Università degli Studi di Napoli Federico II
Ph.D. Program in
Information Technology and Electrical Engineering
XXXV Cycle

Thesis for the Degree of Doctor of Philosophy

Multi-Robot Distributed Strategies for Priority-Based Sanitization of Railway Stations

by
Fabrizio Tavano

Advisor: Prof. Vincenzo Lippiello

Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

to my Father

Multi-Robot Distributed Strategies for Priority-Based Sanitization of Railway Stations

Ph.D. Thesis presented
for the fulfillment of the Degree of Doctor of Philosophy
in Information Technology and Electrical Engineering
by

Fabrizio Tavano

October 2023

Approved as to style and content by

Prof. Vincenzo Lippiello, Advisor

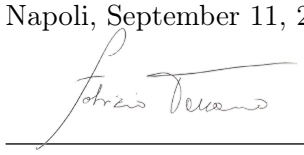
Università degli Studi di Napoli Federico II
Ph.D. Program in Information Technology and Electrical Engineering
XXXV cycle - Chairman: Prof. Stefano Russo

http://itee.dieti.unina.it

Candidate's declaration

I hereby declare that this thesis submitted to obtain the academic degree of Philosophiæ Doctor (Ph.D.) in Information Technology and Electrical Engineering is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.
Parts of this dissertation have been published in international journals and/or conference articles (see list of the author's publications at the end of the thesis).

Napoli, September 11, 2023



Fabrizio Tavano

Abstract

Recent studies have highlighted that shared areas of the railway stations may be locations where there may be contagion by viruses and bacteria during the periods of a pandemic like in the case caused by SARS-CoV-2 diffusion. The propagation of SARS-CoV-2 disease particularly damaged the railway sector because the transportation demand had registered a drastic reduction. People prefer to travel only if strictly necessary, using private cars in alternative to public services such as railway transportation. In this Thesis, answering the Infrastructure Manager Rete Ferroviaria Italiana's request, we propose sanitization strategies that coordinate a team of robots in a dynamic railway station without interrupting the preexisting human activities such as transportation services, catering services, shopping, and ticketing. Since every important Italian station is equipped with a WiFi Meraki Cisco System Network, our aim is to exploit such infrastructure to monitor the positions of mobile devices (tablets and phones) and to evaluate the most crowded areas of the station to be sanitized. Specifically, our approach is to define a heatmap whose colored zones indicate the presence of contamination (prioritized zones) caused by the aggregation of visitors, which can be used by robots as guidance during sanitization activities. In Chapter 3, we propose a multi-robot approach to sanitize railway stations based on a distributed Deep Q-Learning technique. A team of cleaning robots - each endowed with a robot-specific convolutional neural network - learns how to effectively cooperate and sanitize the station's areas according to the associated priorities. In Chapter 4 we extend the previous framework allowing to define teams of robots having different sanitizing strategies/capabilities. In Chapter 5 we illustrate a distributed framework, where a centralized server uses the Hierarchical Mixed Integer Linear Programming to coordinate the robots assigning different zones where the cleaning has higher priority; thanks to the MPC-MILP

approach, we use historical data about the distribution of people and the knowledge about the transportation service of the station, to predict the future dynamic evolution of the position of people in the environment and the spreading of the contaminants. In Chapter 6, we propose a multi-robot online sanitization strategy that combines the Bioinspired Artificial Cockroach Colony Strategy with the 2-type Fuzzy Logic to coordinate together a team of robot sanitizers. We tested our solution considering real data collected by the WiFi network of the main Italian railway station, Roma Termini shared by Rete Ferroviaria Italiana S.p.A. We compared our results together with other methods conventionally proposed for sanitization, applied to the same scenario. The approaches described in this Thesis may also be applied in every indoor public location as shopping centers, restaurants, and industrial sheds, if WiFi Service is available with visitors' positioning information, with a correct number of robots selected considering the total surface of the environment.

Keywords: Deep Reinforcement Learning, MPC-MILP, Cockroach Colony Strategy, 2-type Fuzzy Logic, Distributed Multi-Robot Systems, Sanitization.

Sintesi in lingua italiana

Studi recenti hanno messo in evidenza che aree condivise delle stazioni ferroviarie posson essere luoghi dove sono possibili contagi causati da virus e batteri durante i periodi di pandemia come quello causato dalla diffusione del SARS-CoV-2. La propagazione della malattia SARS-CoV-2 ha particolarmente danneggiato il settore ferroviario perché la domanda di mobilità di trasporto ha registrato una drastica riduzione. La gente ha preferito viaggiare solo se strettamente necessario, usando soprattutto mezzi privati in alternativa ai servizi di trasporto ferroviario. In questa Tesi, rispondendo ad una richiesta del Gestore dell'Infrastruttura Rete Ferroviaria Italiana, si propongono differenti strategie di sanitizzazione per coordinare un team di robot in un ambiente dinamico come una stazione ferroviaria, senza interromperne i servizi di trasporto, la ristorazione, lo shopping, la vendita dei biglietti, ed in presenza dei visitatori. Dal momento che ogni importante stazione ferroviaria è dotata di una infrastruttura di rete WiFi Cisco System, abbiamo deciso di utilizzare tale rete per monitorare le posizioni dei dispositivi mobili (tablet e cellulari) e di valutare quali sono le aree della stazione più affollate che necessitano di sanitizzazione con maggiore priorità. Nello specifico, il nostro approccio consiste nel costruire un heatmap in cui le zone colorate indicano la presenza di zone con più livelli di contaminazione (zone prioritarie) causate dall'aggregazione di visitatori. le zone prioritarie possono essere usate dai robot come guida durante l'attività di sanitizzazione. Nel Capitolo 3, si propone un approccio Multi-robot per la sanitizzazione basato su una tecnica distribuita di Deep Q-Learning. Un team di cleaning robot, ciascuno dotato di una propria rete neurale convoluzionale, impara a cooperare e sanitizzare le aree della stazione considerando le relative priorità associate. Nel Capitolo 4 abbiamo esteso lo studio precedente aggiungendo al framework funzionalità per definire team di robots aventi differenti caratteristiche e strategie

di sanitizzazione. Nel Capitolo 5 si presenta un framework distribuito in cui un server centrale applica il metodo Hierarchical Mixed Integer Linear Programming per assegnare zone differenti a ciascun robot dove le priorità sono maggiori. Grazie all'approccio MPC-MILP, abbiamo usato dati storici sulla distribuzione delle persone, nonchè la conoscenza delle caratteristiche del servizio di trasporto nella stazione per predire l'evoluzione dinamica delle posizioni delle persone e la relativa diffusione della contaminazione. Nel Capitolo 6 proponiamo un metodo di sanitizzazione on-line, che combina la tecnica Artificial Cockroach Colony Strategy con la tecnica 2-type Fuzzy Logic per coordinare un team di robot sanitizzatori. Abbiamo testato le nostre soluzioni impiegando dati reali raccolti dalla rete WiFi della principale stazione italiana Roma Termini, condivisi da Rete ferroviaria Italiana S.p.A. Abbiamo inoltre confrontato le performance dei metodi proposti con altre tecniche comunemente proposte in letteratura per la sanitizzazione, applicandoli al nostro scenario. Gli approcci presentati in questa tesi possono essere applicati in ogni luogo pubblico, come centri commerciali, ristoranti e capannoni industriali, se è disponibile un Servizio WiFi ed un adeguato numero di robot.

Parole chiave: Deep Reinforcement Learning, MPC-MILP, Cockroach Colony Strategy, 2-type Fuzzy Logic, Distributed Multi-Robot Systems, Sanitization.

Contents

Acknowledgements

The research leading to these results has been supported by the Italian Infrastructure Manager Rete Ferroviaria Italiana S.p.A, and by the HARMONY project (Horizon 2020 Grant Agreement No. 101017008).

List of Acronyms

The following acronyms are used throughout the thesis.

DQN	Deep-Q Network
MARL	Multi-Agent Reinforcement Learning
APF	Artificial Potential Fields
MILP	Mixed Integer Linear Programming
MPC	Model Predictive Control
ACS	Artificial Cockroach Strategy
RAM	Random Access Memory
ReLu	rectified linear unit
ML	Machine Learning
HMILP	Hieralchical Mixed Integer Linear Programming
DTDE	Decentralized Training Decentralized Execution
RQ	Research Question

List of Figures

List of Tables

Chapter 1

Introduction

Modern societies are increasingly open and connected, the demand for transport is continuously increasing [18] and the high volume and frequency of passengers moving between cities is an evident cause for rapid spread of diseases across countries [102]. In this regard, the role of railway stations is crucial. Stations are not only nodes of the railway network where passengers are in transit, but also locations for shops, restaurants, places for recreation and aggregation [6, 75]. Passengers gathering in the halls and platforms of the stations, eating at restaurants, and boarding the trains facilitate the transmission of diseases [101]. For this reason, the pandemic caused by the SARS-CoV-2 has spawned a crisis that has affected the railway sector in a significant way [92], for example, by inducing people to prefer cars instead of trains [24]. It is then strategic for the infrastructure managers (such as the Italian Rete Ferroviaria Italiana) to deploy adequate and modern tools to prevent future contagion inside railway stations [76]. In this scenario, disinfectant robot technologies can support in fighting pandemics [93] by reducing the number of people involved in the cleaning process and by enhancing the sterilization performance.

In this direction, we propose an approach to multi-robot sanitization suitable for teams of robots capable of cleaning large human-populated railway stations [66] without interruption of any public service. Specifically, we present distributed methods to generate effective cleaning strategies that drive a set of sanitizing robots towards the most critical regions of the station. Such regions are prioritized by estimating the most populated

areas of the station to be sanitized. For this purpose, we rely on the WiFi infrastructure, which is often available in public environments, to retrieve the position of the smartphones [82] to get an approximated assessment of the distribution and concentration of the crowd. Such estimated distribution of people in the railway station areas is then used to specify a heatmap of prioritized areas to be sanitized, which is dynamically updated and provided to each of the robots to enable them to adapt their cooperative cleaning behaviors continuously. It should be noted that cleaning railway stations in the presence of humans represents a very challenging real-world domain for multi-robot coordination, not only because of the large environment to be effectively covered by multiple agents, but also for the continuous variation of the people distribution that requires dynamic adaptation of the target areas to be reached. In this scenario, we propose in Chapter 3 to deploy a distributed Deep Q-Learning approach, where each robotic agent, endowed with a specific Deep Q-Network (DQN), learns how to adapt its cleaning behavior with respect to the shared estimated heatmap. For each robot, given the current heatmap and position, the generated strategy provides the next adjacent area to sanitize in order to maximize the overall team reward. The peculiarity and originality of the approach rely on the exploitation of the shared dynamic heatmap in combination with a distributed deep learning algorithm, where each agent can independently learn how to effectively sanitize a large and populated environment in cooperation with other agents. While the distributed algorithm permits scaling with the number of robots, the shared heatmap accounts for dynamically updated priorities in large populated environments, which are rarely considered in multi-agent sanitizing frameworks. In our setting, such priorities are associated with the dynamics of contaminated areas, which is modeled by a Gaussian convolution law. In Chapter 4 we investigate on the behavior of a distributed Deep Q-Learning approach, there are more typologies of teams of robots with different capabilities of cleaning and speed. We compared this solution with the homogeneous case. In Chapter 5 we propose a deterministic approach where the sanitization's strategy depends on the participation of the Server that assigns every robot with different heatmap's zones with high priority thanks to the Hierarchical MILP-MPC approach, and then every robot independently builds its paths to the zones with the Artificial Potential Fields technique, clean-

ing the peaks of the priorities they meet along the path. In Chapter 6, inspired by the social behavior of the Cockroaches, we described a new approach where the robots apply and Cockroach Colony Strategy to the cleaning. The WiFi server participates to the cleaning process sending to every robots the heatmap and a FeromonesMap where the robots find respectively the positions of the contaminations and the feromones signals deposited by the members of the team. Moreover, the Server proposes as a destination a different heatmap subarea for each robot thanks to the 2-type Fuzzy Logic combined with the taboo search technique. The robots build their paths using 2-type Fuzzy Logic method considering at every step the presence of the concentration of priority and the pheromones that they may find along the route, their own distance from other members of the team, and their distance from the destination zone proposed by the WiFi Server.

Notice that in this Thesis we focus on the decision and coordination problem (i.e., areas to be cleaned by the robots), while we assume that each cleaning robot is endowed with a suitable navigation system (localization, obstacle avoidance, path and motion planning systems, etc.) to safely reach the areas established by the cleaning strategy. We investigate the system at work in several realistic case studies defined in the context of the Roma Termini railway station. In this scenario, we evaluate the system performance by exploiting both simulated and real data about the people distribution inside the station. The collected results show that the proposed approach is feasible in real-world scenarios, competitive with respect to alternative methods, and scalable with respect to the number of sanitizing agents and the crowding of the station. The main contributions of this work can be summarized as follows:

- We propose a decentralized and scalable DQN framework, where multiple mobile robots independently learn to cooperate for the execution of cleaning tasks in crowded indoor environments. We introduce and discuss several case studies in a real-world challenging environment provided by the Roma Termini railway station. In this scenario, we show how the proposed decentralized DQN framework responds to online priority changes and how it scales with respect to the number of robots (from 2 to 8) and the crowding of the station.

- We extended the DQN approach to the heterogeneous case, comparing the relative performances with the homogeneous one.
- We propose a new distributed approach based on the collaboration of a MILP-MPC Server with robots performing the Artificial Potential Fields technique to sanitize a large environment.
- We propose a new distributed bioinspired Cockroack Colony Strategy combined with 2-type Fuzzy Logic for the sanitization of dynamic environments.
- The proposed methods rely on the station's WiFi infrastructure to monitor the distribution of people in the environment and estimate the diffusion of contaminants, thus allowing to dynamically prioritize the areas to be sanitized.
- We discuss and compare all the proposed methods at work with a real dataset in which the distribution of people is retrieved from a one-day data recording provided by the WiFi Network of Roma Termini, shared by Rete Ferroviaria Italiana S.p.A.
- The performances of the proposed systems is also compared with respect to standard coverage path planning techniques.

The rest of the Thesis is structured as follows. In Chapter 2, we discuss literature related to multi-robot cleaning/sanitizing and area covering strategies. In Chapter 3, we describe the architecture of the proposed DQN framework. In Chapter 4, we present the extended version of the DQN approach considering heterogeneous teams of robots. In Chapter 5, it is illustrated the MILP-MPC Strategy combined with the Artificial Potential Field Method. In Chapter 6, we propose a new bio-inspired Cockroach Colony Strategy with 2-type Fuzzy logic method and we compare the performances of all our discussed approaches together with 2 conventional coverage path planning proposed for the sanitization found in the literature. Finally, Chapter 7 concludes the Thesis and outlines future research directions.

Chapter 2

Background and Related Work

Perception is never purely in the present - it has to draw on experience of the past;(...).We all have detailed memories of how things have previously looked and sounded, and these memories are recalled and admixed with every new perception.

Oliver Sacks, Musicophilia: Tales of Music and the Brain

In this Thesis, we address the issue of sanitizing highly dynamic indoor environments with different adaptive approaches. We use both statistical and deterministic model-based methods capable of responding to a context that evolves over time, such as that of the railway station. In fact, in the railway station, sanitization becomes more urgent in places where there are aggregations of people, and the contamination of the areas increases with the increase in the number of visitors and following their movements. For this reason, we decide to test multiple sanitization methodologies and compare them in a real scenario using one day real data shared by Rete Ferroviaria Italiana S.p.A. In this Chapter, we describe, for each studied approach, the background and related works from which our methods were born. In particular, Section 2.1 describes typical methods proposed for sanitization and cleaning based on Coverage Path Planning (CPP) techniques. Section 2.2 describes the related works for the Deep Q-Learning technique, while Section 2.3 describes the background relating to its extension version

through the use of heterogeneous robot teams. Section 2.4 illustrates the works that inspired us towards the application of the Hierarchical MILP-MPC method, while Section 2.5 shows the studies that led us towards the adoption of a strategy based on the Cockroach Colony Method and the use of 2-type Fuzzy Logic for the coordination of the robot team in its sanitization task.

2.1 Coverage path planning

Several approaches have been proposed in the literature to design multi-robot strategies for cleaning activities. Different frameworks rely on coverage path planning (CPP) methods to reach and sanitize a specific area. Random walk methods [19] have been investigated to randomly and rapidly explore free regions exploiting multiple robotic agents, but these techniques are usually not efficient since robot overlapping and path repetitions are frequent. Other methods are based on fixed-shaped paths (spirals, rectangles, etc.), where each robot is assigned to a specific area of the environment [68, 71, 62, 54, 50, 53, 52, 67, 86]. These approaches are effective in providing a continuous cleaning service that maximizes the coverage and minimizes the idleness of the agents, on the other hand, they are not adaptive and do not provide dynamic prioritization mechanisms with respect to changing environments.

In contrast, we are interested in generating sanitization strategies that take into account the dynamic evolution of the cleaning priorities. In this regard, we estimate the spread of contaminants by monitoring the distribution of people in the station to guide the robots towards sanitizing the most contaminated areas. Related works on CPP consider priorities during patrolling or surveillance tasks.

For instance, in [58], the authors focus on the task of patrolling a given environment exploiting multiple agents and ensuring that static prioritized locations are visited within a predefined time period. Similarly, in [90, 65, 74], a surveillance task is addressed as a periodical revisit problem of a discrete set of sites associated with static priorities or fixed visitation frequencies. These approaches often consider static priorities and graph-based representations of the environments with a limited number of nodes. In our work, in contrast, we consider a distribution of priorities that is

dynamic on the environment's surface, with unpredictable variations following the movements of people. Our environment is represented as a very large gridmap where every cell represents a location point. The priorities are represented with real values assigned to each cell of the gridmap. The real value associated with the cell represents the degree of contamination of the relative location in the environment. In addition to the above methods, other studies exploit the Voronoi tessellation based coverage control (VTCC) proposed by Jorge Cortes et al. [26]. The technique first divides the coverage area into subspaces called Voronoi cells. Each subspace is assigned to a robot, and each robot converges to the centroidal position of the subarea by using Lloyd's algorithm [56], considering a cost function of the system that decreases as the distance between the robot and the centroid decreases. The robots, following the gradient descent control law, move to the center of mass of the Voronoi Tessellation, which balances a distributed density function between the Voronoi segments [25]. These studies are commonly proposed in the literature in applications in which the robot team moves in the environment, attempting to maximize the part of the map that is visible, such as in the case of surveillance of an area of interest via mobile robots with sensors [20, 78, 57, 7]. As for cleaning applications, in [48, 49], a distributed slipform control scheme is proposed for oil spills at sea, targeting thick layers, using Voronoi tessellation to achieve coverage of the oil spill area, navigating the stain to the optimal centroid location, and spraying the dispersant. In this case, the distributed density function is represented by one Gaussian bell function, with one maximum peak at the center of its shape. Here, the robots move in the environment with the gradient descent law to the optimal centroid location, which coincides with the center of mass of the Voronoi cell. The robots converge all together in points with higher values, close to the maximum peak of the Gaussian bell, surrounding the oil stain that spills from the ship. When the time-varying probability density functions are not Gaussian, as in our case, the centers of mass of the Voronoi cells may not coincide with their maximum peaks of the density function; this may cause the robots to be driven toward sub-optimal areas.

2.2 The MARL method

In contrast to these applications described in Section 2.1, we propose in Chapter 3 an approach that permits to dynamically update the robots' behavior with respect to the estimated people distribution in the environment and the associated risk of contaminant. Moreover, we are interested in providing an approach suitable for high resolution priorities in very large railway stations and scalable with respect to the number of robots in the team. In particular, Reinforcement Learning (RL) approaches have been deployed in the context of cleaning or disinfection of structured environments based on double DQN [68] or Actor-Critic Experience Replay (ACER) [50]. These approaches are more flexible than the ones based on CPP, but usually, only single-robot setups are considered and in static contexts where there is not a dynamic propagation of contaminations. On the other hand, Multi Agent Reinforcement Learning (MARL) frameworks are often proposed to ensure flexibility and scalability in different applications like exploration [98], construction [72], or target-capturing [84], while priority-based cleaning issues are less explored. For instance, in [98], the authors proposed a cooperative multi-agent Deep Reinforcement Learning technique to solve multi-target capturing tasks, where a unique end-to-end network is deployed and trained over local robot-centered portions of the environment from each agent. Similarly, in [72], the authors presented a cooperative multi-robot exploration framework exploiting a two-phase multi-task Reinforcement Learning approach with cautiously-optimistic learners in combination with deep recurrent Q-networks for action-value approximation. Here, each robot is endowed with a specific neural network trained on local information from onboard sensors, supporting the modularity and the scalability of the framework. In [51], the authors developed a method to implement path planning and obstacle avoidance based on deep Q-learning with experience replay and a specific heuristic, which guides the robots to avoid the blind exploration of the action space during the training process. On the other hand, since the behavior and the distribution of people in a railway station are hardly predictable, the definition of such heuristic can be challenging. Similarly to our approach, a distributed framework is proposed in [84], where multiple agents learn a collaborative policy in a shared environment using the A3C training method for a construction task. Each

agent perceives the other agents as moving features of the environment (hence no explicit communication is exploited). The effects of the actions are then detected by agents as disturbances to be compensated during the construction task, while the cooperative policy arises from the shared goal. Differently, in [103], the authors consider a decentralized MARL problem where the agents are connected via a time-varying and possibly sparse communication network. This method introduces a complex communication between robots, which have to merge local information from onboard sensors with the observations from the other agents of the team, and may lead to communication overhead [17]. In contrast, we are interested in investigating the case in which the information about the whole environment is shared between the robots through the heatmap exploiting the WiFi infrastructure available in the railway stations. Such heatmap is used to coordinate the agents hence no further explicit communication is needed. A similar approach is also described in [4], where the authors propose a multi-robot path planning algorithm using Deep Q-Learning combined with a Convolution Neural Network (CNN). In this case, a single image is used to represent the map, the obstacles, and the position of the agents in the environment, while the CNN is deployed to produce suitable paths for each robot in order to reach a single target location. Analogously to our work, the authors propose multi-agent Deep Q-learning, however they address a path planning problem with fixed targets, while we are interested in cleaning/sanitizing tasks with dynamically changing prioritized maps.

2.3 The MARL method extended to consider heterogeneous multi-robot teams

The work proposed in Chapter 4 is an extended version of our previous multi-robot sanitizing framework [12, 14] where heterogeneous agents are considered. Specifically, we extended the framework by allowing different robots with different features - such as cleaning range, the shape of the cleaning area, or speed - to cooperate during the execution of the shared cleaning task. Analogously to [12, 14], we propose a framework based on a decentralized deep reinforcement learning, where each robot of the team performs its own learning process. Our claim is that the robot-specific policies produced by this approach permits cooperation between the het-

erogeneous robots without performance degradation of the overall team. The need to consider heterogeneous robots is relevant for the infrastructure manager. Different typologies of robots with different cleaning strategies can be deployed, for instance, by integrating several small (low-cost) robots, having limited sanitizing capabilities, in alternative to bigger but more performing ones. The possibility to increase the number of robots in a team with different or less expensive models without reducing the cleaning performance of the overall system may be convenient in terms of costs but also in terms of hardware and software maintenance [44], especially over prolonged usage periods [73]. In the literature, frameworks that simulate heterogeneous teams of robots are often considered and deployed in several different contexts [89]. For instance, in the pursuit-evasion class of games, robots with different capabilities cooperate to catch moving targets in a shared environment and their pursuit strategies must be adapted with respect to the behavior of the opponent [91, 96]. This case is similar to our domain, where clusters of people appear/disappear/move around the station and the robots' cleaning strategy should be adapted accordingly. The benefit of the coordinated heterogeneous robots is also emphasized in [99] where different robots (aerial and ground vehicles) are deployed. In this work, our main contribution is the design of a heterogeneous framework where multiple mobile robots of different characteristics and typologies learn to cooperate during the execution of cleaning tasks. To evaluate the approach, we consider a very large crowded environment from a real railway station, exploiting WiFi information about the distribution of people in order to assess the performance of different heterogeneous teams of robots. We also propose an assessment of a heterogeneous robotic team in a real case study, using a one-day data recording of the people's movements inside the Roma Termini station, retrieved from the Meraki Cisco System WiFi Network. In this context, the empirical results collected show that the performance of the heterogeneous team is comparable to that of the homogeneous team working under the same conditions.

