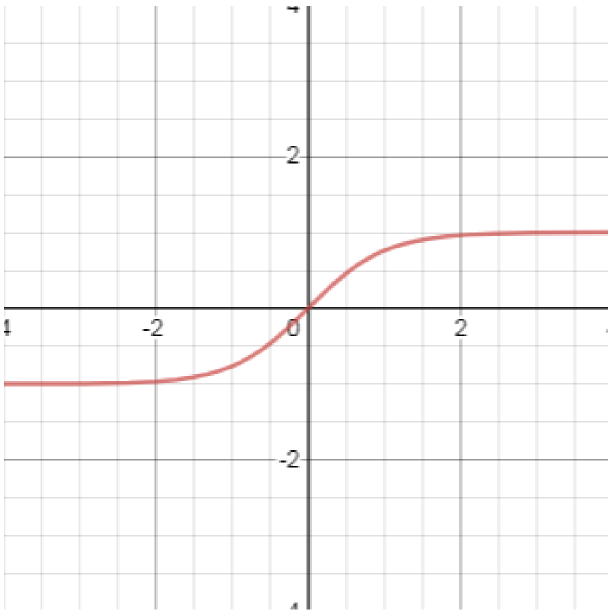

Figure 3.2: *Graph of the tanh function in range [-4;4].*

3.3 Exploration

The robot executes a ballistic motion. A single potential field is created since the obstacle avoidance feature is implicitly given from the random walk itself. The direction of such potential field is randomly chosen each time an obstacle is detected, while its strength is given by a constant value "CRUISE_VELOCITY".

When exploring, the robot has a fixed probability to quit the task. Such parameter has been tuned with a very low vale (0.001) in order to avoid too much segmented behaviours where the robots keep on fluctuating just between the two states of "exploration" and "returning to base".

3.4 Clustering

A robot may join/create a cluster when close to a landmark with a probability described by the following function:

$$p = \frac{(tanh((t - Shift)/Patience) + 1) * tanh(n/NeighborInfluenceLimiter)}{(NeighborInfluenceLimiter * PatienceEnhancer)} \tag{3.2}$$

The probability is proportional to number of robots already in the cluster (n) and to the time spent exploring (t) both. The same formula in 3.1 has been used the time dependence term. The parameter $NeighborInfluenceLimiter$, as the name hints, limits the influence of number of neighbors in the final result.
The denominator is a normalization term so reasonable values can be obtained, and it can be tuned by the parameter $PatienceEnhancer$ that makes the robot wait longer as it is increased.
Figure 3.3 reports the graph of the function in formula 3.2.

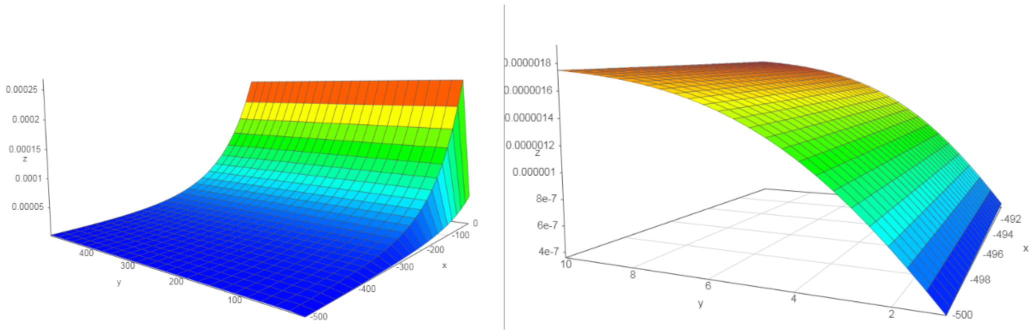

Figure 3.3: *x-axis = time, y-axis = number of neighbors, z-axis = probability. Graph of the plot in domain [-500;0], [1;500] showing the total trend of the function (left) so the time relation can be shown well. On the right, a zoom in in range [-500; -492], [1:10] is executed to show the dependence upon neighbors (indeed, the number of robots in the cluster will never be about the previous order of magnitude* (10^2) *, and so low values, that cannot be appreciated in the left graph, are the the ones of interest.*

Similarly, the probability for a robot to leave a cluster before it is completed is modeled using the following function:

$$p = \frac{(tanh((t - Shift)/Patience) + 1)/tanh(n/NeighborInfluenceLimiter)}{(NeighborInfluenceLimiter * PatienceEnhancer)} \tag{3.3}$$

Formula 3.2 and 3.3 are exactly the same function except for a sign (division instead of multiplication of the two terms that model relation with time and number of neighbors). Indeed, the robot increases its probability to leave as time passes (direct proportionality) but, at the same time, its probability to leave is decreased as the number of perceived neighbors increases (inverse proportionality)(the more robots, the more probability to complete the cluster soon is the underlying reason that may lay under such behavior).
Figure 3.4 reports the graph of the function in 3.3.