2.4 The HMILP-MPC Multi-Robot method for Priority-based Sanitization of railway stations

As previously introduced in Section 2.1, solutions to the cleaning task,in the literature, consider the environment as a grid-map that is partitioned in several parts and assigned to different robots executing fixed shaped cleaning paths (spirals, triangles, rectangles, etc.) [68, 71, 62, 54, 50, 53, 52, 67]. These methods are effective to provide a continuous cleaning service which maximize the coverage and minimizes the idleness of the agents. Additional works regarding the CPP methodology, consider the presence of some zones of the environment with persistent static higher priority, that needs to be cleaned/sanitized with pre-specified higher frequency [58, 90, 65, 74, 22]. Those approaches often consider a graph-based representation of the environments with only a limited number of nodes. In contrast, we are interested to find a solution considering high resolution and dynamic priorities in very large railway stations. An interesting work is described in [22], where the authors solve the multi-robot persistent coverage control problem over a grid environment using Mixed Integer Linear Programming (MILP) to maximize the coverage level of the cells over a finite horizon. In their study, the authors defined terminal constraints to ensure that the agents are always able to terminate their plans in predetermined closed trajectories. Also in our framework we consider persistent priorities from the historical data recorded by Meraki WiFi Network, but in contrast, we are more interested to find a solution that may adapt to different circumstances, following the unexpected changes of priorities of the environment, without fixed trajectories or fixed nodes. Moreover, we chose to design a Hierarchical MILP (HMILP) [8, 33] where the problem of finding different paths for different robots in a common graph is decomposed in a cascade of MILP problems, one for each robot, and each selected path is used to constrain the selection of the followings . This way, robot starting from the same initial conditions are forced to select different paths to increase the coverage. In this context Model Predictive Control (MPC), has often been used to find local solutions in open environments with a small number of obstacles [2]. Unfortunately, including geometric constraints for complex and very large indoor environments with static obstacles makes

the optimization problem very expensive to solve in real-time [1]. This depends by the number of added obstacle constraints, increasing the iteration cost of MPC solvers, and the extra non-convexity making it hard to find good local solutions. For this reason, we propose in Chapter 5 an approach that avoids geometric constraints in the control layer by delegating collision checking and obstacle avoidance to a lower-level Artificial Potential Fields (APF) navigation module. Similarly to our work, in [87] an APF framework controlling and coordinating a group of robots for cooperative manipulation tasks is proposed. Also in [81] APF method and optimal controllers methods are used for path planning of autonomous vehicles. The authors highlighted that the main advantage of the Artificial Potential Fields approach over the other path planning methods is its low calculation cost, even with complex Potential Fields for obstacles and road structures. Following these perspectives, in Chapter 5, we propose a MILP-MPC approach combined with APF, which is able to produce high-level strategies for the cleaning/sanitizing of crowded environments by on-line adapting the agents' behavior with respect to the current distribution of people. In particular, the proposed system relies on anonymous information from the preexisting WiFi infrastructure, which is often available in public environments, to estimate the position of people [82] and to prioritize the sanitizing process toward more crowded areas of the environment. Moreover, we also defined a simple parametric model to estimate the spreading of contaminants (bacteria or viruses) due to the presence of people in order to better estimate the risky areas to be cleaned with higher priority.

2.5 Bioinspired Artificial Cockroach Colony Strategy combined with 2-type Fuzzy Logic

The sanitization of an environment may be seen as a problem of coverage path planning (see also the Section 2.1). In this sense, there are several studies in literature where the authors suggest cleaning and sanitization as possible applications. Classical approaches to the coverage are [61, 68], where the environment is divided into different zones, and every robot covers the assigned subareas using paths with a boustrophedon or spiral shape. Other studies are focused on finding paths that ensure the

cleaning of prioritized zones with higher assigned frequencies than the others [90, 65, 74]. In contrast, we are interested in considering a dynamic environment in which every zone may increments its priority to be cleaned during the day, following the spreading of contamination caused by people's movements and the natural contaminant's diffusion law. Specifically, in Chapter 6 we rely on Ant Colony Strategy (ACS) to guide multiple robots toward our dynamic context. In [105], a coverage planning method in fields with multiple obstacle areas was presented. The method implies that the field is divided into blocks around the in-field obstacles, such that the blocks contain no obstacles. This study applies ACS to add adaptability to the problem of coverage path planning, but in contrast to our method of Chapter 6, the ACS is applied only to find the optimal block traversal sequence formulated as a TSP problem in a static environment. In [23] based on the ant colony algorithm, every robot uses the distance matrix to get the optimization sequence of the sub-areas after the decomposition of the coverage environment. The robot covers the local sub-area through boustrophedon motion. In contrast, in our approach of Chapter 6, every robot needs to decide on a new position at every step, considering the dynamic environmental conditions. Similarly to our approach, in [80], a group of ant-robots utilizes the principle of pheromone-based coordination: each robot deposits pheromones on boundaries of its territory to inform the others about the already covered areas. When full coverage is achieved, each robot patrols its territory by persistently moving on the territory border. In [16] the authors use a multi-robot coverage approach based on honey bee colony behavior. Specifically, they propose a honey bee-inspired pheromone signaling method: if a robot receives a pheromone, it decides where to move by selecting a target destination in the opposite direction of the received pheromone. In [10, 9], the authors combine the effectiveness of the two pheromone-based approaches detailed above while overcoming the major weaknesses of each approach taken alone. These approaches use pheromone communication as a repulsive signal to increment the spreading of the robots and favor the exploration of new areas, as in our case. In contrast, they consider the coverage problem from the point of view of the patrolling and surveillance task in a static environment, not the cleaning in a dynamic context. Together with the pheromone mechanism, an ACS may include another procedure: the daemon actions [31].

Daemon actions are an optional component of the ACS meta-heuristic and can be used to implement centralized actions which single ants cannot perform. Examples are the collection of global information that can be used to decide whether depositing additional pheromones is useful to bias the search process from a non-local perspective. An example is in [83], where a mobile robot team cooperates to detect obstacles. When a robot faces an obstacle during navigation, it will store its location in the global memory, and all the robots can access it. In our case, we deploy a similar strategy: we define a centralized WiFi server that creates a heatmap representing the current global state of the station, also exploiting the 2-type Fuzzy logic to assign attractive regions to the robots depending on contaminated areas in the map. The Fuzzy logic method is useful when processes are too complex for analysis by conventional quantitative techniques or when the available sources of information are interpreted in a qualitatively inaccurate or uncertain way [21] or the mathematical and statistical description of the time-variability is unknown in a variable time system [63], as in our case. Examples of studies in which type-2 Fuzzy is used together with the ACS technique are in [77, 28, 83]. In [69] the authors describe the preference of cockroaches to choose for their paths the shadowed zones in order to feel safer. Inspired by this work, we modeled, in Chapter 6, our robots as cockroach-like agents that prefers at every step of their path the attractive regions with higher heatmap's priority distribution. In [70, 69] the paper shows that in the exploration phase, the cockroaches prefer to visit zones where there are no other members of the colony. To this end, they may also generate a repulsive pheromone to move away other members. In this work we apply a similar strategy, we program our agents to generate a repulsive pheromone to discourage agents aggregations. In [27], the authors provide evidence that the cockroaches in their exploration prefer to choose destination-shadowed shelters with more food and no other members of the colony. In this sense, the robots will find more attractive the areas assigned to each of them by their daemon server, where they find more density of priority and higher distances between the robots. Following these studies, we propose a method of sanitization of large and dynamic environments based on a multi-robot bio-inspired artificial cockroach colony strategy that uses two levels of the 2-type fuzzy logic systems to coordinate a team of robots. The sanitization

follows the rapid and continuous propagation of the contaminations represented in the heatmap, where different colors are associated with different levels of priority of zones. We considered for our simulations the biggest and most important station in Italy, Roma Termini, in the capital. We used for our tests one day of real data about the positions and motion of people in Roma Termini, stored by the current WiFi Infrastructure Network Meraki Cisco System, shared by the Italian Infrastructure Manager Rete Ferroviaria Italiana. We compared our solution with two examples of conventional coverage path planning strategies commonly proposed in the literature for sanitization [61, 68] and two solutions studied specifically for our case of study, capable of adapting their strategies to a dynamic environment where the contamination expands [14, 12, 15, 13].

Chapter 3

A Multi-Robot Deep Q-Learning Framework for Priority-based Sanitization of Railway Stations

Sanitizing railway stations is a relevant issue, primarily due to the recent evolution of the Covid-19 pandemic. In this Chapter, we present a multi-robot approach to sanitize railway stations based on a distributed Deep Q-Learning technique. The proposed framework relies on anonymous data from existing WiFi networks to dynamically estimate crowded areas within the station and to develop a heatmap of prioritized areas to be sanitized. Such heatmap is then provided to a team of cleaning robots - each endowed with a robot-specific convolutional neural network - that learn how to effectively cooperate and sanitize the station's areas according to the associated priorities [1]. The use of autonomous robots trained together to accomplish the sanitization task is strategic to approach in a dynamic environment where the contamination propagates during the day according to the diffusion laws and the movements of the clusters of people in the environment. We have just shown the background and the related works relative to this solution in Section 2.2 The proposed approach is evaluated

[1] Study published in [12, 14]. The source code of the system is available at the following link: https://github.com/Tavano1/MultiRobot_Sanitization_Railway

in a realistic simulation scenario provided by the Italian largest railways station: Roma Termini. In this setting, we consider different case studies to assess how the approach scales with the number of robots and how the trained system performs with a real dataset retrieved from a one-day data recording of the station's WiFi network. The rest of the Chapter is structured as follows. In Section 3.2, we describe the architecture of the proposed framework, the overall learning process, and the dataset used for testing. In Section 3.3, we present the case studies and discuss the collected empirical results.

3.1 Formal Definition of the Problem

In this Thesis, we deal with the multi-robot sanitization task considering a team of robots to clean a large railway station without interrupting any public service. In particular, we study distributed strategies to coordinate together the robots in a dynamic environment where there are prioritized zones to be served. The priorities depend on the movements of the passengers that represent vectors of contamination. The distribution and concentration of the crowd in the environment is represented by an heatmap that shows the current spreading of the priorities (the contamination) in the station. The heatmap is constantly updated by a central server that is responsible to share it to every robot during the cleaning execution task.

Given a 2-dimensional occupancy gridmap M generated from the station's planimetry, we denote by X the set of free-obstacle cells in M where a robot can be located. Each robot of the team can execute a set of actions A, where $a_i \in A$ moves the robotic agent i from the current cell to an adjacent one (we assume 8 possible chess-king-like movements). The gridmap M is associated with a set of possible heatmaps S, each representing priority distributions on the gridmap cells. The goal of our learning problem is to generate, for each robot i, a suitable policy $\pi_i : S \times X \rightarrow A$ that maps agent positions $x_i \in X$ and heatmaps $s \in S$ into robot-specific actions $a_i \in A$ driving the agent from the current cell to the next adjacent one to be sanitized.

3.2 The architecture

The multi-robot DQN approach proposed in this work is based on a decentralized client-server architecture where a team of k robots (clients) interacts with a central system (server) that maintains and updates a shared heatmap, whose hot spots are areas to be sanitized (see Figure 3.1). In this setting, each robotic agent is equipped with a robot-specific DQN to learn how to adapt its cleaning behavior with respect to the shared heatmap. More specifically, for each robot, the associated DQN receives as input the current shared heatmap along with the robot position to generate a policy that provides the next adjacent area to sanitize. The shared heatmap maintained by the server can be described as follows. We assume the server endowed with a map of the environment defined by a 2-dimensional occupancy gridmap whose cells represent $1m^2$ areas of the station. Each cell of the gridmap is associated with a priority level, which depends on the presence of people on that cell, which indicates how risky the area is and how urgently the robots should sterilize it. Such priority levels on the gridmap are represented as a heatmap. Given such heatmap, the goal of the agents is to suitably navigate the gridmap by cleaning the traversed cells in order to reduce risky areas and the associated priorities.

To tackle this problem we propose a distributed multi-robot Deep Q-Learning approach [4], where the involved robots are to independently learn coordinated and effective cleaning policies.

The goal of our learning problem, thus, is to generate, for each robot i, a suitable policy $\pi_i : S \times X \rightarrow A$ that maps agent positions $x_i \in X$ and heatmaps $s \in S$ into robot-specific actions $a_i \in A$ driving the agent from the current cell to the next adjacent one to be sanitized, where M, X S and A are the terms defined in Section 3.1.

A representation of the overall architecture is illustrated in Figure 3.1. The framework is composed of a set of agents, representing mobile cleaning robots, each one communicating with the central server. The role of the server (server-side) is to merge the outcomes of the agents' activities with (anonymized) data about people's positions in order to produce the heatmap for the risky areas to be sterilized.

The role of each agent (agent-side) is to receive heatmap and position from the server, update the robot policy exploiting the agent-specific DQN,

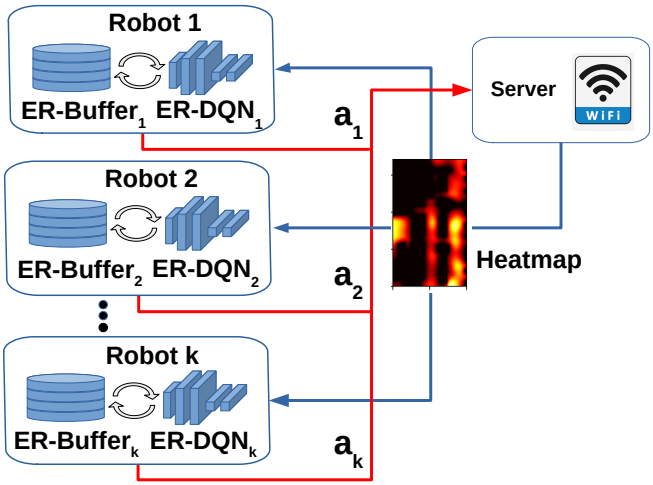


Figure 3.1. Graphical representation of the framework including multiple robotic agents (left) endowed with agent-specific experience replay buffers and neural networks, and a single server (right) exploiting WiFi data to generate a heatmap of priorities (red to yellow spots) shared by the robotic agents. Each robotic agent is equipped with a DQN that receives as input the shared heatmap along with the current robot position to learn the sanitizing policy (i.e, the next $a_1, \ldots, a_k$ actions to execute). Actions are then communicated to the server, which updates the shared heatmap by resetting the cleaned cells.

and generate the next action to be executed.

3.2.1 Heatmap

Heatmap-based models are quite common in literature to represent the spreading of contaminants in indoor environments [97, 60] as well as the distribution and the aggregation of people [95, 40, 29, 36, 55, 37]. In this framework, we rely on a heatmap, whose hot/cold points represent high-/low priority areas to be sanitized depending on the estimated distribution of people on the environment.

The heatmap is then exploited by each sanitizing robot i in order to establish the next area to sterilize according to their specific policy π_i. Specifically, each robotic agent i is associated with a DQN that receives

as input a state-position couple $(s, x_i) \in S \times X$, which is represented as a 2-channel matrix $m \times n \times 2$, where m and n stand for the width and the height of the gridmap. The first channel s is a $m \times n$ matrix representing the heatmap (i.e., the cleaning priorities over the different grids). Each matrix element of the heatmap is a real number in the interval $[0, 1]$, where 1 stands for the maximum priority, while 0 means that no cleaning is needed. This matrix is displayed as a color-coded image (see heatmap in Figure 3.1), where black pixels are 0 priority areas, while colors from red to yellow are for increasingly higher priorities. The second channel x is a binary $m \times n$ matrix representing the position and size of the current cleaning area occupied by the robot, which is 1 for the portions of the environment that are currently in the range of the robot cleaning effect, and 0 otherwise.

In our framework, the update of priorities is performed by the server, which collects the outputs of the individual agents and integrates them, taking into account the position of people and obstacles. As for the update of the heatmap (cleaning priority), it is computed from the position of groups of people (clusters) by modeling the possible spreading of viruses or bacteria using a Gaussian model of dispersion [42]. Specifically, we exploit the periodic convolution of a Gaussian filter $\mathcal{N}(\mu, \sigma^2)$ every ψ steps, where μ, σ^2 and ψ are suitable parameters that can be regulated depending on the meters/pixels ratio, the timestep, and the considered typology of spreading (in this work we assume a setting inspired by the aerial diffusion of the Covid-19 [85]). In our case, we set the μ and σ values according to the spreading parameters proposed in [43, 11].

Notice that since the heatmap update is computed from clusters of people by applying a simulated process, we do not need exact real-time people tracking in the environment, but only an approximated estimate of clusters' positions and movements in railway station areas.

An example of the evolution of a heatmap is proposed in Figure 3.2. Here, starting from a set of static clusters, the probability distribution evolves through the iterative convolution of the Gaussian filter. The convolution process acts at every step by incrementally reducing the magnitude of the elements of the heatmap and distributing such priority over a larger area. This process is then exploited to simulate the effects of the attenuation such as the spreading of contaminants over time.

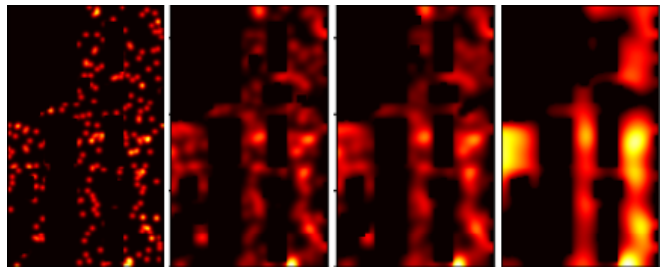


Figure 3.2. Evolution of the priority distribution starting from a random configuration of clusters. The pictures are collected every 10 steps from left (older) to the right (newer). Large black areas are static obstacles to be avoided/ignored by agents.

In Figure 3.2 there are also several fixed black areas that are associated with 0 priority, which correspond to the static obstacles retrieved from the map M. Such unreachable areas are assumed to be always clean, therefore unattractive and implicitly avoided by the robots. During the execution, when a robot i applies an action a_i, it moves to a new cell sterilizing also all the cells that are in the cleaning range of the one reached. This behavior is also emulated by the server, which receives the executed actions from the agents and cleans the corresponding grids by resetting their associated priority to 0.

In order to clarify this process, a portion of the two channels is illustrated in Figure 3.3. In the first channel (left), it is possible to notice that as long as the agent moves through the environment it leaves a wake of cleaned space behind, that is proportional to the cleaning range in the second channel (right). This way, since the priority of already visited areas is 0, agents can indirectly observe their mutual behavior from the priority update, in so avoiding explicit communication.

3.2.2 Multi-agent Experience Replay

In our framework, we propose a multi-agent variation of the experience replay method proposed in [64, 4]. In particular, our training scheme in this multi-agent setting follows a Distributed Training Decentralized Execution (DTDE) approach [39] where each robot updates its own individual policy independently from the others and without explicit information exchange

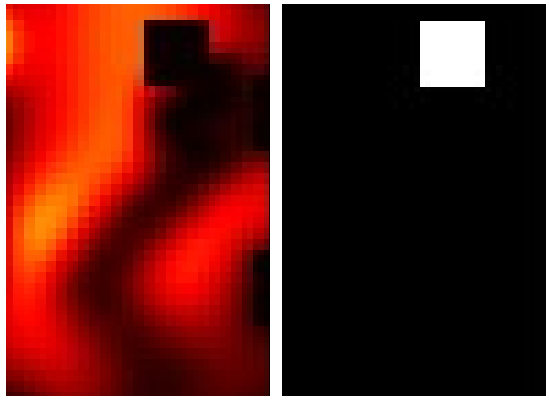


Figure 3.3. Zoomed cut of the 2-channels matrix representing the priority distribution (left) and position and size of the cleaning range of a single agent (right). The motion of the agent produces a low-priority wake in the traversed cells.

between agents. Each of the k agents is then endowed with a specific replay buffer along with specific *target* and *main* DQNs, which are updated synchronously with respect to the position of the agent and the shared environment provided by the server (see Figure 3.1). In particular, for the target and the main networks we deploy a convolutional neural network composed of the following layers: the first layer is a 2D convolutional layer with 32 filters 8×8, strides $(4, 4)$ and ReLU activation; the second is a 2D convolutional layer with 64 filters 4×4, strides $(2, 2)$ and ReLU activation; the third is a 2D convolutional layer with 64 filters 3×3, strides $(1, 1)$ and ReLU activation; the fourth is a Flatten layer; the fifth layer is a dense layer of 512 neurons still with ReLU activation; finally, the output layer is a dense layer composed of 8 neurons with linear activation.

As for the update of the networks, we exploit the Adam optimizer with a learning rate $\alpha = 0.00025$. A graphical representation of such networks is illustrated in Figure 3.4. The 2-channel matrix is firstly processed by the 3 2D convolutional layers then the 3 dense layers are deployed to provide the 8 values of confidence for the 8 possible actions. The id the robot's next action a_i is finally selected as the $argmax$ of those 8 values.

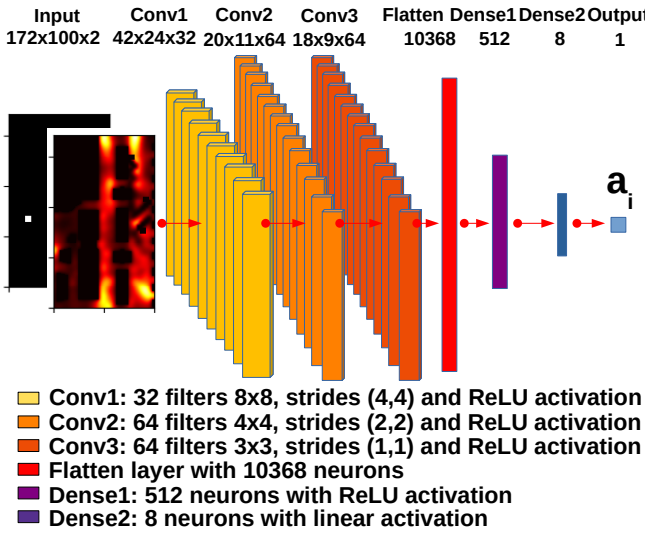


Figure 3.4. Structure of the convolutional neural network used by the robots. Dimensions are reported on top of each layer.

3.2.3 The Learning Process

As introduced in the previous Sections, the proposed learning procedure is composed of an *agent-side* component, where agent-specific training and execution processes take place, and a *server-side* component, where the overall team performance is evaluated and the state heatmap is updated.

In correspondence to these two sides, we define a local reward function r_i (agent-side), which is exploited by each agent to evaluate its performance during the cleaning activity, and a global reward function $r = \sum_i^k r_i$ (server-side) to summarize and evaluate the team performance. The local *reward* function r_i is designed to drive the agents toward prioritized areas of the environment (hot points), while avoiding obstacles and already visited areas (cold points). In this direction, we first introduce a cumulative priority function cp_i that summarizes the importance of a cleaned area:

$$cp_i = \sum_{(j,l)} s(j,l)x_i(j,l) \tag{3.1}$$

which is the sum of the element-wise priorities from matrix s in the area sterilized by the agent i. Specifically, for each grid of the map (j, l) the associated priority value $s(j, l)$ is summed if it has been sanitized by the robot, i.e., when the grid is inside the cleaning range of the robot ($x_i(j, l) = 1$).

The value in Equation 3.1 is then exploited to define the reward r_i for the agent i:

$$r_i = \begin{cases} cp_i & \text{if } cp_i > 0; \\ penalty & \text{otherwise.} \end{cases} \tag{3.2}$$

Specifically, when an agent i sanitizes a prioritized area, the reward is equal to the cumulative value cp_i; otherwise, if no priority is associated to the cleaned area (i.e., $cp_i = 0$) a negative reward $penalty < 0$ is earned (we empirically set $penalty = -2$ for our case studies). This way, agents receive a reward that is proportional to the importance of the sanitized area, while routes toward zero-priority areas, such as obstacles or clean regions, are discouraged. Notice that in this framework, when the action of an agent leads to an obstacle (collision), no motion is performed. This behavior penalizes the agent (no further cleaning is performed), thus producing an indirect drive toward collision-free paths.

The learning procedure for a single robot is described in Algorithm 1. The Algorithm receives as input the id of the agent i and the occupancy gridmap of the environment M used to check actions' feasibility.

For each episode, each agent i selects an initial random position (lines 1-4), which is communicated to the server (line 5). At the beginning of each step, a new heatmap s is received from the server (line 7), which is used to select a new action a_i with an ϵ-probability of a random choice (line 8). The action is then emulated to observe a new agent-specific state (s'_i, x'_i) and a new reward r_i as specified by Equation 3.2. The position x'_i is communicated to the server in order to be integrated into the next heatmap (line 10). The server checks for the task accomplishment (line 11), and the experience replay buffer is then suitably updated (line 12). Finally, if a terminal state is reached (i.e., the variable *done* is true) a new episode is started (lines 13-15).

The server-side learning process is described in Algorithm 2. In this case, when a new episode starts, the random positions of the agents are received (lines 1-4), and a new random heatmap s is produced (line 5). In

Algorithm 1 Agent-side learning algorithm.

```
1: procedure AGENT(i, M)
2:     while true do
3:         done = false
4:         x_i = random_position()
5:         send_position(x_i)
6:         for stp < max_step do
7:             s = receive_heatmap()
8:             a_i = select_action(s, x_i, ϵ)
9:             (s'_i, x'_i, r_i) = emulate_action(s, x_i, a_i, M)
10:            send_position(x'_i)
11:            done = receive_accomplishment()
12:            experience_replay(s, x_i, a_i, r_i, s'_i, x'_i, done)
13:            if done then
14:                break
15:            end if
16:        end for
17:    end while
18: end procedure
```

particular, the random heatmap is created by generating random clusters of people in the free cells of the occupancy gridmap M. At the beginning of each step, a new heatmap s is sent to the agents (line 7). Each agent exploits the heatmap to select a new action according to its policy (see agent-side Algorithm 1, lines 7-10) sending back to the server the newly reached position. These positions are then received by the server (line 8) and used to update the heatmap by resetting (cleaning) the associated priorities (line 9). If the updated heatmap is sufficiently clean (we assume that the task is finalized when 98% of the map is clean) the task-accomplishment is communicated to the agents (lines 10-13) and a new episode can be started (lines 14-16). Otherwise, if the map is still not clean, the Gaussian spreading of contaminants is simulated (line 17).

Algorithm 2 Server-side learning algorithm

```
1: procedure SERVER(k, M)
2:     while true do
3:         done = false
4:         (x_1, ..., x_k) = receive_positions()
5:         s = generate_random_state(M)
6:         for stp < max_step do
7:             send_heatmap(s)
8:             (x'_1, ..., x'_k) = receive_positions()
9:             s = update_state(s, (x'_1, ..., x'_k))
10:            if accomplished(s) then
11:                done = true
12:            end if
13:            send_accomplishment(done)
14:            if done then
15:                break
16:            end if
17:            s = apply_filter(s, N(μ, σ^2), M)
18:        end for
19:    end while
20: end procedure
```

3.2.4 The Real-Case Dataset

In order to verify the performance of our approach, we collected a one-day real-data recording from the Meraki Cisco System WiFi Network of the Roma Termini station. The dataset contains information about the position and the distribution of the people in the station within a 24 hours interval starting from 22:00 on 12 August 2020 to 22:00 on the following day.

The dataset is generated from the raw data containing the GPS positions of the visitors' smartphones detected by the station's WiFi infrastructure. Notice that such positions are not only relative to the devices that are connected to the network, but also to those that are within the WiFi range of the system. Therefore, also the owners of smartphones that are inside the station, but are not connected to the station's WiFi network

are considered. This information is collected by several Access Points distributed over the station, all connected to the same WiFi network. The raw data contains, for each access point, the ID of the access point and the list of the connected devices (smartphones) along with their internet navigation data. Notice that these navigation data are not utilized in this work. Moreover, the Meraki system assigns a unique virtual MAC address to every connected smartphone, which is different from the physical MAC address one, to ensure the privacy of every visitor. We exploited such virtual address to identify the devices and to track them over different access points. To establish the positions of the devices in the station from the initial GPS coordinates, the latitudes and the longitudes are firstly converted into planar coordinates, and then a roto-translation into the station's reference frame is performed. Here we assumed the upper-left corner of the map as the origin of our coordinate system. The resulting collected dataset is included in the supplementary information file *Online Resource 1* with the article [14]. It is represented as a comma-separated values file (.csv) organized as a table with 5 columns: the date, the time in which the visitor's smartphone has been observed from the Meraki system, the virtual MAC address, and the coordinates in the station frame. Additional details about the process of populating and updating the heatmap will be described in Section 3.3.

3.3 Case Studies

In this Section, we propose four case studies illustrating the system at work in a simulated railway station. In this context, we considered the map M as a portion of the Roma Termini railway station (provided us by Rete Ferroviaria Italiana S.p.A.) - which is the largest and one of the most populated Italian stations - considering both randomly generated and real data about the distribution of people.

A graphical representation of the environment is shown in Figure 3.5. We selected a region of 100×172 meters (orange rectangle on the right) in front of the rails, where people usually stand waiting for the incoming trains. From that region, we also isolated shops, stairs, and walls as obstacles to be avoided by the robot during the sanitizing process (binary map on the left).

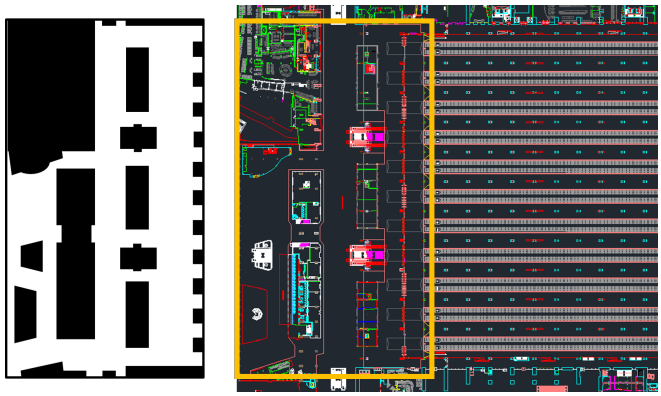


Figure 3.5. Map of the environment (left) selected from the planimetry of the Roma Termini railway station (right). The selected area (orange rectangle) represents one of the most crowded indoor sectors of the station, with an average turnout around 1000 people/per hour.

Agents can move by one pixel in any direction. Hence the set A includes 8 actions (4 linear and 4 diagonal) while, in case one action leads to an inconsistent location (an obstacle or out of bound), the agent stays in the current location. For the sake of clarity, a list of the main parameters is provided in Table 3.1.

The framework has been implemented in Python 3.8.5, using TensorFlow 2.3.1 and Keras 2.4.3, and runs on an Intel Core i9 X-series with 3.3 GHz and 64 GB of RAM.

In the first case study, we assess the system performance during the learning phase considering different numbers of robots (2 to 8 robots). In the second case, a worst-case scenario is considered, where the cleaning performance of robots are assessed considering a variable number of dynamic clusters of people randomly distributed in the station.

In the third case, we propose a realistic scenario considering real data retrieved by the Roma Termini WiFi network during one day of observation. Finally, in the fourth case, we propose a comparison between our method and two additional sanitizing frameworks based on CPP [61, 68].

Table 3.1. Parameters of the framework

Actor	Parameter	Value
exp. replay	discount factor γ	0.99
	maximum ϵ	1.0
	minimum ϵ	0.1
	decay ϵ	$9 \cdot 10^{-7}$
	replay buffer size	10^4
	target network update	10^4 steps
	main network update	4 steps
	batch size	32
WiFi server	refresh period	15 steps
cluster of people	diameter	1 px
robot	cleaning diameter	6 px
	speed	3 px/step
spreading	diameter	5 px
	μ	0
	σ	0.9
environment	dimensions	100x172 px

3.3.1 Case 1: Convergence and Scalability

In this first case study, we designed a simulated station with randomly generated clusters of people to study the convergence and scalability of the proposed MARL approach over different team components. This training setting is intentionally designed to address a generic distribution of priorities that can be encountered by the sanitizing robots during daily cleaning activities.

In order to simulate variable and uniformly distributed clusters in a realistic environment, we set up a suitable emulator of the Roma Termini station in which each location of the map is associated with a 0.02 probability of spawning a new cluster. At the beginning of each episode, a new configuration of the map is generated so a variable number of clusters randomly spawn in different areas of the station whose positions are fixed for the entire duration of the episode. In each step of the episode, the agents acquire the current state from the central server and perform cleaning actions in order to maximize the sanitizing effect, while the server

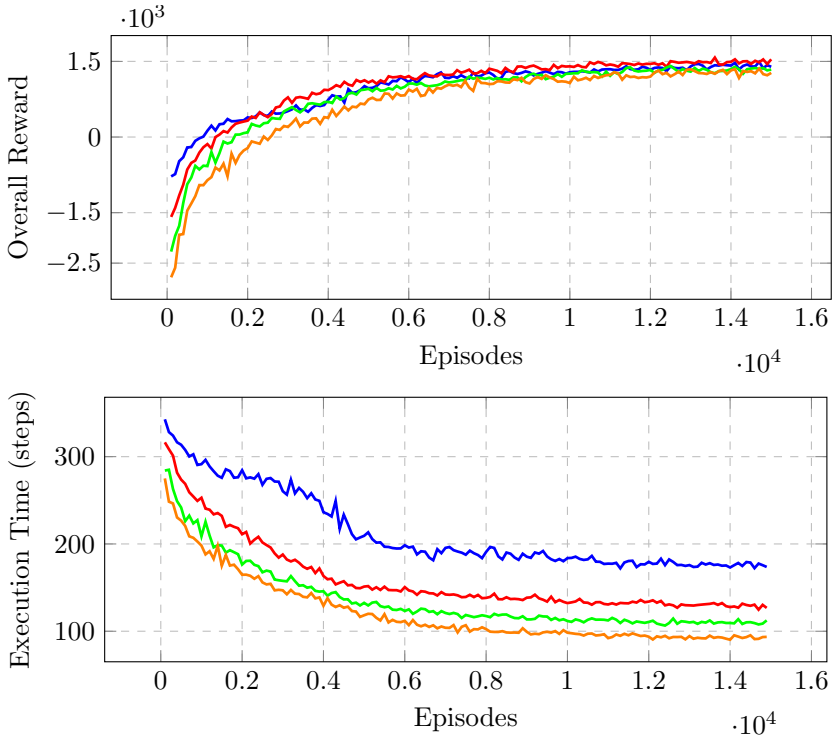


Figure 3.6. Charts of the overall reward (up) and the number of steps needed for task accomplishment (down) considering 2 (blue), 4 (red), 6 (green) 8 (orange) agents.

updates the map by applying the cleaning actions and by expanding the priorities through the Gaussian spreading. The episode ends when agents successfully clean up to the 98% of the map or when a suitable timeout is reached (400 steps in this setting). During the training process, we monitor as quantitative performance measures the overall reward, i.e., the sum of the reward earned by every agent during one episode, and the number of steps needed to accomplish the task.

The values recorded over 15000 training episodes are reported in Figure 3.6. Considering the overall reward (Figure 3.6, up), it is possible to notice that the framework performance is only partially affected by the increased number of robots: all the considered team configurations con-

verge around a value of 1500 after almost 12000 episodes, while only the rate of convergence slightly decreases with the increased number of robots. On the other hand, the time needed to accomplish the task (Figure 3.6, down) clearly decreases as the number of agents increases. Specifically, the 2 agents configuration needs 174 steps on average to accomplish the task, while the 4, 6 and 8 agents ones need 127, 112, and 94 steps, with a time reduction of 27%, 36%, and 46% respect to the 2 robots baseline. These two values indicate that the agents are able to cooperate during the task by minimizing the cross-agents interference (i.e., penalties due to overlapping trajectories) that becomes more likely as the number of agents increases. Specifically, since the agents are always able to successfully clean the map in the later stages of the training - and the maximum achievable reward is similar per episode (random distribution of clusters) - cooperative strategies are expected to reduce the execution time, while keeping the overall reward unchanged. On the other hand, the slight reduction in the reward indicates that the capability to cooperate is still affected by the number of agents, therefore larger teams can be associated with more frequent interference.

3.3.2 Case 2: Random Clusters of People

In order to assess the performance of the system in a realistic dynamic environment, we now propose a different case study where the trained teams are tasked to clean the railway station by considering on-line changing of priorities. In particular, we designed a simulated WiFi server that periodically updates the position of the clusters to a specific frequency while the agents have to continuously adapt their strategies in response to the changed priorities. More specifically, we designed clusters to be randomly generated on the map and randomly updated every 15 steps as to simulate the refresh of the WiFi server. Notice that the resulting uniform distribution of clusters is particularly challenging compared to a real railway station, where people are often grouped near specific areas like shops, info points, or ticket offices (see example in Figure 3.7, up), hence robots can easily converge to these areas and maximize the cleaning effect. In contrast, we propose a worst-case test, where a large number of people is scattered all around the station, and robots should cover a wider area to perform the task.

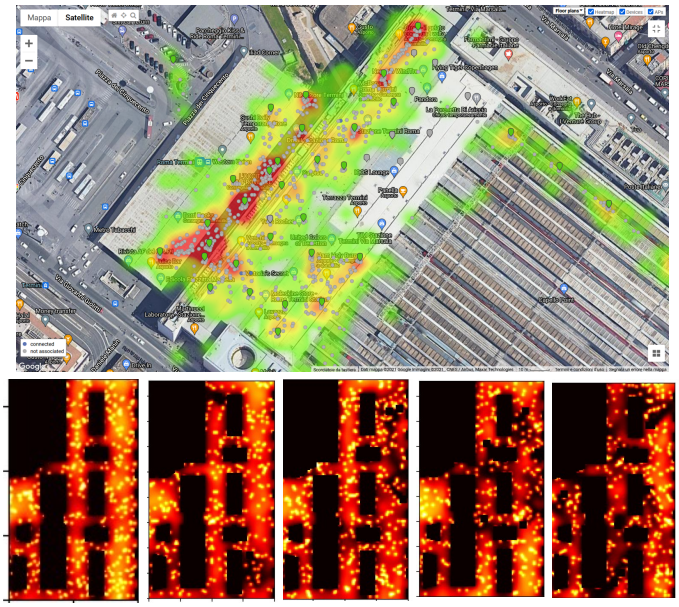


Figure 3.7. Example of the distribution of people inside the Termini station as retrieved from the Cisco Meraki WiFi network (up) and comparison of the 0 to 8 robot settings (down, from left to right) considering the simulated environment with 700 random dynamic clusters after 90 steps of execution.

In order to assess the performance of the teams, we introduce a simple metric c_perc representing the cleaning rate (in percentage) of the map as follows:

$$c\_perc = ((x_{tot} - s_{curr})/x_{tot}) \cdot 100, \tag{3.3}$$

where $x_{tot} = |X|$ is the number of free-obstacles cells of the map, while $s_{curr} = \sum_{(i,j)} s(i,j)$ is the sum of the current priorities for each cell of the heatmap. The value c_perc is then 100% if the map is totally clean (i.e., each cell has 0 priority) and 0% if every cell of the map has maximum priority. For each configuration of teams and clusters, 50 runs of 800 steps are performed. In particular, we tested 6 different environmental settings by altering the maximum number of clusters to be randomly generated in a range between 500 and 1500 clusters. This interval has been selected according to the average number of visitors per hour of the considered

portion of the station (see Figure 3.7, up); moreover, during the runs, the values are designed to be randomly reduced up to the 30% in order to simulate the people entering/exiting the station.

During this test, the server-side process defined in the previous Section (see Algorithm 2) is modified in order to simulate the WiFi refreshing period. The updated process is provided in Algorithm 3. For each run (line 2), an initialization process takes place where a set of random robots' positions and an initial random state are created (lines 3-4). For each step of the run (line 5), when the update period is elapsed, hence new data are available from the simulated WiFi network (line 6), a set of new clusters are spawned (lines 7-8), and the updated heatmap is sent to the robots (line 9). The robots send back their updated poses once the actions are performed (line 10). Finally, after the action execution, the state is updated according to the cleaned areas and the Gaussian spreading of the priorities (11-13).

For this experiment, we have shared the video *Online Resource 2* with the article [14], where a comparison of the 0 to 8 robot settings is shown considering the simulated environment with 500 random dynamic clusters.

In Table 3.2 we propose a comparison between the different team-settings of our method and a 0-robots baseline.

In particular, for each setting, the average (avg) and the average deviation (astd) over 50 runs for both the cumulative *reward* and the cleaning percentage *c_perc* are collected.

These are aggregated values: since we collect the mean and the standard deviation for each one of the 50 runs, we provided *avg* as the average of the means and *astd* as the average of the deviations.

Notice, that the first 100 steps of each run are intentionally discarded, so the collected values are not affected by the transition from the initial empty environment.

The results indicate that, as expected, the cleaning performance is proportional to the number of robots and inversely proportional to the number of people in the station. On average, the value of *c_perc* reduces by 5.3% for every 200 additional clusters and increases by 6.8% between the baseline and the 2-robots setting and by an additional 4.4% for every 2 robots added. Analogously, also the collected *reward* increases according to the size of the team. This suggests that trained robots are able to effectively

Algorithm 3 Server-side testing algorithm

```
procedure SERVER(k, M)
    while true do
        (x_1, ..., x_k) = receive_positions()
        s = generate_random_state(M, MinValue, MaxClust)
        for stp < max_step do
            if stp mod 15 = 0 then
                s = generate_random_state(M, MinValue, MaxClust)

            end if
            send_heatmap(s)
            (x'_1, ..., x'_k) = receive_positions()
            s = update_state(s, (x'_1, ..., x'_k))
            s = apply_filter(s, N(μ, σ^2), M)
        end for
    end while
end procedure
```

cooperate during the execution of the sanitization task. This improved performance can also be noted in Figure 3.7 (down), where the evolution of the heatmap for different configurations of the teams is compared to the 0-robots baseline (first from the left). In this respect, it is possible to notice that, starting from the 2 robot configuration (second from the left), the number of yellow spots (i.e., high-priority areas) significantly reduces, while the cleanliness of the map increases with the number of robots (from left to right).

3.3.3 Case 3: Dynamic Clusters from Real Data

Thanks to the collaboration with the Italian railway infrastructure manager Rete Ferroviaria Italiana, we received one day of real data recording collected the 13 august 2020 by the Roma Termini Cisco Meraki WiFi infrastructure. In particular, the collected data includes the GPS positions of the smartphones owned by the visitors along with an associated id that allows devices to be tracked along the station with around 3 meters of accuracy.

In this case study, the aim is to perform extensive tests of our solution using the received real-data recording showing the behavior of the system in relation to different team settings (from 2 up to 8 robots) and different distributions of people from the different time periods within one day of execution.

In particular, although the station is 24h open, the experiments have been carried out considering a range of time between 6:00 and 22:00 hours. This time interval is associated with a relevant density of people because of the open services and shops, and because of the greater traffic of trains due to tourism, leisure and the work-related commuting in Rome [38]. At the beginning of the experiment, to prevent the dependence of the test from the robot starting point, the initial positions of the agents are randomly chosen between the free-obstacle spaces of the station, while for the environment, we consider a blank initial condition, in which the station is completely clean.

Analogously to the previous case (Subsection 3.3.2), also in this case study, we maintain a 1:1 relation between minutes and steps. We assume a 15 minutes (15 steps) refresh time for the WiFi server to update the heatmap information with the new positions of people from the recorded data. In particular, here we consider a server-side implementation similar to the one proposed in Algorithm 3, but we exploit the real recorded data instead of the randomly generated ones (*generate_random_state* function in lines 4 and 7). If multiple positions are associated with the same id within the 15 minutes interval, the WiFi network is able to collect different waypoints from a visitor's motion, then multiple hot-points are added to the heatmap in order to drive the robots toward the sampled trajectory. A video for this experiment is also available as *Online Resource 3* with the article [14], showing a comparison between the 0 to 8 robot settings.

Since, in this case, the distribution of people is not uniform, in addition to the whole map performance, we also monitor the cleaning effect on the most visited area of the station. Specifically, we selected the most frequented area Z as the portion of the map that lays between the platforms of the trains and their associated access gates (blue rectangle in Figure 3.8).

By empirically analyzing the recorded data, we have verified that in Z, there is far greater probability of finding a gathering of passengers during

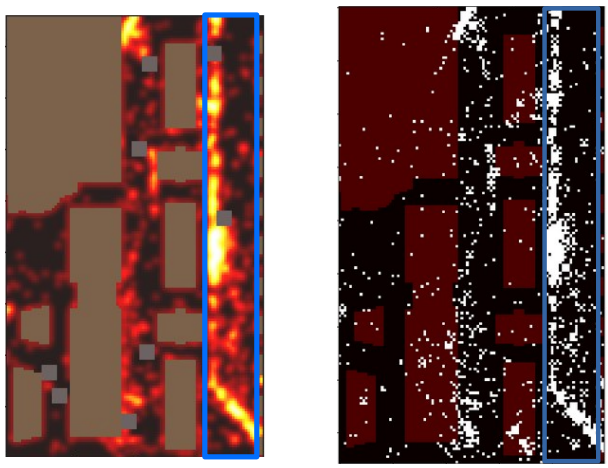


Figure 3.8. Example of our heatmap with a team of eight sanitizing robots (left) and the distribution of people in the Termini station, obtained from GPS position data present in Cisco Meraki WiFi Database (right). The blue rectangle illustrates the zone of the station between the railway platforms and the gates, where there is a high probability of finding a gathering of people.

the day (white pixels in Figure 3.8, right). This condition mainly depends on the presence of the new checkpoints installed near the gates during the period of the Covid-19 pandemic, which people have to pass in order to access the platforms and wait for the trains.

In order to assess the performance on Z, we defined a specific c_perc_z value similarly to the one proposed in Equation 3.3. In this case, we set $x_{tot} = |Z|$ as the number of free-obstacle cells of the map in the zone Z, while $s_{curr} = \sum_{(i,j)\in Z} s(i,j)$ is the sum of the current priorities for each cell of the heatmap in Z.