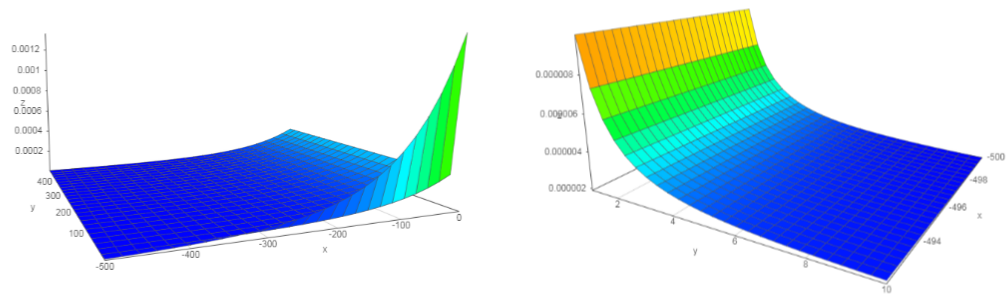

Figure 3.4: *Similal reasonings of figure 3.3. The probability slowly increases as time passes (x-axis) and rapridly decreases as numbe of neighbors increases (y-axis).*

3.5 Returning to base

This state must mixes phototaxis and obstacle avoidance, and so a motor schema approach has been used to easily blend such two tasks.

The phototaxis capability is created by using an attractive potential field produced by the light source. Hence, the resulting force, whose direction guides to robot toward the light source, has a value inversely proportional to the perceived light.

The obstacle avoidance capability is created by using a tangential field around any obstacle so the robot can circumnavigate boxes or other robots. By exploiting proximity sensors, a perpendicular force is created($\pi/2$ from the sensor orientation, and proportional to the proximity). A possible issue is given from the boundary walls that may create local minima in some rare cases. However, such problem may be fixed by introducing a third potential field: a perpendicular one coming from the walls. Moreover, when travelling to base, the robot should not slip into any wall in most of the cases, and so the issue is not of primary importance for the time being.

3.6 Landmark

Landmarks have been modeled using robots too[3].
Landmarks just have two states: not explored, explored.
A landmark starts in the “not explored” state and continuously notifies all the nearby robots (see section 3.7). A permanent switch to the “explored” state is executed once a cluster is created nearby. In this state the landmark do not send notifications any more, and so it becomes a regular object to be just avoided.

3.7 Communication

Robots of the swarm communicate with landmarks and with other robots by using the range and bearing functionality. A total of three communication channel have been used, that is:

1 **Landmark signal**: used by robots to perceive a nearby landmark.

2 **Landmark explored notification**: used to notify a landmark that it should change its state to the “explored” one.

3 **Neighbour signal**: used to count the number of sensed neighbors from the robots.

The communication range is fixed for channel [1] and [2], while it varies case by case for [3]. Indeed, notice that each robot will have its own distance under which it may join the cluster. Such value is randomly selected in a fixed range, and it is required to prevent all robots stopping at the same distance from the landmark (fact that makes the creation of big clusters difficult).

[3]Notice that it is not a constraint nor a simplification. Any other environmental bias (black spots etc.) would also work. Robots have been chosen so that a nice graphic with landmarks status can be seen (red/green leds) and for convenience (e.g., logging, easy to move them instead of creating lots of arenas etc.)

Chapter 4

Performance evaluation

4.1 Evaluation metrics

The system is evaluated by checking how many landmarks are explored as time passes. A simulation is concluded when time reaches a maximum allowed value (3000). For example, if we have the following values:

$$[100, 400, 800, 3000]$$

The first landmark has been explored at time 100, etc. The last one has not been explored during the simulation, and so its value is set to 3000. This way, we can model the trend of the system using a time series where each time t is associated to the number of landmarks that had already been explored in such time.
Figure 4.1 reports a clarifier example of a time series.