Our experiment is performed as follows: firstly, we designed a baseline setup in which the real recorded data are simulated without robots, hence no cleaning is considered. Then this 0-robot baseline is compared with respect to the different teams. In Figure 3.9, the performance in terms of c_perc (up) and c_perc_z (down) for teams of 2 (orange), 4 (green), 6 (red) and 8 (blue) agents are compared with respect to the 0 robots baseline (gray). It is possible to notice that, as expected, the cleaning performance for the whole map (Figure 3.9, up) improves rapidly from the

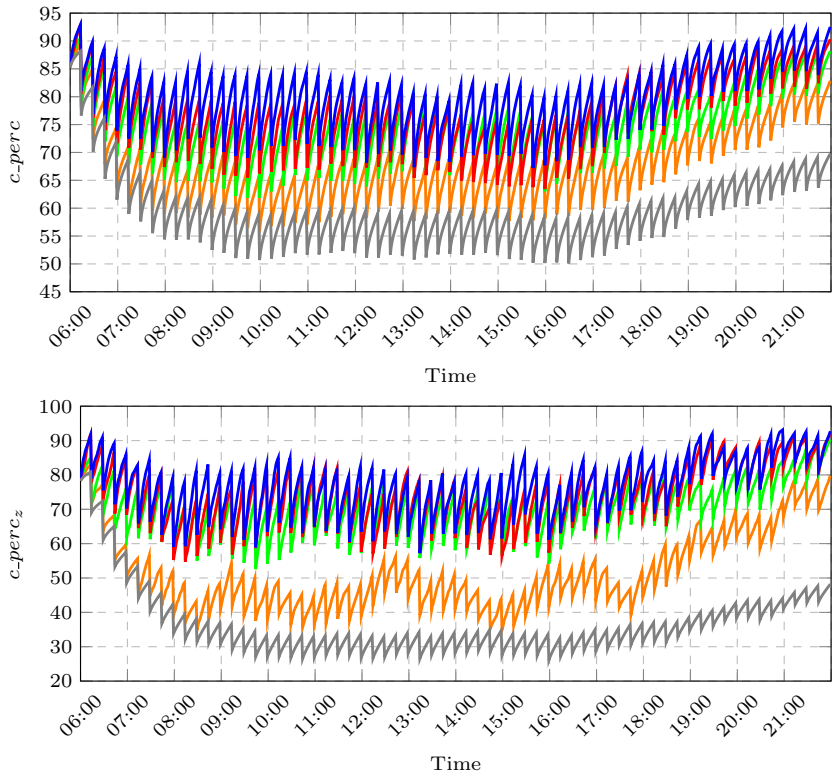


Figure 3.9. Charts of c_perc (up) and c_perc_z (down), considering 2 (orange), 4 (green) 6 (red), 8 (blue) agents and without any robot (gray).

baseline up to the 2 and 4 robots settings, becoming less evident from the 4 to the 8 robots settings. This behavior is more evident in Z (Figure 3.9, down) where the 4 to 8 robots settings strongly outperform the baseline and the 2 robots team. This increment can also be noted from Table 3.3, where the average c_perc and c_perc_z for the different teams over the whole day of execution are shown. By considering the whole map, there is a 20.36% increment of c_perc between the baseline and the 8 robots setting. This increment is almost doubled for the Z area where c_perc_z increases up to 40.28% between the baseline and the proposed framework.

In Figure 3.10 we show details about the 8 robots setting, where the relation between the overall reward (green), the cleaning percentage c_perc

(blue), and the number of visitors in the station (red) is emphasized. It is noticeable that the three values fluctuate with a frequency of 15 steps (equivalent to 15 minutes). This behavior is particularly evident in the c_perc value, whose sawtooth-like behavior directly depends on the 15 minutes refreshing time of the server-side process that continuously updates the heatmap adding new hot points according to the recorded data. In the interval 6:00-18:00, which is associated to an increased number of people along with a wider distribution of priorities, the overall reward increases while the value of c_perc decreases down to a minimum value of 70%. Those behaviors are expected because of the increased complexity of the sanitizing problem associated to the rush hours in the station. In contrast, after 18:00 o'clock, when the number of people in the station significantly decreases, the value of c_perc increases again up to a maximum value of almost 93%.

In order to further highlight the relation between the cleaning performance and the time slots during a single day of execution, we designed an additional experiment where the recorded data are divided into intervals of 2 hours. In particular, we deploy a team of 8 robots in 8 partitions of time from 6:00-8:00 to 20:00-22:00. In this case, we applied for each time interval the server-side Algorithm 3, where the update of the heatmap is executed with real recorded data instead of the randomly generated ones ($generate\_random\_state$ function in line 4 and 7), as described above.

Notice that, in order to decouple each time interval from the previous ones, every simulated interval is started by considering a blank initial condition, in which the station is completely clean. This process allows us to analyze the workload of the team over the different intervals, which is independent from the performance in the previous slots.

In Table 3.4, we provide for each interval the average (avg) and the average deviation (astd) for both the cumulative reward and the cleaning percentages c_perc and c_perc_z over 50 runs. Here it is possible to notice that the performance strongly depends on the number and the distribution of people in the station (workload). Specifically, the 8 robots team shows the worst performance in the three ranges of time (8:00-10.00, 12:00-14:00, and 16:00-18:00), which are associated to the higher quantity of people. In these cases, the average value of c_perc is always lower than 82%. In contrast, better performance are associated to less populated intervals

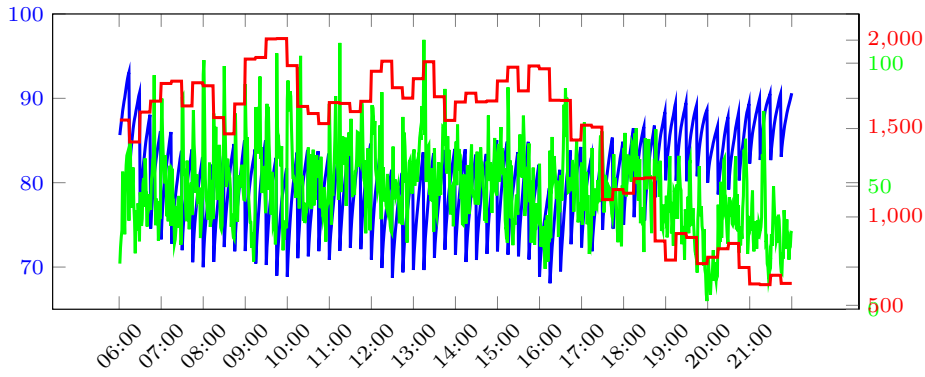


Figure 3.10. Charts of the overall reward (green) and the number of passengers/visitors of the station (red) the percentage of cleaning c_perc for a team of 8 robots in an interval of time between 6.00 and 22.00 o'clock (blue)

(18:00-20:00 and 20:00-22:00) with a average c_perc over 87%. This behavior is also shown in Figure 3.11 where a single run of the framework is provided. In order to ensure the same performance of cleaning in every interval of time, a possible solution can be to provide a cleaning strategy where the number of robots on the team can be dynamically adapted depending on the specific intervals.

3.3.4 Case 4: Comparison with Alternative Techniques

In this section, we propose a comparison between our MARL framework and two typical CPP-based methods for robot cleaning. The first method is a spiral-based [61] exploration of the environment, which is divided into k non-overlapping sub-areas of equal dimensions, each one assigned to a single robot. Sub-areas are partitioned by rectangular decomposition and such partitions are covered by spiral paths. The second method is boustrophedon-based [68] exploiting a modified version of the boustrophedon environment decomposition for partitioning and boustrophedon paths for coverage.

The comparison has been carried out considering the real-data from the previous case study and teams of 4 robots for each method. Also in this case the values of c_perc and c_perc_z are used to assess the clean-

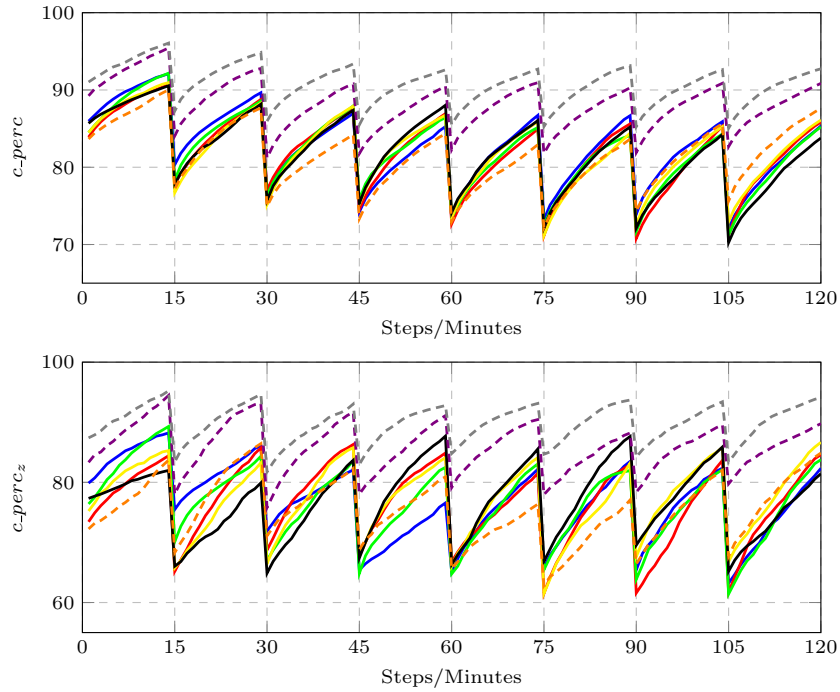


Figure 3.11. Charts of the value of c_perc (up) and c_perc_z (down), achieved by a team of 8 robots in different time intervals: range 6:00-8:00 (blue), 8:00-10:00 (red), 10:00-12:00 (green), 12:00-14:00 (yellow), 14:00-16:00 (black), 16:00-18:00 (orange), 18:00-20:00 (dashed violet), 20:00-22:00 (dashed gray).

ing performance. The results of the tests are shown in Figure 3.12. The proposed MARL framework (green) outperforms the 2 alternatives in the whole station (+5% in average) as well as in the Z area (+22% in average). This increment is particularly relevant in Z, where cleaning performance is always greater than 50% while both the spiral-based (red) and the boustrophedon-based (gray) performance are often under 45% of cleaning. This difference is mainly due to the flexibility of the MARL framework, which is able to adapt the cleaning strategy of the robots depending on the environment, while the CPP-based methods are mainly based on predefined coverage strategies.

Table 3.2. comparison between testing with different numbers of robots and passengers

robots	clusters	*reward*		*c_perc*	
		avg	*astd*	*avg*	*astd*
0	500	N/A	N/A	64.09	2.54
	700	N/A	N/A	56.96	2.71
	900	N/A	N/A	51.46	2.84
	1100	N/A	N/A	47.25	2.98
	1300	N/A	N/A	43.75	3.16
	1500	N/A	N/A	40.87	3.33
2	500	11470	2392	73.63	2.60
	700	10048	3570	65.50	3.32
	900	9651	4058	59.06	3.46
	1100	8553	4602	53.73	3.74
	1300	7103	3932	48.94	3.91
	1500	4340	4113	44.49	3.89
4	500	21026	1721	79.03	2.68
	700	19422	4748	71.39	3.69
	900	19452	4239	65.63	4.04
	1100	17678	7766	59.80	4.30
	1300	16607	6380	55.08	4.79
	1500	11899	7580	49.27	4.82
6	500	26420	2166	81.85	2.64
	700	28124	2625	76.02	3.30
	900	29969	2315	71.00	3.90
	1100	31147	2306	66.62	4.44
	1300	32257	2997	62.58	4.88
	1500	32193	4654	58.44	5.24
8	500	31424	1354	84.32	2.57
	700	30713	2545	78.09	3.39
	900	29561	3367	72.23	4.17
	1100	29654	3253	67.36	4.55
	1300	29511	3679	63.11	5.10
	1500	29174	4007	59.21	5.33

Table 3.3. Percentage of the cleaning in the real-case testing

Team	avg c_perc	avg c_perc_z
0	59.74	36.71
2	67,81	51,63
4	74.74	70.43
6	77.01	74.04
8	80.10	76.99

Table 3.4. Results of a team composed of 8 robots in different time intervals.

rng	$reward$		c_perc		c_perc_z	
	avg	$astd$	avg	$astd$	avg	$astd$
6-8	5917	342.55	82.57	4.73	77.90	6.21
8-10	6353	353.92	81.86	4.96	76.48	6.47
10-12	6120	648.00	82.27	4.29	76.19	6.14
12-14	6460	266.21	81.91	4.57	76.76	6.00
14-16	5692	571.00	81.81	4.97	76.48	6.28
16-18	5289	586.36	82.61	3.77	78.05	4.85
18-20	5083	294.38	87.99	3.00	84.79	4.28
20-22	4207	205.06	90.20	2.57	87.85	3.39

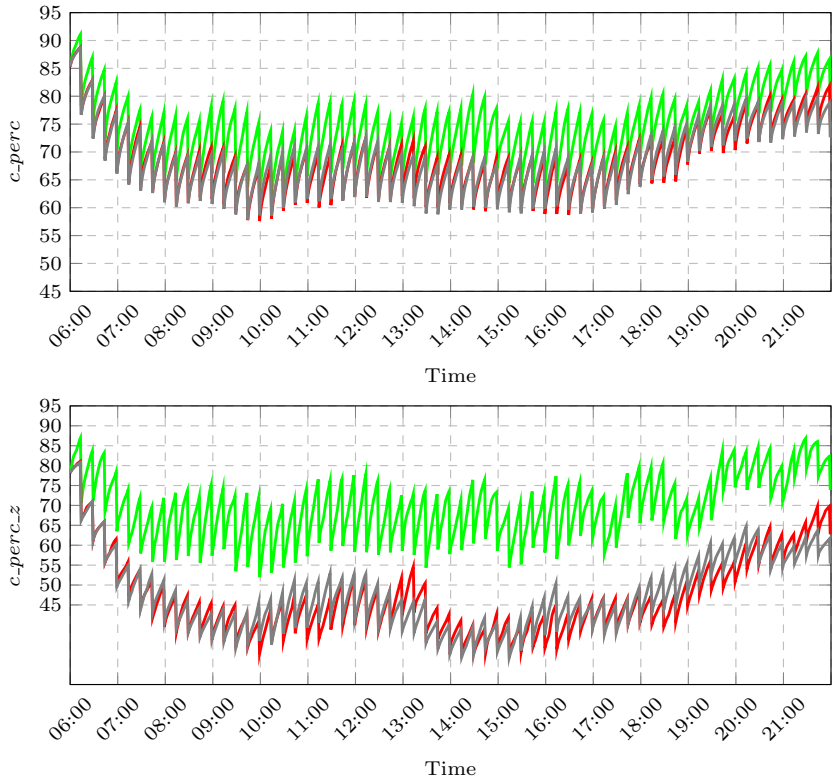


Figure 3.12. Comparison between the proposed MARL framework (green) and the alternative spiral-based (red) and boustrophedon-based (gray) CPP ones considering both c_perc (up) and c_perc_z (down) values. In all settings a team of 4 robots is deployed.

Chapter 4

Toward a Heterogeneous Multi-Robot Framework for Priority-Based Sanitization of Railway Stations

In this Chapter we present a new framework for the prioritized multi-robot sanitization of railway stations based on Deep Reinforcement Learning. The proposed framework represents an evolution of the study presented in Chapter 3. It allows us to define teams of robots having different sanitizing strategies/capabilities, e.g., faster robots rapidly sanitizing small areas in cooperation with slower but long-range ones[1]. Here, robot-specific policies are defined in order to accommodate the different capabilities of the single agents, while two global metrics are defined to assess the performance of the overall team. This capability of managing heterogeneous teams is an important requirement for the Railway Infrastructure Manager Rete Ferroviaria Italiana S.p.A., which plans to verify to what extent different technologies or different strategies can be combined to reduce costs or increase cleaning efficiency. We tested our framework considering real data collected by the WiFi network of the main Italian railway station, Roma Termini, comparing its results with a similar Deep Reinforcement

[1]Study published in [15]. The source code of the system is available at the following link: https://github.com/Tavano1/MultiRobot_Sanitization_Railway

Learning system where homogeneous robots are employed. The rest of this Chapter is structured as follows. In Section 4.1, we describe the architecture of the proposed framework along with the main components and the overall learning process. In Section 4.2, we focus on the experiments about the convergence and performance of the proposed heterogeneous team in comparison with the homogeneous one.

4.1 The Architecture

The multi-robot DQN approach proposed in this work is an evolution of the decentralized client-server architecture presented in [12, 14], where it is now possible to specify different characteristics of each single robot. In particular, the team is composed of k robots with different capabilities, each endowed with a robot-specific policy. The robots interact with a central system (server) that maintains/updates a shared representation of the station in the form of a heatmap whose hot spots are areas to be sanitized. Specifically, we represent the environment as a 2-dimensional gridmap whose grids outline $1m^2$ areas of the station. Grids are then associated with a priority level (the heatmap), which depends on the distribution of people in the station and indicates how risky the area is and how urgently the robots should sterilize it. The goal of the agents is to suitably navigate the gridmap cleaning the traversed grids in the process, in so minimizing the risky areas and reducing the level of priority on the overall map. We formalize our domain as a distributed multi-robot Deep Q-Learning problem [4] where a set of agent-specific policies $(\pi_1, \ldots, \pi_k)$ should be found for the k robots in order to guide each agent toward the cleaning targets. The formal definition of the problem is done in Section 3.1, where we defined the terms M, X, A, S and the policy pi_i for each robot. A representation of the overall architecture is depicted in Figure 4.1. The framework includes a team of different typologies of mobile cleaning robots. Every typology is characterized by a different dimension of the area that agents sanitize during their movements in the environment. Each robot communicates with a single shared WiFi server that is responsible for building a heatmap of the Roma Termini railway station. The server updates the heatmap considering the information about the agents' cleaning activities and the (anonymized) data on the location of people, which

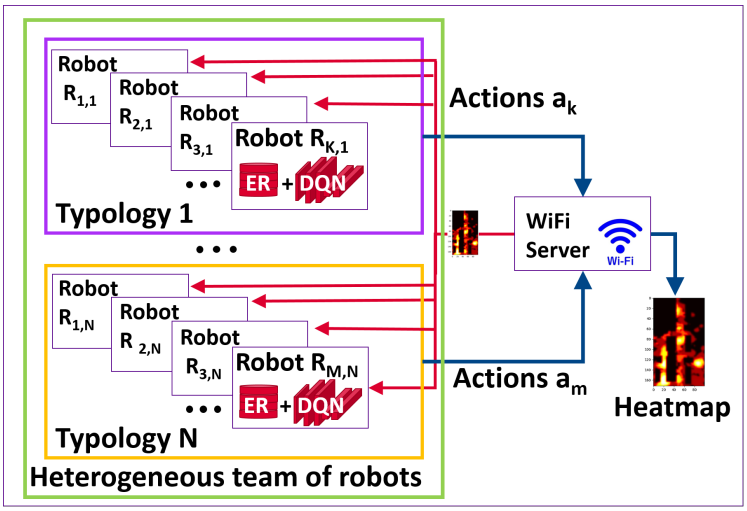


Figure 4.1. Graphical representation of the framework including multiple agents (left), each endowed with agent-specific experience replay buffers and networks, along with a single server (right) that exploits WiFi statistics to provide a heatmap of priorities (red to yellow spots) for the agents.

are used to define the risky areas to be sterilized. The role of each agent is to elaborate the heatmap by means of an agent-specific DQN and to update the local strategy π_i considering their specific capabilities, the environmental settings and the priorities in the map.

4.1.1 Heatmap Definition and Updates

The gridmap representing the environment is built from the real planimetry of the Roma Termini railway station, which has been provided to us by Italian Infrastructure Manager Rete Ferroviaria Italiana. The area of the station selected for our experiments is depicted in Figure 4.2 (yellow box). We defined this area because, on the one hand, it represents the indoor part of the station, where open air and wind cannot attenuate contamination and, on the other hand, it includes the areas of the station where it is more likely to have crowed areas. Selected sectors include: access gates for the railway lines, commercial activities like shops, restaurants, ticket offices, waiting rooms, and luggage storage. Starting from this gridmap, we

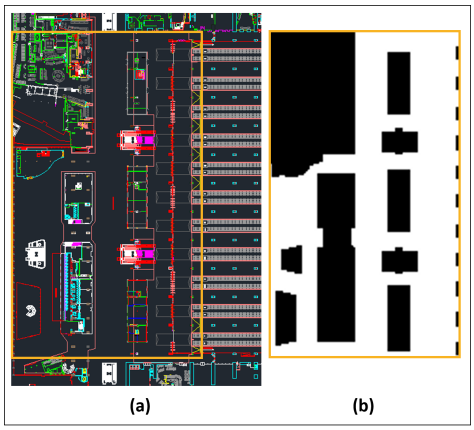


Figure 4.2. Planimetry of the Roma Termini shared by Rete Ferroviaria Italiana (a) and the selected occupancy gridmap (b).

design a heatmap where populated areas are associated with colored spots (from red to yellow) representing the cleaning priority that the heterogeneous team should take into account during the sanitizing process. More specifically, the resulting heatmap has a dimension of 100×172 pixels and a resolution of $1m^2$ per pixel. During every step of the execution, each robot of the team decides the new position to reach in order to start the cleaning action depending on its own specific capability. After a movement, each robot cleans at a fixed cleaning rate (i.e., 4 pixels per step) a cleaning area of fixed dimensions and shape. Each robot in the team has its assigned dimension and shape of the cleaning area. This area-cleaning process is simulated by holding the robot in the current pose for a certain number of steps which depends by its cleaning rate. In our framework, the WiFi Server constantly communicates with all the members of the team to update of the shared heatmap. Specifically, the server updates the heatmap by removing the cleaning priorities of areas sanitized by the robots, while new priorities are also added as colored spots at the positions of newly detected people. Furthermore, at every step of the execution, the server updates the priorities on the heatmap by simulating the natural spreading and the attenuation of contamination over time. This effect is computed from the position of people (clusters) by modeling the possible spreading of viruses or bacteria using a Gaussian model of dispersion [42]. Specifically,

we exploit the periodic convolution of a Gaussian filter $\mathcal{N}(\mu, \sigma^2)$ every ψ steps, where μ, σ^2 and ψ are suitable parameters that can be regulated depending on the meters/pixels ratio, the timestep, and the considered typology of spreading (in this work we assume a setting inspired to the aerial diffusion of the Covid-19 [85]). In our case, we set μ and σ according with the spreading parameters proposed in [43, 11]. An exemplification of the evolution of a heatmap configuration is provided in Figure 4.3. The convolution process acts at every step by incrementally reducing the magnitude of the elements of the heatmap matrix, while distributing the priority on a wider area. Notice that in Figure 4.3 there are several black areas (0 priority) that are regions of space associated with the static obstacles of the environment (shops, rooms and walls inside the station). These areas are assumed to be always clean, hence unattractive for the robots. When an agent moves with an action $a_i \in A$, it sends the new position to the WiFi Server. The region of the heatmap in the neighborhood of the newly reached position, with the cleaning area assigned to the agent, is cleaned by the server, which then sets to 0 the associated priority level when updating the heatmap.