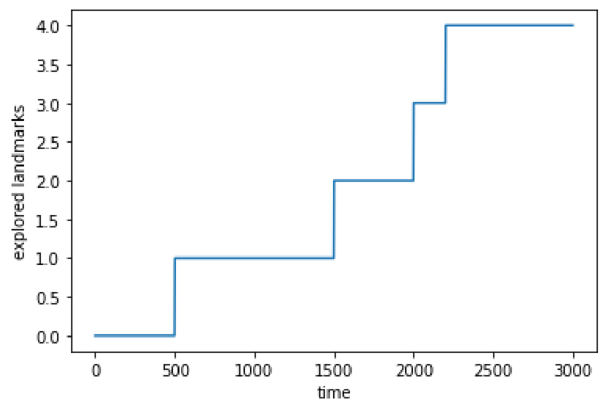

Figure 4.1: *Example time series with exploration times [500, 1500, 2000, 2300].*

Moreover, we define a measure of performance (*goodness*) to estimate how well the system behaves so that we can compare different combinations of parameters in a quantitative way. To do this, we consider the area under the time series (i.e., the integral of the function). Such area can assume values in range [0; 12000] (0 if no landmark has been explored during the simulation, 12000 is a theoretical max value given by $simulation\_length * n\_landmarks$). The main idea is that the greater the area, the sooner landmarks had been explored. Notice that, each value is then normalized in [0;1] for convenience.
Figure 4.2 reports a clarifier example of lower and upper bounds.

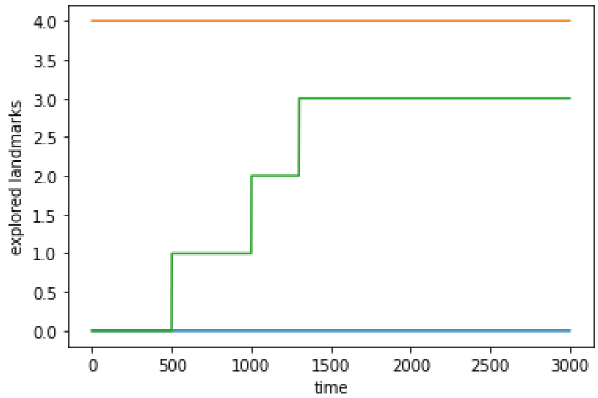

Figure 4.2: *Example time series with exploration times [500, 1000, 1300, 3000] (green). Upper and bound bounds in orange and blue respectively. The integral of the blue function is simply 0, while the one of the orange is given by 3000 * 4.*

Note: Notice that, in order to compare solutions with different expected cluster size, such measure should be refined since the latter parameter influences the exploration times in a non negligible manner. To overcome such issue, a penalty factor may be added to cluster with lower sizes.

4.2 Parameters definition

When executing performance evaluation stage, we choose just a subset of parameters and configurations the system can be tuned upon since the total number of combination is exponential, and so it would require too much time to explore them all using a grid search technique. In particular, the following aspects are the one considered as constants:

- Size of the arena: it is assumed to use a 4x4 meters arena.

- Position of the base: it is assumed to locate the base in the center of the arena.
- Few controller parameters: all non-deterministic transition parameters, gain factor in motor schema modules etc.
- Type of exploration: it may be interesting to test the system using different exploration strategies aside from the ballistic motion.
- Number of landmarks: it is assumed to have four landmarks

The system is evaluated by changing the following parameters:

- Number of employed robots: 20, 30, 40.
- Expected cluster size: 1, 3, 5.
- Number of obstacles scattered around the arena: 0, 10, 20
- Position of the landmarks

The first analyzed setup of the arena is reported in figure 4.3. The four landmarks are all placed on a same side of the arena and they are aligned so that a straight line parallel to one edge of the arena is created. Then, aside from this basic arena configuration, further evaluations will be made using different deployments of the landmarks (at the four cardinal points, at the four corners, random distribution etc.).

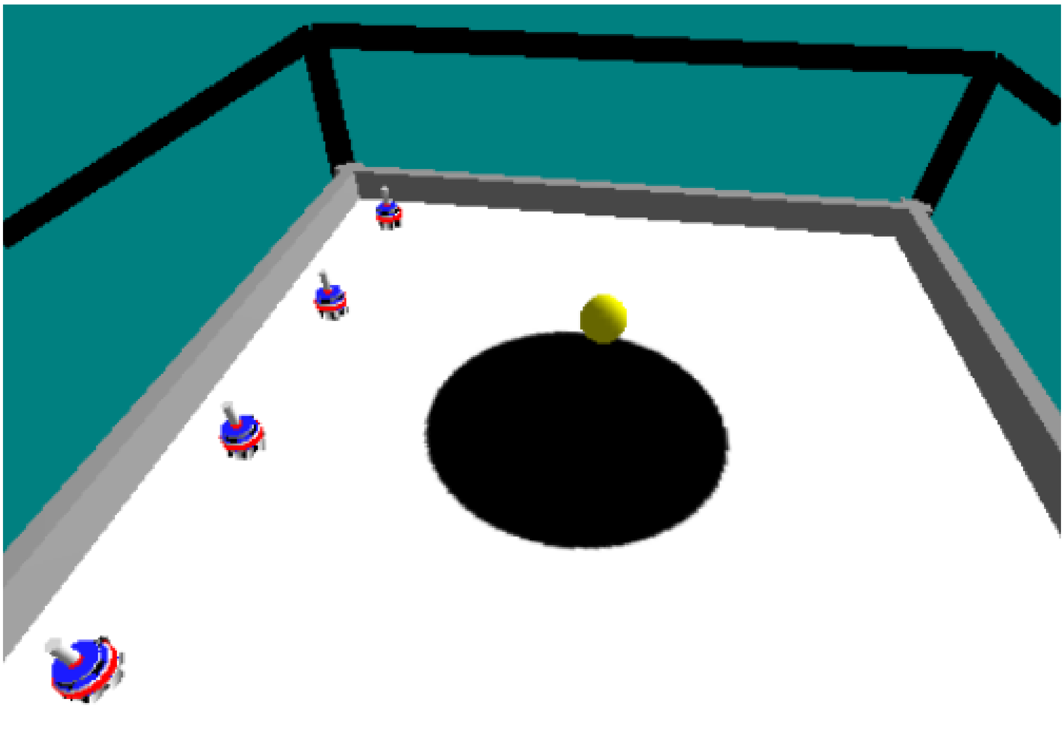

Figure 4.3: *Setup of the arena with the four landmarks located on a straight line parallel to one edge of the arena.*

4.3 Basic arena

When considering the basic setup of the arena, table 4.1 reports the *minimum guaranteed level of service* (i.e. worst values among all the executed runs) and the goodness of each casuistry. Figure 4.4 shows the comparison of the cases reported in table 4.1.

Number of robots

An increase in the number of robots improves the overall performance values in general. Indeed, we can notice that the system can not reach the goal using cluster size three just when using twenty robots. Moreover, we obtain the best results when using the maximum number of robots (forty). However, when dealing with an higher number of robots than forty and if keeping the same size of the arena, they may disturb each other, and so further evaluations should be made and no general rule can be inferred.

Besides, notice that the minimum guaranteed level of service can be tricky.

Indeed, by looking to raw performance data (available in attached files), it is confirmed that better average performance values are obtained despite a same minimum guaranteed level of service for cluster size five.

Cluster size

As expected, exploration times increase proportionally to the cluster size. Indeed, aside from the trivial fact that more robots should gather, it is seen that the gathering operation itself is harder because of spatial constraints (harder for a robots to get close to a landmark if few robots are already there).
An expected size greater than three makes the creation of a cluster harder, and often few landmarks remain unexplored at the ending of the simulation. As observed from empirical tests, such issue is mainly due to spatial problems constraints (difficulty to actually aggregate since there is not enough space). A greater time limit has been probed, and it emerged to be that clusters will be created anyway eventually. Moreover, few tests have been executed on wider arenas (e.g. 6x6, 8x8 meters) and, despite higher times due to the greater distances to travel, cluster creation resulted to be simplified.

Number of obstacles

The insertion of obstacles is a key factor since it emerges a great difference among the study cases. Indeed, the performance will decrease independently from the adopted cluster size, and this is quite obvious since the boxes make exploration and aggregation harder (in some observed cases even impossible by leaving not enough empty space to create a proper cluster). However, notice that the gap between thirty and forty robots is filled up when using cluster size three or greater (rather, slightly better performance using thirty robots, 40 robots + 20 obstacle + 4 landmarks may create too harsh conditions for the navigation).