4.1.2 Multi-agent Experience Replay and the Learning Process

In our framework, we propose a multi-agent variation of the experience replay method proposed in [64, 4, 12]. In particular, our training scheme exploits a Distributed Training Decentralized Execution (DTDE) approach [39], where each robot is independent during both the execution phase and the training phase, while its own individual policy is updated by considering only its own experience, without explicit information exchange between robots. In this framework, our idea is to exploit this DTDE approach to allow robots of different types to cooperate in a heterogeneous team. Robot-specific capabilities are: the travelling speed of the robot in the map (denoted by the movement length in Table 4.1), the shape and the dimensions of the areas that the robots are able to clean after each movement, and the time that the robot takes to clean the reached area (denoted by the cleaning speed in Table 4.1). In order to ensure that every robot learns by its own experience, each of the k agents is endowed with a specific replay buffer, along with specific *target* and *main* DQNs, which

Table 4.1. Parameters of the framework

Actor	Parameter	Value
exp. replay	discount factor γ	0.99
	maximum ϵ	1.0
	minimum ϵ	0.1
	decay ϵ	$9 \cdot 10^{-7}$
	replay buffer size	10^4
	target network update	10^4 steps
	main network update	4 steps
	batch size	32
WiFi server	refresh period	60 steps
cluster of people	diameter	1 px
long-range robot	cleaning area	25 px
	cleaning speed	4 px/step
	movement length	2 px
	cleaning shape	square
mid-range robot	cleaning area	17 px
	cleaning speed	4 px/step
	movement length	2 px
	cleaning shape	hexagon
short-range robot	cleaning area	9 px
	cleaning speed	4 px/step
	movement length	1 px
	cleaning shape	square
spreading	diameter	5 px
	μ	0
	σ	0.9
environment	dimensions	100x172 px

are synchronously updated with respect to the position of the agent and to the shared environment provided by the server (see Figure 4.1). The target and the main networks are two identical convolutional neural-network composed of the following layers: the first layer is a 2D convolutional layer with 32 filters 8×8, strides $(4, 4)$ and ReLU activation; the second is a 2D convolutional layer with 64 filters 4×4, strides $(2, 2)$ and ReLU activation; the third is a 2D convolutional layer with 64 filters 3×3, strides $(1, 1)$ and ReLU activation; the fourth is a flatten layer; the fifth layer is a dense layer of 512 neurons still with ReLU activation; finally, the output layer

is a dense layer composed of 8 neurons with linear activation. The input of the neural network is an image with 2 channels of dimensions 100x172 pixels. In the first channel there is the heatmap, represented as matrix where each element is a real number in the interval $[0, 1]$ where 1 is the maximum priority and 0 means that no cleaning is needed. This matrix can be displayed as a color-coded image (see map in Figure 4.3), where black pixels are associated with 0 priority areas, while colors from red to yellow are for increasingly higher priorities. The second channel x is a binary $m \times n$ matrix (100x172 pixels in our case) representing the position and size of the cleaning area of the robot in the heatmap, which is 1 for the portions of the environment that are currently in the range of the robot cleaning effect, and 0 otherwise. In order to update the networks, we apply the Adam optimizer with learning rate $\alpha = 0.00025$. A local reward function r_i is defined, to permit each agent to evaluate its performance during the cleaning activity in the training process. The local *reward* function r_i is designed to give a benefit to the agents that reach prioritized areas of the environment (hot points), while there is a penalty if a robot meets a fixed obstacle or an already visited area (cold point) in the heatmap. In this direction, we firstly introduce a cumulative priority function cp_i that summarizes the importance of a cleaned area,

$$cp_i = \sum_{(j,l)} s_i(j,l) x_i(j,l) \tag{4.1}$$

represented in Equation 4.1 as the sum of the element-wise priorities from matrix s_i in the area sterilized by the agent i (where $x_i(j,l) = 1$). Such value is then exploited to define the reward r_i for the agent i as follows:

$$r_i = \begin{cases} cp_i & \text{if } cp_i > 0; \\ penalty & \text{otherwise.} \end{cases} \tag{4.2}$$

Specifically, when an agent i sanitizes a priority area, the reward is equal to the cumulative value cp_i; otherwise, if no priority is associated with the cleaned area (i.e., $cp_i = 0$) a negative reward $penalty < 0$ is earned [32] (we empirically set $penalty = -2$ for our case studies). This way, agents receive a reward that is proportional to the importance of the sanitized area, while routes toward zero-priority areas, such as obstacles or clean regions, are

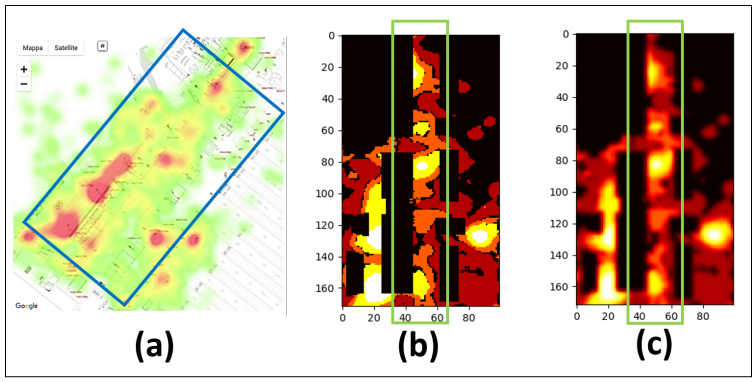


Figure 4.3. Generation of the heatmap from Meraki data. From left to right, the starting georeferenced Meraki data (a) are converted into a robot-frame heatmap (b), which is then updated by the server through Gaussian convolution after 100 timesteps(c).

discouraged. Notice that in this framework, when the action of an agent leads to an obstacle (collision), no motion is performed. This behavior penalizes the agent (no further cleaning is performed), thus producing an indirect drive toward collision-free paths. We define also an *overall reward* function $r = \sum_i^k r_i$ to summarize and evaluate the team performance as illustrated in Figure 4.4.

4.2 Experiments

In this section, we show how the proposed heterogeneous multi-robot framework can be deployed in a realistic environment. As illustrated in the previous sections, we consider Roma Termini station (the largest and most populated Italian railway station) as the environment for our experiments. The station is endowed with several access points managed through a Meraki platform of Cisco System WiFi Network that allows remote operators to monitor the information about the presence and the positions of mobile devices (smartphones) all over the station. This information is exploited by the system (WiFi Server) to estimate the distribution of people and then to update the heatmap shared by the heterogeneous team. An example of the distribution of people retrieved from the Meraki system can be found

in Figure 4.3 (a). We consider the WiFi Server to receive an updated distribution of people every 1 hour. The information from the Meraki is then converted into a heatmap for the robots by associating each location with a priority value proportional to the density of people. Since the information from the Meraki are georeferenced, the retrieved value are finally rotated, translated and scaled in order to match the reference frame of the robots (see Figure 4.3 (b)). Thanks to the collaboration with Rete Ferroviaria Italiana, we obtained an entire day of recording of the Meraki system (2 September 2021) to be exploited for our experiments. In order to assess the performance of the proposed heterogeneous framework, we compare its performance with respect to a similar framework in which a homogeneous team is deployed. Specifically, we assume two teams, both composed of 4 robots: the first team (homogeneous) is composed of 4 mid-range sanitizing robots, while the second team (heterogeneous) is composed of 2 types of agents, namely, 2 short-range robots and 2 long-range ones. The parameters (ranges and velocities) for these 3 categories are shown in Table 4.1. In our tests, we consider for each robot the same cleaning speed. The two teams are associated with equal values of the total sum of the cleaning areas. The movement length of each robot, after the conclusion of the sanitization of its cleaning area, is equal to the ray of its own cleaning area. In the first case study we have compare the convergence of the two teams during the training phase by randomly generating heatmaps to be sanitized by the robots. In the second case, we exploit the one-day recorded data from the Meraki system to test and compare the cleaning performance of the 2 teams in a realistic scenario.

4.2.1 Case 1: The Training Process

In this case study, we compare the convergence of the two teams (heterogeneous and homogeneous) during the training phase. At every training episode we randomly generate priority distributions over the station. The resulting heatmap is left evolving by Gaussian convolution at every timestep, in order to emulate the spreading of the contaminants over time (see Figure 4.3 (c)). The task of the two teams is then to clean the whole map (maximize reward) by avoiding robots overlapping or transitions on clean areas (minimize penality). More specifically, at the beginning of every episode, the framework generates a new configuration of the map by

uniformly distributing clusters of people inside the heatmap. Each location of the map has a 0.02 probability to generate a new cluster. The positions of the clusters, which represents the priority that the robots must sanitize, do not change position for the whole duration of the episode. Notice that in a real case, people are not equally distributed in a station. The training process is designed to generate a challenging context for the teams, while supporting the robots to build a generic policy, which is able to cope with a wider range of possible distributions. In every step of the episode, the WiFi server is responsible for the update of the heatmap: the server firstly deletes the priorities of the areas corresponding to spots cleaned by the robots, secondly it applies the Gaussian convolution to simulate the evolution of the contaminants in the indoor environment. The updated heatmap is exploited by the robots to decide in which direction the next cleaning should be performed. The action of moving toward a location and cleaning the related area is performed according to the specific robot type (Table 4.1). For instance, a mid-range robot (such as the ones of the homogeneous team) is able to cover a distance of 2 px and to clean an hexagonal area of 17 px in 4 steps (around 5 m^2 in 4 minutes). The episode ends when agents successfully clean up to the 98% of the map or when a suitable timeout is reached (400 steps in this setting). During the training process we monitor as quantitative performance measures the overall reward, i.e., the sum of the reward earned by every agent during one episode, and the number of steps needed to accomplish the task. The convergence of the training process for the two teams is depicted in Figure 4.4. We can observe that the two teams are able to converge to a total reward of about 1200 in 20000 episodes. Despite the overall performance seems quite comparable, the homogeneous team converges slightly faster than the heterogeneous one. This was expected given the additional complexity associated with the heterogeneous case, in which the robots are to learn how to suitably combine their different capabilities in order to effectively accomplish the shared task.

4.2.2 Case 2: Testing with Real Data

In a second case study, we considered a more realistic scenario in which the real data from the Meraki system are used to continuously update the heatmap (once every 60 minutes). In order to assess the performance of

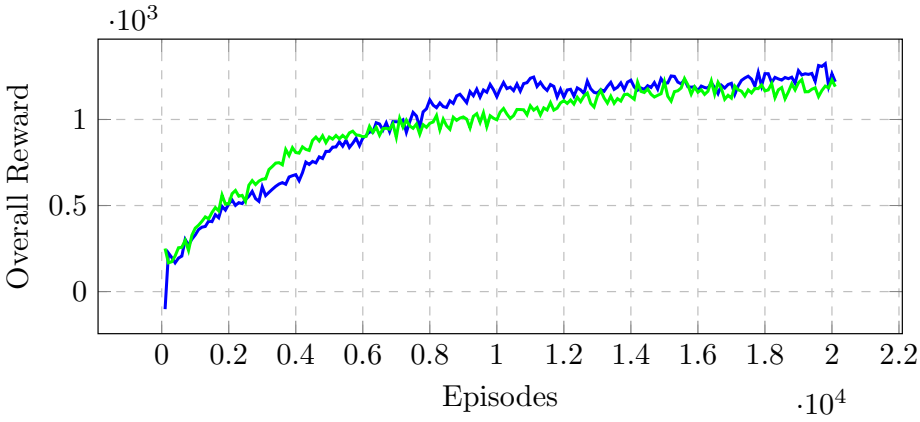


Figure 4.4. Comparison of the convergence between the heterogeneous team (green) and the homogeneous team (blue).

the teams, we introduce two simple metrics to evaluate the efficiency of the cleaning actions of the two teams. Specifically, given the heatmap, we measure how clean it is (in percentage) with the c_perc value defined as follows:

$$c\_perc = ((x_{tot} - s_{curr})/x_{tot}) \cdot 100, \tag{4.3}$$

where $x_{tot} = |X|$ is the number of free-obstacles cells of the map, while $s_{curr} = \sum_{(i,j)} s(i,j)$ is the sum of the current priorities for each cell of the heatmap. Notice that the c_perc is 100% if the map is totally clean (i.e., each cell has 0 priority) and 0% if every cell of the map has maximum priority. We also in introduce the value n_perc representing the percentage of safe spots in the map, i.e., the percentage of number of cells whose priority/risk level is less than a given threshold:

$$n\_perc = (x_{le}/x_{tot}) \cdot 100, \tag{4.4}$$

in this case x_{le} represents the number of cells with low priority. For this experiment, we empirically set the safety-threshold to 0.2. In this way we can measure the percentage of the cleaned surface, given the resolution $1m^2$ for pixel of the heatmap. The above metrics are used to assess the performance of the teams over the whole map. On the other hand, since the distribution of people in the station is not uniform, there are some

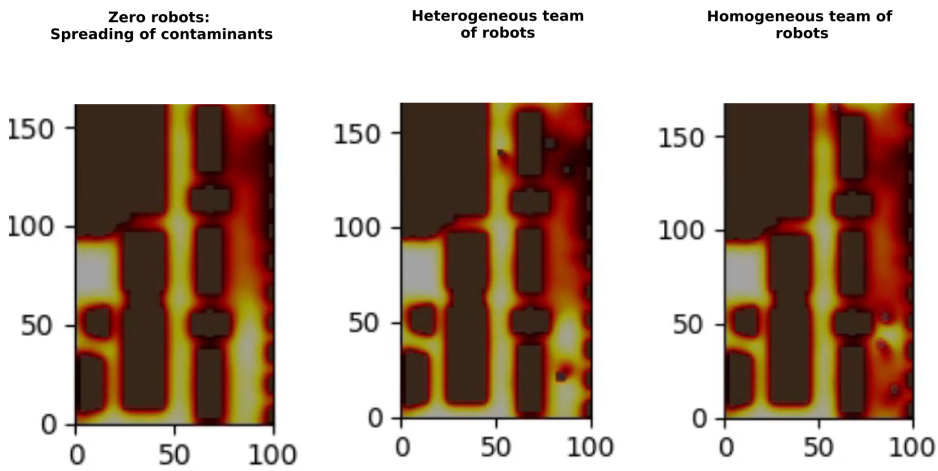


Figure 4.5. Snapshots of the 3 settings at work: no-robot baseline, heterogeneous and homogeneous teams (from left to right).

areas of the station that are generally more populated. This is the case, for example, of the central corridor of the state (green rectangles in Figure 4.3), which is close to the access gates, where people usually crowd. To provide an additional assessment of the performance of the teams, we also measure the c_perc_z and n_perc_z values of the central corridor. In Figure 4.5, we show some snapshots of the system at work. The results of the cleaning process over 13 hours of execution (from 6:00 a.m. to 7:00 p.m, i.e., the busiest time of the day) are illustrated in Figure 4.6. In the latter charts, we compare the performance of the heterogeneous team (green) with respect to the homogeneous one (blue) and a no-robot baseline (grey), where the contamination evolves naturally. From the charts we can notice that both teams obtain similar results. Considering the overall c_perc values (upper charts), the homogeneous team performs slightly better than the heterogeneous one, with an average difference of about 2%. For both teams the improvement with respect to the baseline becomes more evident (over 5%) when the station becomes more crowded (after 2:00 p.m.). As for the c_perc_z measurement, for both teams the improved performance with respect to the baseline looks slightly more evident (up to 10%). In

this case, it is possible to notice that at 9:00 a.m., in the central corridor, the heterogeneous team outperforms the homogeneous one by a mean value equal to 4% and 7% for c_perc_z and n_perc_z respectively. These results suggest that while a homogeneous team of robots performs better than the heterogeneous one on the overall map, the latter seems to work more effectively on restricted and highly populated areas.

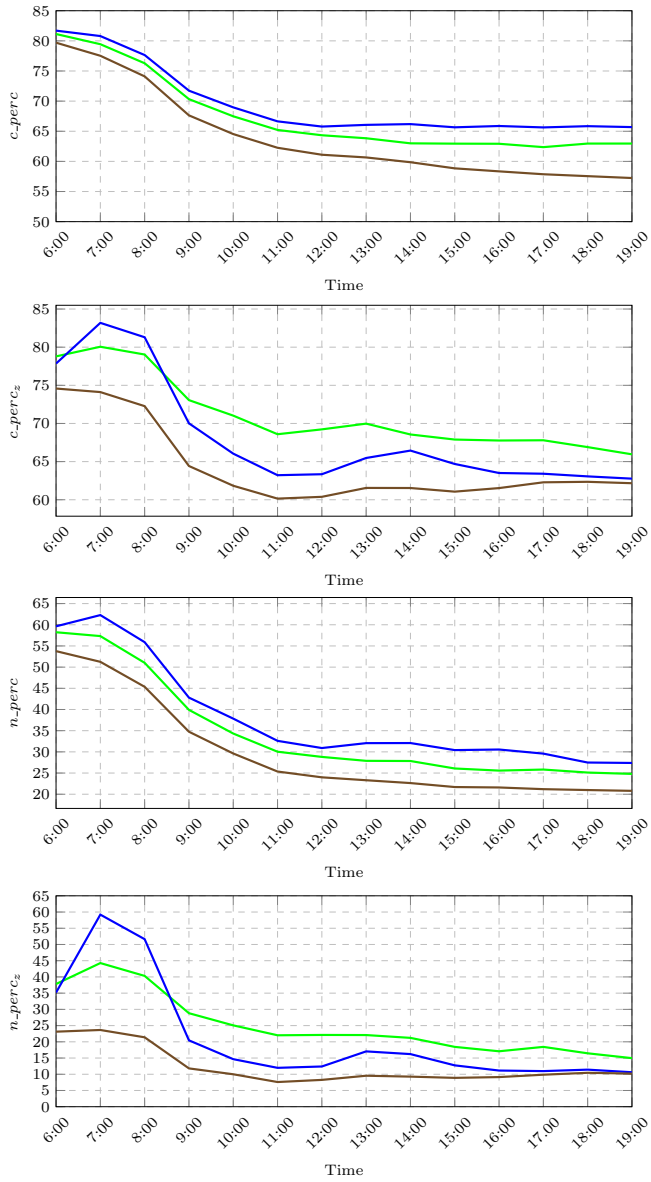


Figure 4.6. Plots of the values c_perc, c_perc_z, n_perc and n_perc_z for the heterogeneous team (green), the homogeneous team (blue) and the baseline with zero robots (grey).

Chapter 5

Combining Hierarchical MILP-MPC and Artificial Potential Fields for Multi-Robot Priority-based Sanitization of Railway Stations

In the previous Chapters, we showed decentralized sanitization frameworks that use Deep Reinforcement Learning approach to answer to the propagation of contamination in a dynamic environment. Every decision about the robots' movements is made by our solution considering the current state of the environment regarding the distribution of the priorities that vary in the time during the cleaning action. In this Chapter, we propose a distributed framework, driving a team of robots to sanitize vast dynamic indoor environments, such as a railway station, but with a deterministic approach. A centralized server uses the Hierarchical Mixed Integer Linear Programming (HMILP) to coordinate the robots, assigning different zones where cleaning is a priority. We investigated the use of a deterministic Model-Based approach to verify the use of historical infor-

mation about people's movements in the past. In particular, thanks to the Model Predictive Control approach, we use historical data about the distribution of people and the knowledge about the transportation service of the station to predict the future dynamic evolution of the position of people in the environment and the spreading of the contaminants[1]. Each robot navigates the large environment represented as a gridmap, exploiting the Artificial Potential Fields technique in order to reach and clean the assigned areas. We tested our solution considering real data collected by the WiFi network of the main Italian railway station, Roma Termini. We compared our results with a Decentralized Multirobot Deep Reinforcement Learning approach. In particular, our main contributions can be summarized as follows:

- We propose a MPC-MILP framework combined with APF, where multiple mobile robots cooperates during the execution of cleaning tasks into a very large crowded environment, exploiting WiFi information about the distribution of people, historical data and the knowledge about the transportation service in the station.
- We tested the behavior of a teams of four robots considering two weeks of real data recorded by the Meraki Cisco System WiFi network of the Roma Termini station.
- We compared behavior of our solution with the Multi Agent Reinforcement Learning (MARL) framework [12]. For this experiment we have shared the video MARLvsHMILP.mp4, at the link: http://wpage.unina.it/fabrizio.tavano/video/MARLvsHMILP.mp4.
- We provided an additional experiment showing how the proposed system can be adapted to consider on-line requests for the sanitization of specific areas. For this experiment we have shared the video CleaningSpecificArea.mp4, at the link: http://wpage.unina.it/fabrizio.tavano/video/CleaningSpecificArea.mp4.

The background, the related works and the reasons of the technical choices regarding this study are discussed in Chapter 2, in Section 2.4. The rest

[1]Study published in [13]. The source code of the system is available at the following link: https://github.com/Tavano1/MultiRobot_Sanitization_Railway

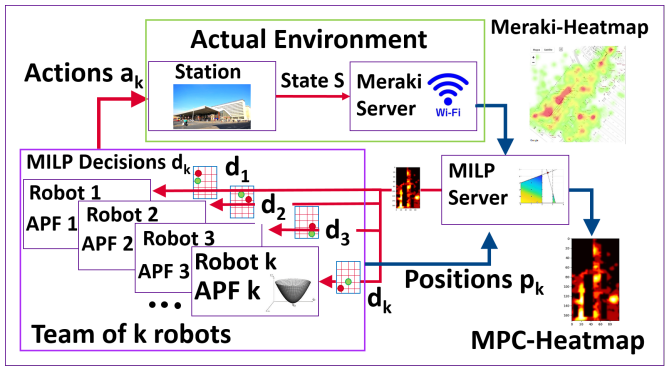


Figure 5.1. Distributed client-server architecture, where the MILP server sends the MPC-Heatmap and the destination nodes to the robots that decide their root thanks to the APF motion path technique.

of the Chapter is structured as follows. In Section 5.1 we describe the architecture of the proposed framework, In Section 5.2 we focus on the case studies about the comparison with the MARL technique presented in [12, 14].

5.1 Architecture

The architecture is based on a distributed client-server architecture represented in Figure 5.1, where a team of robots is on-line connected to a unique server. We decided to follow this choice because our idea is to find a solution that responds to the demand of Rete Ferroviaria Italiana (RFI), using the resources and technologies that are just present in the Italian railway stations as the Meraki WiFi Cisco System Infrastructure that records information on passenger foot traffic that flows the station in form of a Meraki Heatmap.

As shown in Figure 5.1, we consider that the MPC Server receives the Meraki heatmap as visualized in the Meraki Dashboard, with an updating period of 1 hour. In the Meraki heatmap, the colors represent the current distribution of people in the station. This image is converted in a new MPC-Heatmap formatted as gridmap of dimensions 100×172 pixels, in which the colors are now indicating the level of contamination of the areas

in the environment, as shown in Figure 5.2 (a). More formally, we define the gridmap M, the set of free obstacle grids X in the map, the set of the possible heatmaps S and the set of actions Act as described in Section 3.1. The possible actions a_i that the robot can take are the 8 moves in the 8 chess-like directions allowed in a gridmap (up, down, right, left, and the four diagonals). A robot at every step sanitizes a squared cleaning area with dimensions 3×3 pixels of the heatmap. Each element of the MPC-Heatmap is a real number in the interval $[0, 1]$ where 1 is the maximum priority and 0 means that no cleaning is needed. This matrix can be displayed as a color-coded image (see MPC-Heatmap in Figure 5.2 (b)) where black pixels are associated to 0 priority areas, while colors from red to yellow are for increasingly higher priorities. We have also modeled the behavior of the contaminants to spread and attenuate during the time. As for the update of the heatmap (cleaning priority) it is computed from the position of people (clusters) by modeling possible spreading of viruses or bacteria using a Gaussian model of dispersion as in [42]. Specifically, we exploit the periodic convolution of a Gaussian filter at every timestep, where μ and σ are suitable parameters that can be regulated depending on the meters/pixels ratio, the timestep, and the considered typology of spreading (in this work we assume a setting inspired to the aerial diffusion of the Covid-19 [85]). In our case, we use the values of μ and σ as in [12]. The effects of spreading and attenuation of the priorities at every step of time in the MPC-Heatmap is shown in Figure 5.2 (c) after 100 steps. This spreading process is also considered as a model for our MPC method in order to estimate the evolution of the priority over time.

Our strategy of sanitization has 4 main steps in its execution as shown in Figure 5.3. We consider that the cleaning starts with a delay of 30 minutes from the first reception of the first Meraki Heatmap in the morning. Following the arrow of the timeline, we consider a first step where the MPC-Server receives the update of the Meraki Heatmap from the Meraki Server (step A). The MPC-Server applies the HMILP as in [8] to decide the destinations of every robot (step B). In this phase, the MPC Server makes use of information about historical data about the distribution of people in the station in order to make a prediction on the conditions over the next 30 minutes. After (step C), there is a new update of the Meraki Heatmap, and finally, (step D), A new HMILP decision is executed to

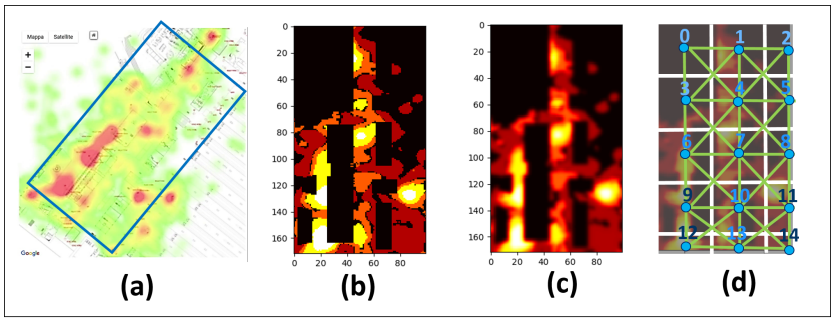


Figure 5.2. Heatmap representation of the Roma Termini Station: Meraki Heatmap (a) MPC-Heatmap (b) MPC-Heatmap after 100 timesteps, MPC-Heatmap with the associated graph.

correct the first prevision considering the new Meraki Heatmap update, without contribution of the historical knowledge. The latter correction is needed to compensate for the unpredictable behavior of the people.