Miscellaneous

- As expected, the landmarks on the sides are often the last to be explored because their harder to reach (and it is harder to create cluster around them too).

Robots	Cluster size	Obstacles	Time				Goodness
20	1	0	161	258	343	1081	0.84
20	1	10	300	329	1705	1931	0.64
20	1	20	266	370	863	2771	0.64
20	3	0	559	1231	2730	3000	0.37
20	3	10	3000	3000	3000	3000	0.0
20	3	20	3000	3000	3000	3000	0.0
20	5	0	3000	3000	3000	3000	0.0
20	5	10	3000	3000	3000	3000	0.0
20	5	20	3000	3000	3000	3000	0.0
30	1	0	115	372	414	1032	0.83
30	1	10	234	418	548	2112	0.72
30	1	20	317	623	994	3000	0.58
30	3	0	869	1165	2401	3000	0.33
30	3	10	1449	1962	2981	3000	0.21
30	3	20	1461	2667	2990	3000	0.15
30	5	0	3000	3000	3000	3000	0.0
30	5	10	3000	3000	3000	3000	0.0
30	5	20	3000	3000	3000	3000	0.0
40	1	0	175	236	367	862	0.86
40	1	10	268	410	420	1022	0.82
40	1	20	307	580	1018	2069	0.66
40	3	0	833	834	970	2566	0.53
40	3	10	1542	1633	2394	2453	0.33
40	3	20	1534	2815	3000	3000	0.13
40	5	0	3000	3000	3000	3000	0.0
40	5	10	3000	3000	3000	3000	0.0
40	5	20	3000	3000	3000	3000	0.0

Table 4.1: *Evaluation of the system on 27 proposed combinations. For each casuistry, it is reported the moment of exploration of each landmark. Every value is the worst value among the registered ones. Moreover, the goodness of each combination is reported too.*

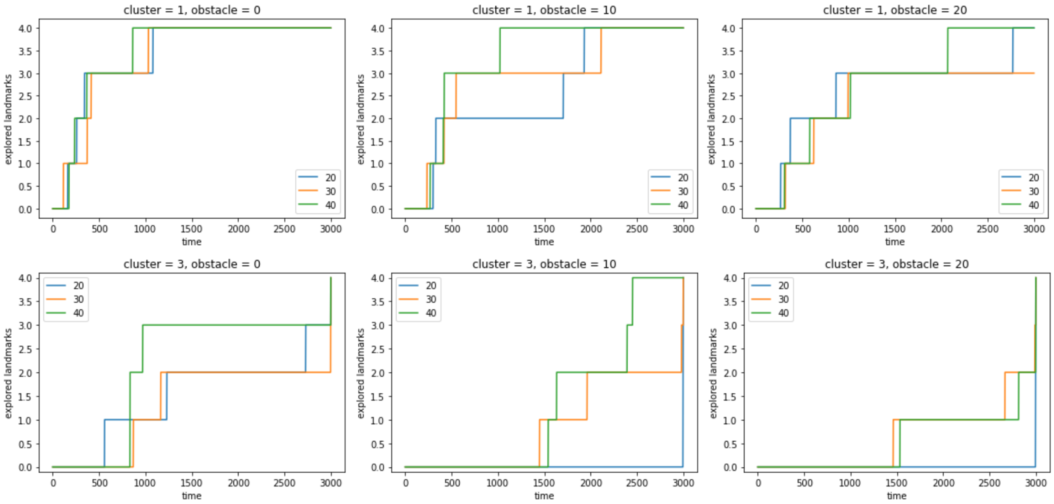

Figure 4.4: *Performance on each casuistry using cluster size one and three for minimum guaranteed level of service (cluster size five is omitted since its values would always have been zero.*

4.4 Further case studies

Now, the system is probed using a different positioning of the landmarks in order to check its robustness on different cases. In particular, while fixing the cluster size to three and the number of obstacle to ten, the system is evaluated by choosing random locations for the landmarks. Configurations are dropped when the landmarks are spawned in the base or at least two of them are too close (because of possible interferences when counting neighbors through the range and bearing system).
Table 4.2 reports the observed results. As we can notice, the positioning of the landmarks is a key factor since it greatly influences the results. Indeed, the goodness assumes very high/low values depending on very lick/unlucky configurations, and so the variance between the exploration times is higher than the one observed when just using the basic arena.
Moreover, as expected, the system behaves better when employing the maximum number of robots (forty).