The server is also responsible to verify the positions of the people in the environment thanks to the Meraki heatmap and to receive the positions of every robot inside the station. Those information are used to build MPC-Heatmap (Figure 5.2) with different colored zones. The MPC-Server extracts information about the positioning of the people in the station from the current Meraki heatmap, it receives the positions of every robot and their chosen paths and cleans relative zones in the MPC-Heatmap to prepare the new updated map. MPC-Server provides a partitioning of the current MPC-Heatmap in a discrete and limited number of areas with a fixed step of sectioning (in our case we choose 20 pixels); every resulting subarea will be represented by a single value in an ordered list of partitions $G1$. Considering $i \in N = \{0, n\}$, where n is the total number of partitions of the MPC-Heatmap, the value of the element i of the list $G1$, is q_i, defined as the sum of the priority risk of the relative portion of MPC-Heatmap.The MPC Server applies the same procedure to realize a second ordered list $G2$, using a second heatmap MPC-Heatmap2, which is used to estimate the future expected distribution of people. MPC-Heatmap2 is built considering the knowledge about the transportation service of the station and the average behavior of people in the historical data. RFI defines 4 categories of days to organize the transportation service in the

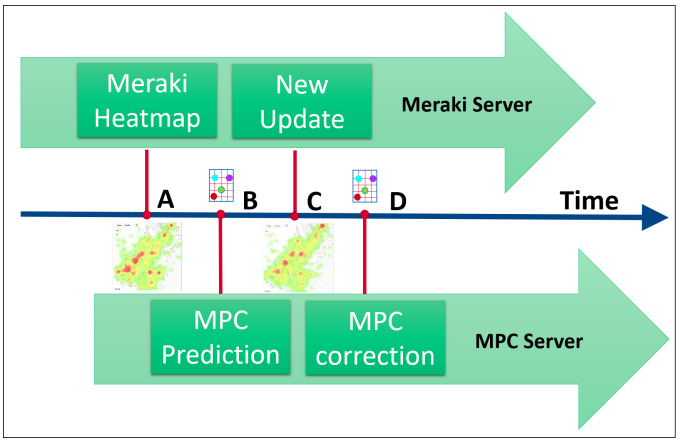


Figure 5.3. Four main steps of execution in the strategy of Sanitization presented in this study.

station Roma Termini: the working day, from Tuesday to Thursday; the Friday, the day before the holidays, including the Saturday; the holiday, including Sunday, The day after the holidays, as the Monday.

In Figures 5.4 the different numbers of departures in Roma Termini, grouped in steps of 1 hour, where every color is associated to a different category. There are differences in number of train services between the 4 categories. RFI defines 4 types of transportation services: The passenger high-speed service by Trenitalia company (TI Pax HSS), the passenger high-speed service of other private companies (OC-HSS), long haul standard low-speed service (TI-PAX-SU) and local transportation (TI-PAX-REG) as shown for example in Figure 5.4 for the cases of working day and holiday. The MPC-Heatmap2 is so realized considering the average of historical Meraki Heatmaps that represent the conditions of the station in the days that belong the same category of day of the current MPC-Heatmap used to built $G1$, but an hour later. Considering $i \in N = \{0, n\}$, where n is the total number of partitions of the MPC-Heatmap2, the value of the element i of the list $G2$, is p_i, defined as the sum of the priority risk of the relative portion of MPC-Heatmap2. Starting by the knowledge of the last received position of every robot in the MPC-Heatmap, the MPC-Server finds for every robot their current starting node in the graph. Then, it cal-

culates the next nodes for every robot, considering the contribution of the values of the nodes of two ordered lists, $G1$ and $G2$. Every robot receives from the MPC-Server the updated MPC-Heatmap and a list of destination nodes that it must visit, following the indications of the HMILP results. Every node represents a specific portion of the MPC-Heatmap. Every robot, acting independently and with autonomy, verifies the positions of the peaks of priorities in the portion of MPC-Heatmap, considering the nodes of its path in the graph. It sorts the found peaks starting from the higher value to the lower value. In case of peaks of the same value, they are positioned in the list from the most closed to their current position to the farthest one. Finally, every robot autonomously uses the APF method to reach the found peaks in the list. We assume that the team of robots is constantly connected with the common MPC-Server thanks to the WiFi railway service of the station. The robots, acting independently, are also responsible for the process of obstacle avoidance. When a robot meets an obstacle, it calculates the shortest path to turn around. It starts to count the number of necessary steps to avoid the obstacles in both directions, then it chooses the shortest route to avoid them. When a priority peak is reached, the robot starts to clean following a spiral shape path centered in correspondence of the coordinates of the found peak in the gridmap as in [52, 53]. We consider that our robot sanitizes an area of the gridmap of dimension 9 pixels at every timestep, where the timestep has a duration of 1 minute as in the study [12, 14]. The Model Predictive Control (MPC) consists of solving an online finite horizon open-loop optimization problem using the current state as the initial state for the team of robots. The solution to the problem gives a sequence of control inputs for the entire control horizon; however, only some elements of the optimal input sequence are applied to the team of robots. The number of used inputs depends, in our case, on the number of peaks of priorities that are present in the MPC-Heatmap, and by the horizon time. In our context, Meraki Server sends an update of the Meraki Heatmap with a periodicity of 1 hour, so our horizon control is also of 1 hour.

The graph representation used to describe the problem in this study is similar to the ones proposed in [106, 107, 30]. We define the Graph $G = (N, A)$, where $N = \{0, 1, \ldots, n\}$ is the set of vertexes and $A = \{(i, j) : i, j \in N, i \neq j\}$ is the set of arcs. Vertex 0 denotes the depot of

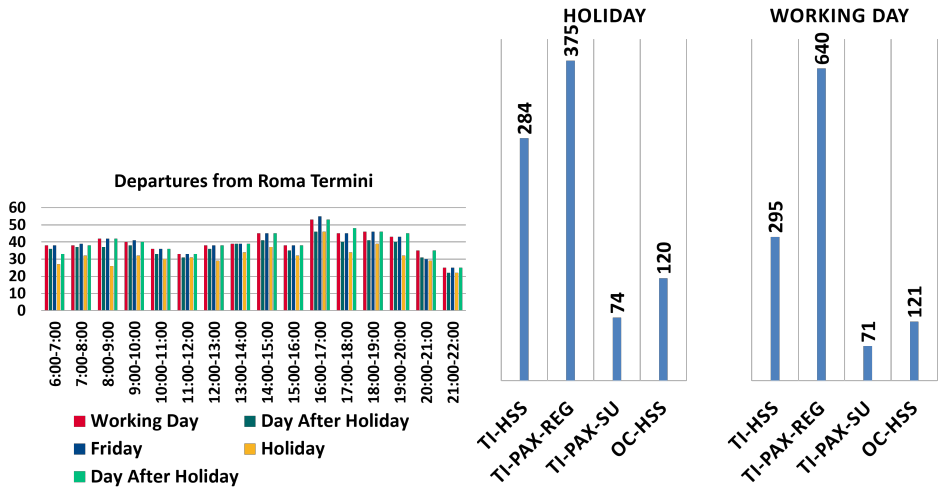


Figure 5.4. Comparison between RFI categories of day for the transportation service in Roma Termini station. To the left different number of departure of trains, at different hours and different categories. To the right, different typology of services for two categories: Holiday and Working Day.

the robots, while the other vertexes correspond to different areas of the partitioned gridmap MPC-Heatmap. Each area i must be sanitized so every node has a non-negative priority number $q_i + p_i$ that represents the total quantity of risk priority in the relative area of the MPC-Heatmap added to the predicted value relative to the MPC-Heatmap2. Each arc between any two vertexes i and j is associated with a distance $d_{i,j}$ (in our solution it represents the reward for the robot that travels between the nodes i and j). The value associated to the arcs takes into account also of the distance of the path in the gridmap to reach the destination node and the presence of fixed obstacles. The reward assigned to the arc is lower in case diagonal movements with respect to the axis of the gridmap (as shown in Figure 5.2 (d)) or in case the robot may meet a wall or other fixed obstacles to be avoided during its travel in the arc. The weight of the edge is calculated by the equation: $d_{i,j} = |(q_j + p_j)|, \quad i, j \in A$. In the HMILP path decision, it is important to take in account also of the cost of the distance that a robot cover to reach the node. In general, a greater distance, depending, for example by the presence of an obstacle,

may reduce the efficiency of the total percentage of cleaning inside the horizon time because our robots also cleans the path to arrive at its target as in [12, 14]. For this reason, with reference to Figure 5.2 (d) we define three kinds of arcs: the straight arcs that connect two nodes in the same x-axis or same y-axis of the MPC-Heatmap. Examples of straight arcs in Figure 5.2 (d) are the connections between the node 0 and 1, or 0 with 3; the diagonal arcs as the arcs as between nodes 0 and 4 or 1 and 5. A third type of arc is those in which the robot meets an obstacle in his path in the gridmap following the connection tracked by that arc. d in G represents a benefit for the robot that runs the relative arc. For this reason the arcs that are associated with a greater distance, as in the case of the presence of obstacles, are also associated with a lower benefit. In particular, we consider the following formula to define the weights of the three defined typologies of arcs:

$$d_{i,j} = \begin{cases} d_{i,j}, & \text{straight arcs,} \\ d_{i,j}/2, & \text{diagonal arcs} \\ d_{i,j}/2, & \text{obstacles in the path} \end{cases} \tag{5.1}$$

At the beginning of the horizon time, the robots start their new path in the graph from the node that was the last reached in the previous hour. The first action that the MPC Server must do is the positioning the robots from the deposit to the current starting node where the robot has sent its actual position. This is done considering a dummy arc with a very high gain. The first step of the path of every robot from the deposit to the starting node is removed from the final path considered for the cleaning and sent to every robot at the beginning of their decentralized operations. The notations, including sets, parameters, and variables used in the model, are listed in Table 5.1. The MILP model and constraints for our multi-robot sanitization problem are the following as in [79] :

Table 5.1. Notations used in the mathematical model

Notation	Description
$\boldsymbol{N}$	set of all nodes $\boldsymbol{N} = \{1, \ldots, n\}$
$\boldsymbol{N_0}$	N and the depot $\boldsymbol{N_0} = \boldsymbol{N} \cup \{0\}$
$\boldsymbol{K}$	total number of k robots
$\boldsymbol{Q}$	max value of sanitization of a single robot
$d_{i,j}$	benefit assigned to the arc between the nodes, $i, j \in \boldsymbol{N_0}$
$x_{i,j}$	is 1 if the k-th robot travels the arc (i,j)
u_i	total priority sanitized by the kth robot
q_i	priority of $G1$
p_i	priority of $G2$

$$s_w(t+k+1|t) = s_w(t+k|t) + h_w(t+k|t) + u_w(t+k|t) \tag{5.2}$$

$$\max(\boldsymbol{z}) \tag{5.3}$$

$$\sum_{i}^{\boldsymbol{N_0}} \sum_{j}^{\boldsymbol{N_0}} x_{i,j} d_{i,j} \geq z \tag{5.4}$$

$$\sum_{j}^{\boldsymbol{N_0}} x_{i,j} = 1, i \in \boldsymbol{N} \tag{5.5}$$

$$\sum_{i}^{\boldsymbol{N_0}} x_{i,l} - \sum_{j}^{\boldsymbol{N_0}} x_{l,j} = 0, \quad l \in \boldsymbol{N}, i \neq j, i \neq l, j \neq l \tag{5.6}$$

$$p_i + q_i \leq u_i, i \in \boldsymbol{N} \tag{5.7}$$

$$x_{i,j} \in \{0, 1\} \tag{5.8}$$

We have defined a simple parametric model to estimate the spreading of contaminants (bacteria or viruses) due to the presence of people in order to better estimate the risky areas to be cleaned with higher priority. The model is represented in equation (5.2). the term s represents the state of an area w in the environment large 3×3 m^2 in the environment, with dimensions equal to the cleaning area of a robot. The state s is the concentration of priority inside the area w in the gridmap M, corresponding to a zone with dimensions 3×3 pixels. the equation affirms that the current density of priority ant the timestep $t+1$ inside the considered area depends on the value of s at the previous timestep, added to two contribution. The first term represents the effect of the spreading due to the Gaussian convolution h, applied to the heatmap at every timestep to model a Gaussian diffusion law of the contaminants. The second term u is the action of the cleaning due to the robot sanitization. We then define the term z as the quantity of the priorities that a robot may remove during its path in Graph G. The objective function of the model is given in (5.3), which maximizes the quantity of priority that is deleted in the graph G. Here, z represents the sum of the benefits d along a path selected on the graph G. Constraint set (5.4) indicates that the objective may not exceed the sum of every benefit of all the arcs of G. Constraint set (5.5) ensures that each section of the MPC-Heatmap is visited exactly once. Constraint set (5.6) implies that every robot that leaves a node will reach a new different node. Constraint set (5.7) represents a limit for the quantity of sanitization done in a node i that cannot be less of the priority assigned to the node i of the Graph G. Our approach to solving the multi-robot optimization problem is the hierarchical MILP as in [8, 33]. The MPC Server repeats the MILP calculus for every single robot in a sequence. At every step, it finds the path for one robot, puts to zero the nodes of the Graph G belonging to the path of the previous robot, and calculates the new path for the following robot. The MPC-Server repeats this method for every following robot. This way, the MPC Server considers different graphs for every robot to prevent two robots from selecting the same path, even if they start from the same node. The MPC Server calculates the value of the total reward of the whole team as the sum of the single rewards gained by every robot for each permutation of the sequence of the 4 robots and chooses the sequence with the higher result. MPC Server also considers a different deposit for

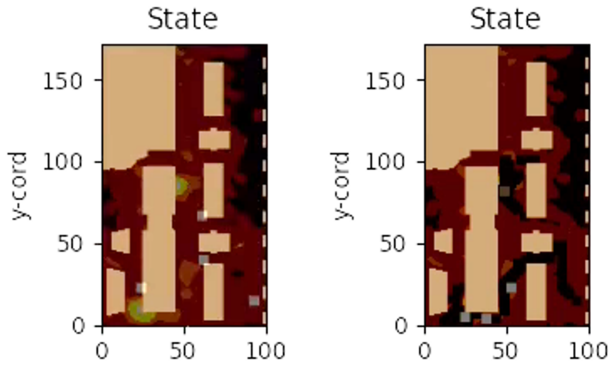


Figure 5.5. On the left, the distribution of priorities at the 6:00 a.m. of the 6 September 2021 in our simulation. On the right, the action of cleaning of 4 robots driven by HMILP-MPC control, and moving in the environment that represent Roma Termini station, thanks to the APF method.

every robot, positioned in different places of the Graph G and fixed at the beginning of the simulation. The assignment of a different deposit to more than one robot may be changed online by human intervention to correct the operation of cleaning by a remote control room. In Figure 5.5 we see the action of cleaning 4 robots, represented by the white squared dots. The cold zones, in black, are the path that they leave when they clean a zone of the environment. The environment is represented by the MPC-Heatmap of Roma Termini, where 1 pixel corresponds to 1 square meter.

5.2 Experiments and Comparison with MARL technique

In order to assess the performance of the system in a realistic dynamic environment, we propose a case study where the presented method is compared with the solution proposed in [12, 14]. Thanks to the collaboration with RFI, we received a Meraki Heatmap for every hour of the day representing the whole planimetry of the station between 30 August 2021 to 12 September 2021. We have verified that in the considered period, there

were not occurred exceptional events that may change the ordinary behavior of the station (for example, national or religious holidays, the opening of the schools etc). Moreover, the two weeks are temporally consecutive. For these reasons, there are no changes in the behavior of passengers that frequents the station. We have calculated the Euclidean distance between every MPC-Heatmap built at every different hour of the available days, and we have verified that the difference of the average Euclidean distance of the days that belong in the same category is lower than the ones in the other cases. In the available dataset, we can predict the distribution and the density of people in the station in a working day of the second week of data, using the average of the working days of the first week at the same hour. In order to assess the performance of the teams of robots, we define a simple metric c_perc as in [12, 14], representing the cleaning rate (in percentage) of the map as follows:

$$c\_perc = ((x_{tot} - s_{curr})/x_{tot}) \cdot 100, \tag{5.9}$$

where $x_{tot} = |X|$ is the number of free-obstacles cells of the map, while $s_{curr} = \sum_{(i,j)} s(i,j)$ is the sum of the current priorities for each cell of the heatmap. The value c_perc is than 100% if the map is totally clean (i.e., each cell has 0 priority) and 0% if every cell of the map has maximum priority. We compared the quality of cleaning for the day 6 September 2021, using for the HMILP the mean values of the heatmaps of the days: 31 August, 1, and 2 September 2021, considered working days in the RFI category definitions. As it is shown in Figure 5.6, the two methods have comparable results and approximately the same shape of the curve. They may be considered valid alternatives to the proposed problem.

Our solution has an important advantage with respect to the MARL method presented in [12, 14]. It has the possibility to be controlled by a remote site by a technical staff in a control room. This represents an essential innovation in the solving of the problem proposed by RFI because Roma Termini represents a challenging environment where exceptional events may occur and form a new aggregation of people. For this reason, it is required a solution that may permit correcting the activity of cleaning the robots, assigning greater priority to a specific area. This feature is enabled by exploiting the graph-based representation of the environment. We can online select for every robot a different deposit where they conclude their

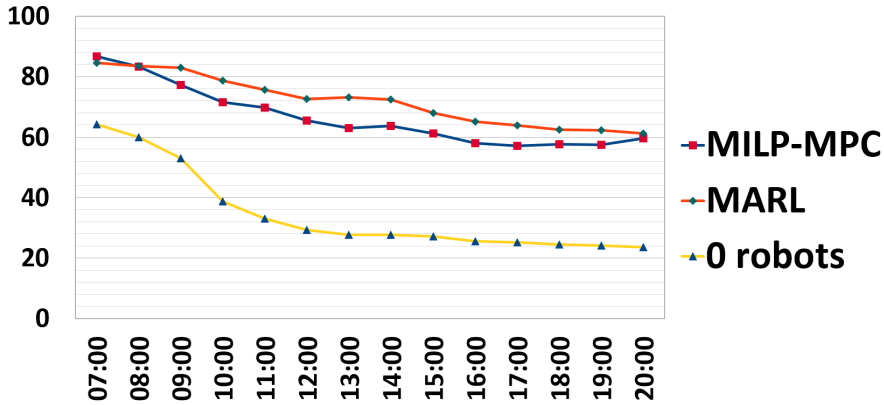


Figure 5.6. Simulation of sanitization of the day 6 September 2021. The first two curves represent *c_perc* for the two cases: MARL,HMILP-MPC. The yellow curve represents the spreading of the contaminants without any robot action.

mission, choosing between those available in the station. It is also possible to assign a gain to the nodes inside the subarea that we want to select. We have verified this possibility by simulating a situation in which it is occurred a rapid increment of priority risk in a zone Z where nodes 7, 10, and 13 are involved. We selected that zone Z, multiplying the weight of this node with a gain equal to 10. We also assigned to the robots the deposits in nodes 10 and 13, while in normal conditions, their deposits are assigned to nodes 2, 7, 9, and 11. In order to assess the performance on Z we defined a specific c_perc_z value similar to the one proposed in Equation 5.9. In this case, we set $x_{tot} = |Z|$ as the number of free-obstacle cells of the map in the zone Z, while $s_{curr} = \sum_{(i,j)\in Z} s(i,j)$ is the sum of the current priorities for each cell of the heatmap in Z. The results of the cleaning are shown in table 5.2, where we compare the case in which we have not selected any subarea (case 1) with the case in which we selected the zone Z (case 2). In column 2 and 3, there are the relative results of c_perc_z for the two cases at different times of day in the simulation. We can see that the team of case 2 demonstrates to sanitize better the zone Z, reaching a peak of cleaning at the 18:00 o' Clock where c_perc_z is equal

Table 5.2. Comparison between case 1 and case 2

Time	Case 1	Case 2
17:00	48.895	56.871
17:30	32.733	61.778
18:00	47.469	79.425
18:30	33.945	54.611

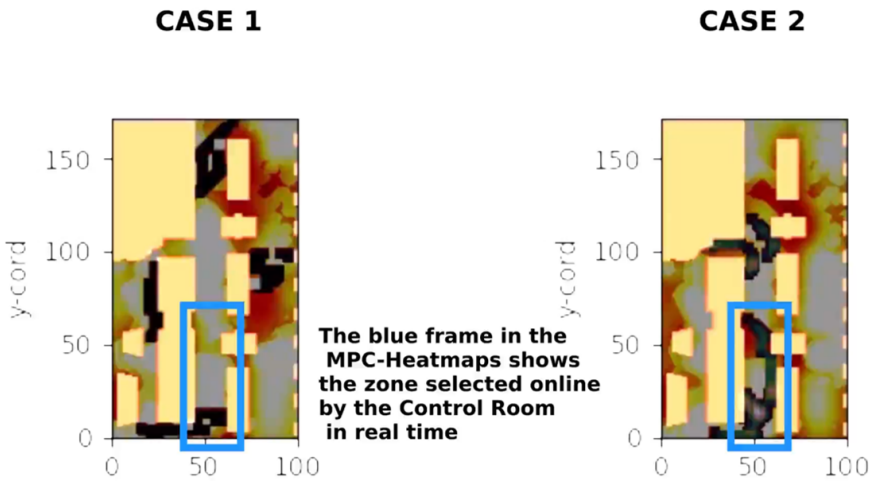


Figure 5.7. A snapshot of the simulation at time 17:30. The heatmap of Case 2 is cleaned at 61.778 percent. The Z zone is highlighted with blue borders.

to 79.425 %, in contrast to the 47.469% of the team of the case 1. A representative moment of the simulation is shown in Figure 5.7.

Chapter 6

Bioinspired Artificial Cockroach Colony Strategy combined with 2-type Fuzzy Logic for the Priority-Based Sanitization of Railway Stations

In the previous Chapters, we investigated on two different technologies to drive a team of robots, applying Deep Reinforcement Learning (Chapters 3 and 4) and MILP-MPC approach (Chapter 5). We empirically understand that there is the possibility to increment the cleaning performances studying for a strategy oriented to maximize the spreading of robots in the environment, reducing the mutual interferences and overlapping between the cleaning paths of the robots collaborating in the team. In this study [1], we propose a multi-robot online sanitization system that exploits the information about the position of people and combines the Bioinspired Artificial Cockroach Colony Strategy with the 2-type Fuzzy

[1]Study published in [94]. The source code of the system is available at the following link: https://github.com/Tavano1/MultiRobot_Sanitization_Railway

Logic to coordinate together a team of robot sanitizers. In particular, we were inspired by the social behavior of Cochroacks Colony during their infestation phase, where the insects explore the environment attracted by different food deposits and decide their paths considering the shadowed zones, the presence of pheromones, and maximizing distances by the other members of the colony. Moreover, we have considered a central server that acts like a Daemon as defined in [31]. The Server is responsible for segmenting the heatmap in several subareas and assigning a different subarea to each robot, using the taboo search mechanism and 2-type Fuzzy Logic. The Server selects a subarea considering the density of priority inside it, the distance of it from the robots, and the number of robots just present inside the considered subarea. The objective of the server is to assign each robot to a different subarea, preferring one not chosen before, with a higher possible concentration of priority, and maximizing the spreading of the colony in the heatmap. The selected subarea will be an attractive destination for the assigned robot, which considers it as input in its autonomous building algorithm of its own cleaning path in the environment. Robots, thanks to the 2-type Fuzzy Logic and the implemented pheromones communication, select their steps at every instant of time, preferring the zones where there are higher values of priorities and higher distances from the other members of the colony and their pheromones. These characteristics of the algorithm facilitate the spreading of the robots' team inside the environment, increasing the quality of the cleaning action. The solution's performances are compared with those of four different methods deployed in the same scenario, using real data shared by RFI S.p.A., showing better results. The background, the related works, and the reasons for the technical choices regarding this study are discussed in Chapter 2, in Section 2.5. The rest of the Chapter is structured as follows. In Section 6.1 we describe the architecture of the proposed framework; in Section 6.2, we focus on the comparison with the MARL technique presented in [12, 14] described in Chapter 3, the MILP-MPC technique [15], presented in Chapter 5 and two cases of Coverage Path Planning techniques proposed in the literature for sanitization task [61, 68].

6.1 Architecture of the System and the Environment

The architecture we propose for sanitizing the railway stations is shown in Figure 6.1. The server has the role of the daemon as defined in [31, 83]. It is responsible for building the heatmap, storing the information about the distribution of the pheromones in the environment in a different gridmap that we call PheromonesMap, building a graph representation of the environment, and assigning to each robot a different destination node. In this manner, it assigns every (cockroach-like) agent with a different attractive contaminated destination zone thanks to the 2-type Fuzzy logic. Every robot will receive this data, and at every timestep, it decides its next area to be cleaned in the gridmap thanks to the 2-type Fuzzy system. The heatmap is a gridmap whose hot/cold points represent high/low priority areas to be sanitized depending on the estimated distribution of people in the environment. Each matrix element of the heatmap is a real number in the interval $[0, 1]$, where 1 stands for the maximum priority, while 0 means that no cleaning is needed (see heatmap in Figure 6.2 (left), where this matrix is displayed as a color-coded image, black pixels are 0 priority areas, while colors from red to yellow are for increasingly higher priorities). The heatmap (cleaning priority) is computed from the position of a group of people (clusters) by modeling the possible spreading of viruses or bacteria using a Gaussian model of dispersion [42]. Specifically, we exploit the periodic convolution of a Gaussian filter $\mathcal{N}(\mu, \sigma^2)$ at every step, where μ, σ^2 are suitable parameters that can be regulated depending on the meters/pixels ratio, the timestep, and the considered typology of spreading. In our case, we set the μ and σ values according to the spreading parameters proposed in [43, 11]. The values of the used parameters μ, σ^2 are respectively 0 and 0.9, with a diameter of spreading equal to 5 pixels as in [15, 14, 13]. The heatmap after the Gaussian filter application is shown in Figure 6.3, (d)). More formally we define the terms M, S, X and K and Ac (set of the possible actions a_i of a robot) as presented in the Section 3.1. A graphical representation of the environment is shown in Figure 6.3. Using the planimetry shared by Rete Ferroviaria Italiana, we selected a region of 100×172 meters (orange rectangle in 6.3 (a)). The areas of shops, restaurants, and walls represent the fixed obstacles on the map. From this

portion, we realize a black and white gridmap (6.3 (b)). We consider that the WiFi server receives an updated distribution of people every 1 hour from the Meraki Cisco System (an example of Meraki heatmap is in Figure 6.3 upside). The information from the Meraki is then converted into a heatmap for the robots by associating each location with a priority value proportional to the density of people. Since the information from the Meraki is georeferenced, the retrieved value is finally rotated, translated, and scaled in order to match the reference frame of the robots (see Figure 6.3 (c)). In our simulations, the heatmap has a dimension 100×172 pixels, with a resolution of 1 pixel per $1m^2$, and the timestep corresponds to 1 minute so that the Meraki will share its information about the presence of people every 60 timesteps. Agents can move by one pixel in any direction. Hence we define the set A that includes 8 actions a_i (4 linear and 4 diagonal). At every timestep, we consider that a robot will sanitize an area of 9 pixels, corresponding to a surface of $3m^2$ of the station.

The server is responsible for receiving, at every timestep, the current position from every robot. It verifies people's movements in the environments thanks to the information received from Meraki Cisco System WiFi Infrastructure. At Every hour (60 timesteps in our simulation), the server builds the heatmap considering the received information about the people's distribution. At every timestep, a Gaussian filter is applied to the heatmap to simulate the effects of the spreading and attenuation of the contaminants inside the environment. The server also updates the information about the presence of pheromones on a gridmap named PheromonesMap (Ph in the Algorithm 4 and 5), with dimensions 100×172 pixels like the heatmap, applying values 1 to the cells corresponding to the 3×3 robot's cleaned area of the gridmap, around the robot's position at every robot's step. The pheromone is represented by a real number in the range $[0, 1]$. It has the value of 1 at the first moment it is applied in PheromonesMap. An example of PheromonesMap is shown in Figure 6.2 (right side). At every step, the server implements pheromone evaporation, the process by which the pheromone trail intensity on the grids automatically decreases over time. Specifically, the server at every timestep subtracts a constant value equal to 0.1 at every non-zero cell of PheromonesMap. The heatmap and the PheromonesMap are sent to every robot of the team at every timestep. At every 60 timesteps, the server also builds a graph representation of the

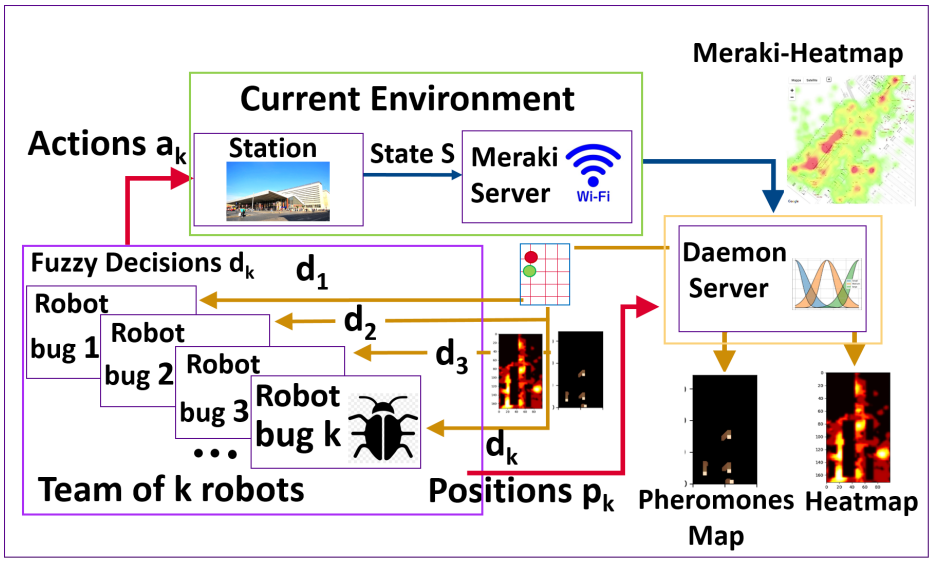


Figure 6.1. Graphical representation of the proposed architecture including multiple robotic agents (left) and a single server (the Daemon Server on the right) exploiting WiFi data to generate a heatmap of priorities (red to yellow spots) and the destination nodes for each robot thanks to the Fuzzy Logic System. It updates also the PheromonesMap.

,

environment, $G = (N, Ar)$ where $N = \{0, 1, \ldots, i\}$ is the set of vertexes and $Ar = \{(i, j) : i, j \in N, i \neq j\}$ is the set of arcs. The server uses the heatmap s to realize G. Specifically, it provides a partition with fixed steps of 40 pixels along the coordinates x and y of s (Figure 6.3 (e)) to obtain 15 different zones as in [13]. A different node then represents a single zone. The server calculates the total amount of priority inside every single zone $q_i \in Q$ where $i \in N$ and Q is the set of q_i for each zone ($\mathbf{card}(Q) = \mathbf{card}(N)$). Every node is associated with the q_i of its relative zone. Each node is connected to the others. Every arc has a cost $d_{i,j}$ with $(i, j) \in Ar$: the Euclidean distance between the centers of the zones in the heatmap represented by the starting and destination nodes in G. The server uses a 2-type Fuzzy decision layer with taboo search [88, 3] to establish the destination nodes $dest_k \in Dest$ of each robot k, where $k \in K$ and

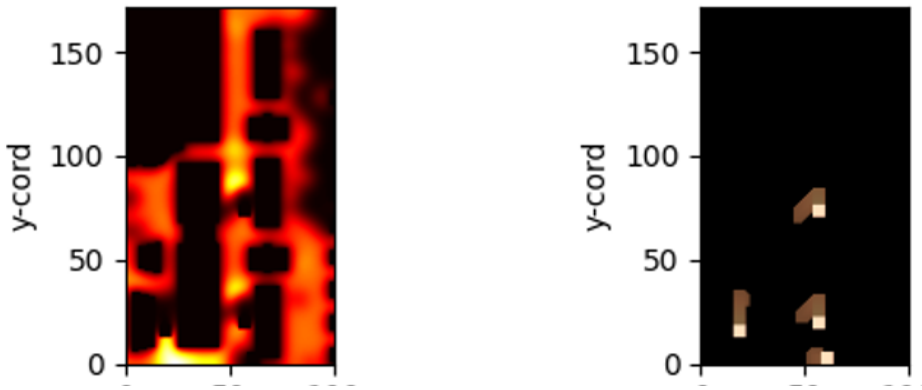


Figure 6.2. Example of Heatmap (left) and PheromonesMap (right) during the sanitization process. The yellow lines in PheromonesMap represent the deposited pheromones along the paths of 4 robots in the gridmap.

Dest is the set of all destination nodes with cardinality **card**(K). Every robot receives its destination node at every 60 timesteps. The activities of the server are described in the Algorithm 4. In the simulation, the term *max_step* (line 2) is a parameter that we set to 780, considering 13 hours of real data and a duration of 1 minute for a timestep. The server, at every timestep, receives the positions of the robots (line 3). At every 60 timesteps: the server rebuilds the heatmap (line 5), adding new priorities in the gridMap M depending on the movements of people, builds the Graph (line 6), puts to zero the cells of the PheromonesMap for its initialization (line 7), verifies the numbers of robots inside each node (line 8), applies the 2-type Fuzzy system with taboo search to obtain the destination nodes for each robot (line 9), sends the destination node to every robot (line 10). At every timestep: the server updates the PheromonesMap (line 12) applying the pheromones to the current positions of the robots and implementing the evaporation process, updates the heatmap applying the Gaussian filter and puts to 0 the pixels of the zones currently cleaned by the robots (line 13), and sends to the robots the updated heatmap, the updated PheromonesMap, and their current positions of the robots to all the team (lines 14-16). The set of the 2-type Fuzzy decision system exploited by the server is shown in Table 6.1. The Fuzzy system is executed to evaluate each graph node in order to find for every robot $k \in K$ its

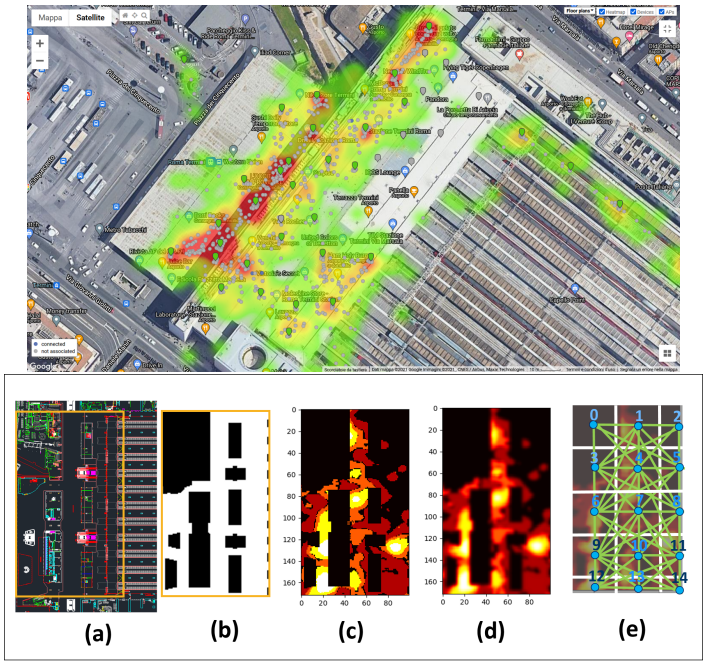


Figure 6.3. Example of the distribution of people inside the Roma Termini station as retrieved from the Cisco Meraki WiFi network (up) and the process of the realization of the heatmap and the relative full connected graph (down): the planimetry of the station Roma Termini shared by Rete Ferroviaria Italiana S.p.A. (a), the black-white image (b); the heatmap (d); the heatmap after the Gaussian convolution filter applied for 50 steps (d); partitioning of the heatmap to obtain the corresponding graph (e).

destination node $dest_k \in Dest$. It receives, as input 1, $d_{i,j,k}$, that is $d_{i,j}$ with $(i,j) \in Ar$, the distance between a node i in N with the current node j where the specified robot k is positioned. Input 2 is the concentration of the priorities for the node q_i with $i \in N$. Input 3 is $nr_i \in Count\_Rob$, the number of robots currently situated in the node i, where $Count\_Rob$ is the set of the numbers of robots per node with dimension $\mathbf{card}(N)$. The Fuzzy system gives as output a real value in the range $[0,1]$, which represents how much the considered node i is convenient such as destination node $dest_k$ for a determined robot k. The $\mathbf{card}(N)$ evaluations of the Fuzzy system for a robot k are collected in a list, and $dest_k$ will be the

maximum value of the list. Specifically, the Fuzzy rule base is designed for the Fuzzy system to assign a higher real-value output for a node i, considering the robot k, the higher the priority density q_i, the lower the number of robots nr_i to favor the spread of the colony, and the lower the number of $d_{i,j,k}$ to reduce the crossing distance from the robot and its next destination.A taboo search approach is developed to reduce the possibility that more robots are assigned with the same destination node as in [88, 3]. In particular, after a node i is assigned with a specific robot k and becomes $dest_k$, the server assigns in G to that chosen node i very high values of distances $d_{i,j}$ in Ar for each $j \in N$, $j \neq i$ and sets a value 0 to priority's concentration q_i in the next Fuzzy evaluation lists for the remaining robots different than k. The taboo search with the cycles of Fuzzy evaluations is repeated changing the robots' order in its execution, considering all the possible permutations without repetitions $p = \mathbf{card}(K)!$. The server obtains p different $Dest$, and selects that with the higher sum of q_i for each node $i \in Dest$. In the Fuzzy system, For every input, we define a different domain that represents the universe of discourse, as shown in Table 6.1. For each domain, we report the minimum, the maximum value, and the number of samples generated for it (Min, Max, Num) [41]. We created then for each input three different Gaussian intervals type-2 Fuzzy sets denominated Small, Medium, and Large ($SmlH$, $MedH$, $LrgH$ with H the number of input in the Tables) with uncertain standard deviation values [41] as illustrated in the Table 6.1. For each Gaussian set, we report in the table the parameters, consisting of the mean, the standard deviation center, the standard deviation spread, and the height of the set (respectively $Mean$, Std_Center, Std_Spread, $Height$). For the output, we defined one domain and ten different Gaussian intervals type-2 Fuzzy sets with uncertain standard deviation values. The sets and their parameters are illustrated in the same table 6.1. So we have defined our rules as reported in Table 6.2. We used an interval type-2 Mamdani Fuzzy inference system. The type reduction plays a key role in achieving crisp values from Type-2 Fuzzy sets. In our case, we used the type reduction algorithm: Enhanced Iterative Algorithm with Stopping Condition [41]. At every step, each robot k decides autonomously which zone $x_{r,l}$, with r, l coordinates in the gridmap M, is necessary to be cleaned inside the heatmap s. The process is described in Algorithm 5. At the beginning of the simulation,

every robot is located at random position as in [14, 13, 15] (line 2). The robot k receives the new destination node $dest_k$ at the beginning of the hour, at every 60 timesteps (line 5). At every minute, it receives the updated heatmap s, the PheromonesMap Ph from the server, and the current positions of the other robots in the map $(x_1, \ldots, x_k)$ (lines 7, 8, and 9). Every robot k will decide autonomously at every timestep its next position x_{new}. It may choose one of 8 different zones $x'_{r,l}$, because it may do 8 different chess-like movements $ai \in A$: up, down, left, right, and towards the diagonal positions. To make its decision, it finds an evaluation output as a real value in the range $[0, 1]$ by the 2-type Fuzzy Logic for each possible next step. The Fuzzy decision system developed for the robot uses the following inputs: the amount of pheromone $Amount\_pher$ (input 1) considering 3×3 grid-zone around its next possible position choice $x'_{r,l}$ in the PheromonesMap, the total amount of priority $Amount\_prior$ (input 2) in the 3×3 grid-zone around its next possible position selection $x'_{r,l}$ in the heatmap, $dist\_dest_k$ (input 3), defined as the Euclidean distance in the gridmap M of the new considered position $x'_{r,l}$ from the center of the heatmap's zone represented by the destination node $dest_k$. Moreover, the robot evaluates the distances of the new considered position $x'_{r,l}$ from the current position $(x_i, \ldots, x_k)$ of the other robots and selects the minimum of them, $Dist\_x_{min}$ (line 10), that represents the input 4. We have studied a rule base in order to obtain a higher evaluation number if the higher is the total amount of priority $Amount\_prior$ to maximize the removal of priorities at every step, if the lower is the amount of pheromone $Amount\_pher$, where higher concentration corresponds to a less attractive movement selection, if the lower is the value $dist\_dest_k$, and if the higher is $Dist\_x_{min}$ to maximize the spread of the colony. The decision is taken thanks to the 2-type Fuzzy Logic process and the taboo search. Thanks to the 2-type Fuzzy logic, we list an evaluation output for every next new possible position selection x' (one of 8 positions corresponding to 8 chess-like possible movements $a_i \in A$). The list's maximum value will be the robot's next new position x_{new} (line 11). It applies the action to the environment (line 12). Finally, it sends the new position to the server (line 13). For every input, we defined a different domain that represents the universe of discourse, as shown in table 6.3. We then created, for each input, three different Gaussian interval type 2 Fuzzy sets denominated Small,

Algorithm 4 Server-side cockroach colony algorithm

procedure SERVER(k, M)
  for $stp < max\_step$ **do**
    $(x_1, \ldots, x_k)$ = receive_positions()
    if $stp == 0\, OR\, stp\%60 == 0$ **then**
      s = apply_real_recorded_data(M,$ClusterPeople$)
      G,N,Ar,Q = build_graph(M, s)
      Ph = initialize_Pheromones_Map(M)
      $Count\_Rob$ = N_Robs_Per_Node(M,G,N,$(x_1, \ldots, x_k)$)
      $Dest$ = Fuz_Tab_S(G,N,Ar,Q,$(x_1, \ldots, x_k)$,$Count\_Rob$)
      send_Destination_Nodes($Dest$)
    end if
    Ph = update_Pheromones_Map(M,$(x_1, \ldots, x_k)$)
    s = update_heatmap(s, $\mathcal{N}(\mu, \sigma^2)$, M,$(x_1, \ldots, x_k)$)
    send_heatmap(s)
    send_Pheromones_Map(Ph)
    send_Current_Robots_Positions($(x_1, \ldots, x_k)$)
  end for
end procedure

Medium, and Large ($SmlH$, $MedH$, $LrgH$ with H the number of input in the Table) with uncertain standard deviation values [41]. For the Gaussian sets, we report in the Table the parameters in Table 6.3 as in the server case. Also, we defined the output's domain and ten different Gaussian intervals type-2 Fuzzy sets with uncertain standard deviation values as in the case of the server-side Fuzzy system, whose values are shown in Table 6.1. So we have defined our rule base as reported in Table 6.4. We used an interval type-2 Mamdani fuzzy inference system and the type reduction algorithm: Enhanced Iterative Algorithm with Stopping Condition [41] as in the server case.

6.2 The Case Study

In this Section, we compare our approach with four different methods studied for the sanitization task. Thanks to the collaboration with

Algorithm 5 Agent-side cockroach colony algorithm.

```
1: procedure AGENT(i, M)
2:     x_i = random_position()
3:     for stp < max_step do
4:         if stp == 0 OR stp%60 == 0 then
5:             dest_i = receive_destination_Node()
6:         end if
7:         s = receive_heatmap()
8:         Ph = receive_Pheromones_Map()
9:         (x_1, ..., x_k)=receive_positions()
10:        Dist_x_min=min_distance((x_1, ..., x_k))
11:        x_new = select_action2typeFL(s, x_i,dest_i,Dist_x_min,Ph)
12:        emulate_action(s, x_new, M)
13:        send_position(x_new)
14:    end for
15: end procedure
```

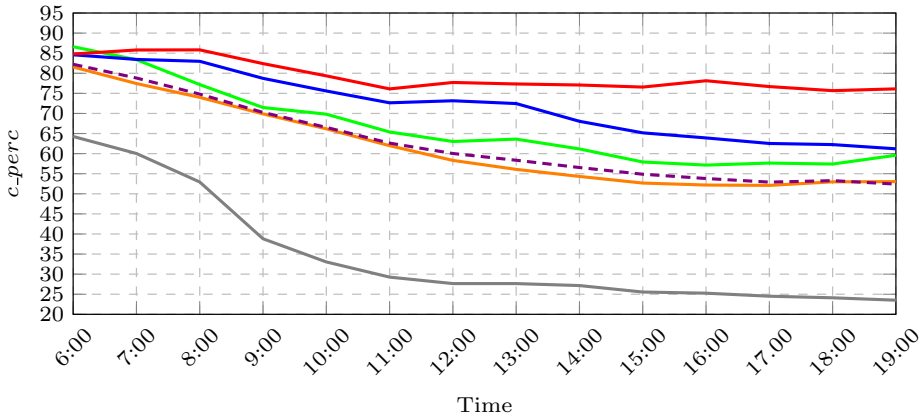


Figure 6.4. Comparison between the proposed Bio-inspired Cockroach Colony method (red) and the alternative MARL framework (blue), MPC-MILP technique (green), the spiral-based (violet dashed line) and boustrophedon-based (orange) CPP ones considering c_perc value. In all settings, a team of 4 robots is deployed. The grey line shows the behavior of c_perc with zero robots.