When comparing such results to the ones obtained on the basic arena (figure 4.5), it emerges that better results are generally obtained on the random ones: this is probably due to the fact that, since in the basic arena all landmarks are placed on the same side, robots that start exploring the opposite part will not find any landmark for a while, and so they are vain for the first part of the simulation. Obviously, such considerations can not be made for random deployment of the landmarks in general, and so better performance are obtained since all robots take part in the creation of the clusters. In the end, notice that the clustering behavior emerges independently from the number of robots when using random arenas, while it is not true when using twenty robots on the basic arena (at least in the specified time boundaries).

Note: Such evaluations may be tricky (or even misleading) since more attempts on further cases should be made in order to apply any statical technique to validate the results.

Robots	Cluster size	Obstacles	Time				Goodness
			231	886	2388	3000	0.45
			396	640	3000	3000	0.41
20	3	10	112	975	2645	2871	0.45
			454	1112	1921	2678	0.48
			432	1001	2319	2992	0.43
			325	344	1481	1684	0.68
			101	863	907	3000	0.59
30	3	10	494	653	3000	3000	0.40
			575	1166	3000	3000	0.35
			1310	1667	1718	3000	0.40
			524	800	815	1060	0.73
			183	942	2445	2771	0.47
40	3	10	529	854	1926	2714	0.49
			960	1278	2167	3000	0.38
			102	672	719	3000	0.62

Table 4.2: *Evaluation of the system using random positioning of the landmarks. For each casuistry, it is reported the moment of exploration of each landmark for a random setup of the landamrks. Moreover, the goodness of each combination is reported too.*

Figure 4.5: *Comparison of the performance obtained on basic arena and on random ones.*

Chapter 5

Conclusions

The system has been developed by mixing different well-known techniques such as finite state automata and motor schema. Moreover, the most common swarm intelligence principles have been applied too (local sensing realized through multi-channel range and bearing communication, non-deterministic state transitions with positive/negative feedbacks). The realization of few parts has been made by following the state of the art for such tasks by implementing common sense techniques (exploration for swarm robotics through random walk, obstacle avoidance though well-known potential fields etc.).
In the end, the system has been evaluated upon multiple configuration of the arena and controller both by changing parameters such as the position of the landmarks, the expected cluster size and number of obstacles. It emerged that the position of landmarks and obstacles is a key factor, while the number of robot is decisive for emerging clustering behaviour.

Further developments may include the implementation of any different exploration strategy aside form the ballistic motion and a refinement of the modules based upon the motor schema approach by inserting a further potential field so walls of the arena are considered too. Moreover, a complete testing of the system on multiple different arenas and controller configurations may be executed so that enough data is available to apply any statistical test (or tuning strategy based on it such as the *F-race* algorithm).

Bibliography

[1] https://link.springer.com/chapter/10.1007/978-3-030-25332-5_19

[2] https://www.hindawi.com/journals/jr/2019/6914212/

[3] https://arxiv.org/ftp/arxiv/papers/1306/1306.1144.pdf

[4] https://www.researchgate.net/profile/Dr_Anish_Pandey/publication/317101750_Mobile_Robot_Navigation_and_Obstacle_Avoidance_Techniques_A_Review/links/59266dad458515e3d45393b3/Mobile-Robot-Navigation-and-Obstacle-Avoidance-Techniques-A-Review.pdf

[5] https://www.hindawi.com/journals/jat/2018/5041401/

[6] http://anderslyhnechristensen.com/pubs/alchristensen_alifex_holeavoidanceandphototaxis_2006.pdf

[7] https://www.hindawi.com/journals/jat/2018/5041401/

[8] https://www.hindawi.com/journals/jat/2018/5041401/

[9] https://www.sciencedirect.com/science/article/pii/S0004370220300047

[10] https://www.sciencedirect.com/science/article/pii/S0925231215010486

[11] https://link.springer.com/chapter/10.1007/978-3-319-21407-8_2

[12] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.310.7632&rep=rep1&type=pdf

[13] https://www.sciencedirect.com/science/article/pii/S0004370220300047#br0140