Table 6.1. Gaussian interval type-2 Fuzzy sets for the inputs and the output of the Server-side 2-type Fuzzy Logic System

Var	*G_Set*	*Domain*			*Mean*	*Std_C*	*Std_Spr*	*H*
		Min	*Max*	*Num*				
Input 1	Sml1	0	150	1000	0	8	5	1
	Med1	0	150	1000	75	8	5	1
	Lrg1	0	150	1000	150	8	5	1
Input 2	Sml2	0	450	1000	0	45	15	1
	Med2	0	450	1000	225	45	15	1
	Lrg2	0	450	1000	450	45	15	1
Input 3	Sml3	0	4	1000	0	0.15	0.05	1
	Med3	0	4	1000	2	0.15	0.05	1
	Lrg3	0	4	1000	4	0.15	0.05	1
output	LB	0	1	1000	0	0.02	0.01	1
	LM	0	1	1000	0.11	0.02	0.01	1
	LA	0	1	1000	0.22	0.02	0.01	1
	MB	0	1	1000	0.33	0.02	0.01	1
	MM	0	1	1000	0.44	0.02	0.01	1
	MA	0	1	1000	0.55	0.02	0.01	1
	HB	0	1	1000	0.66	0.02	0.01	1
	HM	0	1	1000	0.77	0.02	0.01	1
	HA	0	1	1000	0.88	0.02	0.01	1
	HAA	0	1	1000	1	0.02	0.01	1

Table 6.2. Rule base for the Server-side 2-type Fuzzy Logic System

*Input*1	*Input*3	*Input*2	*Output*	*Input*1	*Input*3	*Input*2	*Output*
Sml1	Sml3	Sml2	LA	Sml1	Med3	Sml2	LA
Med1	Sml3	Sml2	LM	Med1	Lrg3	Sml2	LM
Lrg1	Med3	Sml2	LB	Lrg1	Lrg3	Sml2	LB
Sml1	Med3	Med2	MA	Sml1	Sml3	Med2	MA
Med1	Lrg3	Med2	MM	Med1	Sml3	Med2	MM
Lrg1	Lrg3	Med2	MB	Lrg1	Med3	Med2	MB
Sml1	Sml3	Lrg2	HAA	Sml1	Med3	Lrg2	HA
Med1	Sml3	Lrg2	HM	Med1	Lrg3	Lrg2	HM
Lrg1	Med3	Lrg2	HB	Lrg1	Lrg3	Lrg2	HB

Table 6.3. Gaussian interval type-2 Fuzzy sets for the inputs and the output of the Agent-side 2-type Fuzzy Logic System

Var	*G_Set*	*Domain*			*Mean*	*Std_C*	*Std_Spr*	*H*
		Min	*Max*	*Num*				
Input 1	Sml1	0	9	1000	0	0.8	0.4	1
	Med1	0	9	1000	4	0.8	0.4	1
	Lrg1	0	9	1000	9	0.8	0.4	1
Input 2	Sml2	0	9	1000	0	0.8	0.4	1
	Med2	0	9	1000	4	0.8	0.4	1
	Lrg2	0	9	1000	9	0.8	0.4	1
Input 3	Sml3	0	200	1000	0	15	10	1
	Med3	0	200	1000	100	15	10	1
	Lrg3	0	200	1000	200	15	10	1
Input 4	Sml4	0	200	1000	0	15	10	1
	Med4	0	200	1000	100	15	10	1
	Lrg4	0	200	1000	200	15	10	1

Rete Ferroviaria Italiana, we received an entire day of recording of the Meraki system (6 September 2021) to be exploited for our experiments. The results are shown in Figure 6.4. The first method (orange line) is a spiral-based [61] exploration of the environment, divided into k non-overlapping sub-areas of equal dimensions, each assigned to a single robot. Sub-areas are partitioned by rectangular decomposition, and a spiral path is used to cover such partitions. The second method (dashed violet line) is boustrophedon-based [68] exploiting a modified version of the boustrophedon environment decomposition for partitioning and boustrophedon paths are employed for coverage. The third method (blue line) uses a model-free Deep Reinforcement Learning (MARL) [12, 14], that we have presented in Chapter 3. The fourth method (green line) is based on a distributed method based on the Hierarchical MPC-MILP approach combined with the Artificial Potential Fields technique [13], described in Chapter 4. It considers the information about the railway transportation service (arrivals and departures of the trains in the station) together with the historical data regarding the movements of people inside the environment stored by Meraki Cisco System to predict the future cluster's positions. The grey line represents the priorities' evolution caused by the contaminants' spread and

attenuation following the Gaussian diffusion law, considering zero robots. Every method is applied considering a team of 4 robots. To compare these methods, we defined the following metric c_perc [15, 14, 13] that represents the cleaning rate (in percentage) of the map as follows:

$$c\_perc = ((x_{tot} - s_{curr})/x_{tot}) \cdot 100 \tag{6.1}$$

where the term s_{curr} is the total amount of priority that the team of robots has removed, and the term x_{tot} represents the total free-obstacle area of the environment. As we can see in Figure 6.4, the Cockroach Colony approach (red line) overcomes the other methods. The average of c_perc in the day is 79.26 for the Cockroach method, while it is 71.91 for the MARL. The methods based on the movements of the robots with fixed shapes (spiral-based and boustrophedon-based) are less performing because their sanitization technique does not take into account the evolution of the contamination in a dynamic environment. Moreover, the proposed Cockroach Colony approach outperforms the MARL method every hour, especially after 10:00, arriving to gain 15% at 19:00. In the proposed method, robots generally don't clean the same zones during the action of cleaning. There are fewer overlapping in their paths following the updating of contaminant concentration depending on the movements of people during the day. They remain well distributed on the map because the WiFi server, at every hour, gives each of them a different target zone to be reached for the cleaning. Each robot deposits its repulsive pheromone on the path, so there is less interference between the members at every step while they move, attracted by a higher concentration of priorities.

Table 6.4. Rule base for the Agent-side 2-type Fuzzy Logic System

*Inp*1	*Inp*3	*Inp*2	*Inp*4	*Outp*	*Inp*1	*Inp*3	*Inp*2	*Inp*4	*Outp*
Sml1	Sml3	Sml2	Lrg4	LA	Sml1	Sml3	Lrg2	Sml4	HAA
Med1	Sml3	Sml2	Lrg4	LM	med1	Med3	Sml2	Sml4	HM
Lrg1	Med3	Sml2	Lrg4	LB	Lrg1	Med3	Sml2	Sml4	HB
Sml1	Med3	Med2	Lrg4	MA	Sml1	Lrg3	Sml2	Lrg4	LA
Med1	Lrg3	Med2	Med4	MM	Med1	Lrg3	Med2	Lrg4	LM
Lrg1	Lrg3	Med2	Med4	MB	Lrg1	Sml3	Med2	Lrg4	LB
Sml1	Sml3	Lrg2	Med4	HA	Sml1	Sml3	Med2	Lrg4	MA
Med1	Sml3	Lrg2	Med4	HM	Med1	Med3	Lrg2	Med4	MM
Lrg1	Med3	Lrg2	Sml4	HB	Lrg1	Med3	Lrg2	Med4	MB
Sml1	Med3	Sml2	Sml4	LA	Sml1	Lrg3	Lrg2	Med4	HA
Med1	Lrg3	Sml2	Sml4	LM	Med1	Lrg3	Sml2	Med4	HM
Lrg1	Lrg3	Sml2	Sml4	LB	Lrg1	Sml3	Sml2	Sml4	HB
Sml1	Sml3	Med2	Lrg4	MA	Sml1	Sml3	Sml2	Sml4	LA
Mdm1	Sml3	Med2	Lrg4	MM	Med1	Med3	Med2	Sml4	LM
Lrg1	Med3	Med2	Lrg4	MB	Lrg1	Med3	Med2	Sml4	LB
Sml1	Med3	Lrg2	Lrg4	HA	Sml	Lrg3	Med2	Lrg4	MA
Med1	Lrg3	Lrg2	Med4	HM	Med1	Lrg3	Lrg2	Lrg4	MM
Lrg1	Lrg3	Lrg2	Med4	HB	Lrg1	Sml3	Lrg2	Lrg4	MB
Sml1	Sml3	Sml2	Med4	LA	Sml1	Sml3	Lrg2	Lrg4	HA
Med1	Sml3	Sml2	Med4	LM	Med1	Med3	Sml2	Med4	HM
Sml1	Med3	Sml2	Sml4	LB	Lrg1	Med3	Sml2	Med4	HB
Sml1	Med3	Med2	Sml4	MA	Sml1	Lrg3	Sml2	Med4	LA
Med1	Lrg3	Med2	Sml4	MM	Med1	Lrg3	Med2	Med4	LM
Lrg1	Lrg3	Med2	Sml4	MB	Lrg1	Sml3	Med2	Sml4	LB
Sml1	Sml3	Lrg2	Lrg4	HA	Sml1	Sml3	Med2	Sml4	MA
Med1	Sml3	Lrg2	Lrg4	HM	Med1	Med3	Lrg2	Sml4	MM
Lrg1	Med3	Lrg2	Lrg4	HB	Lrg1	Med3	Lrg2	Sml4	MB
Sml1	Med3	Sml2	Lrg4	LA	Sml1	Lrg3	Lrg2	Lrg4	HAA
Med1	Lrg3	Sml2	Med4	LM	Med1	Lrg3	Sml2	Lrg4	HM
Lrg1	Lrg3	Sml2	Med4	LB	Lrg1	Sml3	Sml2	Lrg4	HB
Sml1	Sml3	Med2	Med4	MA	Sml1	Sml3	Sml2	Lrg4	LA
Mdm1	Sml3	Med2	Med4	MM	Lrg1	Lrg3	Lrg2	Sml4	MB
Lrg1	Med3	Med2	Sml4	MB	Sml1	Lrg3	Lrg2	Sml4	HA
Sml1	Med3	Lrg2	Sml4	HA	Med1	Sml3	Lrg2	Sml4	HM
Med1	Lrg3	Lrg2	Sml4	HM	Lrg1	Sml3	Sml2	Sml4	HB
Lrg1	Lrg3	Lrg2	Sml4	HB	Sml1	Med3	Sml2	Lrg4	LA
Med1	Med3	Med2	Med4	LM	Med1	Med3	Sml2	Lrg4	LM
Lrg1	Med3	Med2	Med4	LB	Lrg1	Lrg3	Med2	Lrg4	LB
Sml1	Lrg3	Med2	Med4	MA	Sml1	Lrg3	Med2	Lrg4	MA
Med1	Lrg3	Lrg2	Med4	MM	Med1	Sml3	Med2	Med4	MM
Lrg1	Sml3	Lrg2	Sml4	MB					

Chapter 7

Conclusions

In recent years, we assisted in the rapid diffusion in the world of serious illnesses, as in the case of Covid-19 disease. To respond to the necessity of sanitization of common spaces such as malls and railway stations, different adaptive strategies to drive a team of robots for the sanitization of large indoor dynamic environments are proposed, studied, and compared together. We decided to test our solution in a particular real case: the main railway station of the Capital of Italy, Roma Termini. The station is equipped with a WiFi network that collects anonymous information about passengers in the station during the day. Our approach uses this information to localize passengers inside the station and to develop a map of possible risky areas to be sanitized. In particular, the information about the presence of people aggregations is organized as a heatmap where a wide range of colors in the map, from red to yellow and white, represents different levels of contamination of the zones. The areas with a high presence of people were marked with light colors (yellow and white). These marked zones correspond to the higher priorities places where it is more urgent to clean. Thanks to the collaboration with Italian Railway Infrastructure Manager Rete Ferroviaria Italiana S.p.A., we have tested our solution with real data captured and stored by the WiFi Network of the station Roma Termini. In Chapter 3, we proposed a scalable Deep Q-Learning approach to multi-robot sanitization of railway stations. The proposed framework exploits real-time data about the distribution of people in the environment to generate a priority map of the areas to be sanitized. Such a map is then

exploited by a set of robots, each endowed with a convolutional neural network, to learn an effective high-level strategy that optimizes the overall sanitization processes. We tested the proposed framework in a realistic simulated scenario, which was designed in cooperation with the Italian railway infrastructure manager (RFI S.p.A.), considering the largest and most populated Italian railway station (Roma Termini) as a case study. We assessed the performance of the proposed framework in different case studies, firstly by considering a worst-case scenario where random clusters of people are scattered along the station, then by considering a more realistic setting in which the distribution of people is retrieved from a one-day data recording provided by the Meraki Cisco System WiFi Network of Roma Termini. The collected results show that the proposed framework is capable of generating long-term strategies for a team of robots in order to suitably sanitize large indoor environments, such as railway stations. We also discussed the scalability of the proposed method with respect to the number of involved robots and the density of people in the station. Finally, we illustrated the approach's effectiveness with respect to alternative CPP-based methods for robot cleaning. As a limitation of the approach, we can observe that the time needed to train multiple DQNs for our MARL framework increases with the number of robots (each new robot is associated with a new DQN). Notwithstanding the offline and distributed training process, such increment may impair the deployment of the framework in very large robotic teams. To mitigate this problem, the networks can be distributed on different machines - our DTDE approach is particularly suited for such a solution - and epsilon-greedy mechanisms [47] may be deployed for action selection instead of the current uniform sampling. In our study, we focused on the generation of cleaning policies, assuming that the robots were always able to reach and completely clean target areas. On the other hand, navigation and cleaning failures (e.g., due to the presence of people or unexpected obstacles) should also be addressed in a real-world implementation. In future work, we plan to deploy and assess the proposed approach in more complex scenarios where also cleaning failures and social navigation issues are considered. To this end, routing algorithms [45, 100, 34] and human action recognition methods [5, 59, 46, 104, 35] will be investigated and integrated. We are also interested in learning methods to adapt the dimension and the strategies

of the robot team with respect to the heatmap status in order to reduce power consumption and maintenance costs while preserving the quality of the sanitization. In Chapter 4, we continue our study regarding MARL strategies, proposing an extension of the framework described in Chapter 3. In particular, we deployed a DTDE (Decentralized Training, Decentralized Execution) scalable Deep Q-Learning approach to train a cooperative team of heterogeneous robots, each one equipped with its own convolutional neural network, for the execution of a shared sanitization task. The robots are divided into different typologies that are distinguished for several characteristics, such as the traverse speed of the environment, the dimensions, and the shapes of the cleaning areas that are sanitized after every step movement. This study was promoted by the Italian Railway Infrastructure Manager Rete Ferroviaria Italiana, who asked for a framework in which it is possible to set as parameters the technical specifics of any robot. It is so possible to study a common strategy in which robots of different brands or generations work together, adapting their different policies to the common cleaning task. To evaluate the framework, we proposed a comparison between heterogeneous and homogeneous teams of robots in a realistic domain obtained from a dataset that collects 1 day of data recording of the WiFi network of the Roma Termini railway station. The empirical results show that, despite the additional complexity given by heterogeneity, the heterogeneous system behaves similarly to the homogeneous version of the approach in terms of training convergence. On the other hand, the heterogeneous system works better than the homogeneous one in restricted and populated environments, while the cleaning performance on the overall map is slightly reduced compared to the homogeneous case. As a future work, we plan to investigate larger robot teams equipped with more complex high-level heuristics suitable for increasingly larger real-world environments. The MARL frameworks discussed in Chapters 3 and 4 only use current information about the positioning of people's clusters inside the station and the current distribution of the priorities inside the heatmap. Continuing our investigation about the sanitization strategies, We verified the possibility of using historical data about the positions of people recorded by the WiFi infrastructure Network Meraki and the knowledge about the transportation service in the considered railway station to predict the movement of clusters of passengers with a horizon

time of 1 hour. Following this purpose, in Chapter 5, we presented a new technique based on the Hierarchical MILP-MPC method with APF for the on-line cleaning of very large and dynamic environments such as a railway station. In particular, we present a multi-robot distributed approach for the cleaning task of a large and dynamic environment, where the decisions about the cleaning path are given by the participation of two decision layers of the process: in the first level, every robot receives a list of subareas which is necessary to traverse respecting the assigned order to clean them in accord to the concentration of their priorities, decided by a common centralized MPC-Server thanks to the Model Predictive Control Method; at a second step, each robot builds its path through the indicated target zones thanks to the Artificial Potential Fields Technique (APF). Every robot autonomously finds the peaks of priorities inside its assigned destination subareas and reaches them, building its path with the APF technique. We have compared our solution with the MARL technique proposed in Chapter 3, highlighting the advantages derived by the use of a graph and the MILP: the on-line control of the sanitization strategy from remote human staff. We have also tested the quality in a real case simulation with real data from the Roma Termini Station, the largest and most populated station in Italy. In future work, we plan to test the solution with real robots and investigate how the proposed MPC approach can be integrated into a MARL framework to improve the cleaning performance. It is also interesting to verify the possibility of using the MPC method in the process of training the MARL solution. Our investigation on sanitization strategies continued, considering the idea of finding a method that may increment the cleaning performances of the teams, setting rules for team coordination to promote the spreading of the robots in the environment, reducing, as a consequence, the interference and overlap between the cleaning paths of the robots, in particular cleaning repetition of same neat areas by different robots. Following this idea, in Chapter 6, we proposed a new strategy for on-line sanitization of railway stations based on a bio-inspired Cockroach Colony Strategy combined with 2-type Fuzzy logic decision systems in a dynamic environment. In particular, in this method, we investigate the use of a strategy based on the social behavior of the insects during their phases of dynamic exploration and infestation of a large environment. The robots spread alone in different regions and use artificial pheromones to alert the

other colony members about their presence and the areas they just explored. The pheromones act as repulsive signals. Consequently, the robots decided to be far from the pheromones of the others. Every time, the WiFi Infrastructure Network of the railway station Meraki Cisco System sends an update of the heatmap. A central server (Daemon-Server) assigns each robot a different subarea, applying the taboo-search method with 2-type Fuzzy logic. The chosen subareas are all different for each robot. The chosen subareas represent an attractive stimulus for the cockroach-like robot. The robot considers its distance from the attractive assigned subarea when it builds its cleaning path in the environment. At every instant of time, the robot chooses its next step, also considering other parameters like the presence of pheromones of other members of the team, the local concentration of priority that it finds where it will move, and the distance from the other robots. To select its next step, the robot processes all this information by applying the 2-type Fuzzy Logic technique. The choice of the 2-type Fuzzy Logic as a decision system is adequate when processes are too complex for analysis by conventional quantitative methods or when the available sources of information are interpreted in a qualitatively inaccurate or uncertain way [21], or the mathematical and statistical description of the time-variability is unknown in a variable time system [63], as in our case. To test this solution we used real data shared by Rete Ferroviaria Italiana regarding the positions of people inside Roma Termini, as done in the previous Chapters. The results are described in Section 6.2, and shown in Figure 6.4, where all the presented methods in this thesis are compared together with two other conventional CPP techniques recently proposed for sanitization tasks. The test highlighted the better performances of our new technique in terms of cleaning percentage. The combined use of the Daemon-Server as defined in [31], assigning different attractive zones to each robot thanks to the taboo search mechanism, together with the use of repulsive pheromones communications, we obtained a better spreading of the robots inside the heatmap, reducing path repetition and interference between the members of the team. Although, in this work, we focused only on the definition and testing of the high-level strategy, we wish to continue this research by integrating this method with lower-level navigation strategies to consider the avoidance of moving obstacles. We also plan to investigate heterogeneous teams as in [15], considering different roles for

the agents and hierarchies inside the insect colony.

Bibliography

[1] Olov Andersson, Oskar Ljungqvist, Mattias Tiger, Daniel Axehill, and Fredrik Heintz. Receding-horizon lattice-based motion planning with dynamic obstacle avoidance. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4467–4474, 2018.

[2] Olov Andersson, Mariusz Wzorek, Piotr Rudol, and Patrick Doherty. Model-predictive control with stochastic collision avoidance using bayesian policy optimization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4597–4604, 2016.

[3] Omer Atli and Cengiz Kahraman. Fuzzy resource-constrained project scheduling using taboo search algorithm. *International Journal of Intelligent Systems*, 27(10):873–907, 2012.

[4] Hyansu Bae, Gidong Kim, Jonguk Kim, Dianwei Qian, and Sukgyu Lee. Multi-robot path planning method using reinforcement learning. *Applied Sciences*, 9(15), 2019.

[5] Hritam Basak, Rohit Kundu, Pawan Kumar Singh, Muhammad Fazal Ijaz, Marcin Woźniak, and Ram Sarkar. A union of deep learning and swarm-based optimization for 3d human action recognition. *Scientific Reports*, 12(1):1–17, 2022.

[6] Luca Bertolini. *Station areas as nodes and places in urban networks: An analytical tool and alternative development strategies*, pages 35–57. Physica-Verlag HD, Heidelberg, 2008.

[7] Manuel Boldrer, Paolo Bevilacqua, Luigi Palopoli, and Daniele Fontanelli. Graph connectivity control of a mobile robot network with mixed dynamic multi-tasks. *IEEE Robotics and Automation Letters*, 6(2):1934–1941, 2021.

[8] C. Branca and R. Fierro. A hierarchical optimization algorithm for cooperative vehicle networks. In *2006 American Control Conference*, pages 6 pp.–, 2006.

[9] Bastian Broecker, Ipek Caliskanelli, Karl Tuyls, Elizabeth I Sklar, and Daniel Hennes. Hybrid insect-inspired multi-robot coverage in complex environments. In *Towards Autonomous Robotic Systems: 16th Annual Conference, TAROS 2015, Liverpool, UK, September 8-10, 2015, Proceedings 16*, pages 56–68. Springer, 2015.

[10] Bastian Broecker, Ipek Caliskanelli, Karl Tuyls, Elizabeth Ida Sklar, and Daniel Hennes. Social insect-inspired multi-robot coverage. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2015.

[11] G. Buonanno, L. Morawska, and L. Stabile. Quantitative assessment of the risk of airborne transmission of sars-cov-2 infection: Prospective and retrospective applications. *Environment International*, 145:106112, 2020.

[12] R Caccavale, V Calà, M Ermini, A Finzi, V Lippiello, and F Tavano. Multi-robot sanitization of railway stations based on deep q-learning. In *8th Italian Workshop on Artificial Intelligence and Robotics, AIRO 2021*, pages 34–39, 2022.

[13] R Caccavale, M Ermini, E Fedeli, A Finzi, E Garone, V Lippiello, and F Tavano. Combining hierarchical milp-mpc and artificial potential fields for multi-robot priority-based sanitization of railway stations. In *The 16th International Symposium on Distributed Autonomous Robotic Systems 2022*. Springer Proceedings in Advanced Robotics, 2023.

[14] R Caccavale, M Ermini, E Fedeli, A Finzi, V Lippiello, and F Tavano. A multi-robot deep q-learning framework for priority-based sanitization of railway stations. *Applied Intelligence*.

[15] Riccardo Caccavale, Mirko Ermini, Eugenio Fedeli, Alberto Finzi, Vincenzo Lippiello, and Fabrizio Tavano. Toward a heterogeneous multi-robot framework for priority-based sanitization of railway stations. In Agostino Dovier, Angelo Montanari, and Andrea Orlandini, editors, *AIxIA 2022 – Advances in Artificial Intelligence*, pages 387–401, Cham, 2023. Springer International Publishing.

[16] Ipek Caliskanelli, Bastian Broecker, and Karl Tuyls. Multi-robot coverage: A bee pheromone signalling approach. In *Artificial Life and Intelligent Agents: First International Symposium, ALIA 2014, Bangor, UK, November 5-6, 2014. Revised Selected Papers 1*, pages 124–140. Springer, 2015.

[17] Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11), 2021.

[18] Weert Canzler and Andreas Knie. Mobility in the age of digital modernity: why the private car is losing its significance, intermodal transport is winning and why digitalisation is the key. *Applied Mobilities*, 1(1):56–67, 2016.

[19] Zuo Llang Cao, Yuyu Huang, and Ernest L. Hall. Region filling operations with random obstacle avoidance for mobile robots. *J. Field Robotics*, 5:87–102, 1988.

[20] Andrea Carron and Melanie N Zeilinger. Model predictive coverage control. *IFAC-PapersOnLine*, 53(2):6107–6112, 2020.

[21] Oscar Castillo, Héctor Neyoy, José Soria, Patricia Melin, and Fevrier Valdez. A new approach for dynamic fuzzy logic parameter tuning in ant colony optimization and its application in fuzzy control of a mobile robot. *Applied soft computing*, 28:150–159, 2015.

[22] Maria Charitidou and Tamás Keviczky. An milp approach for persistent coverage tasks with multiple robots and performance guarantees. *European Journal of Control*, 64:100610, 2022.

[23] Zhang Chibin, Wang Xingsong, and Du Yong. Complete coverage path planning based on ant colony algorithm. In *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, pages 357–361. IEEE, 2008.

[24] Francesca Ciuffini, Simone Tengattini, and Alexander York Bigazzi. Mitigating increased driving after the covid-19 pandemic: An analysis on mode share, travel demand, and public transport capacity. *Transportation Research Record*, 0(0):03611981211037884, 0.

[25] Jorge Cortés and Magnus Egerstedt. Coordinated control of multi-robot systems: A survey. *SICE Journal of Control, Measurement, and System Integration*, 10(6):495–503, 2017.

[26] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on robotics and Automation*, 20(2):243–255, 2004.

[27] Kathryn A Daltorio, Brian R Tietz, John A Bender, Victoria A Webster, Nicholas S Szczecinski, Michael S Branicky, Roy E Ritzmann, and Roger D Quinn. A model of exploration and goal-searching in the cockroach, blaberus discoidalis. *Adaptive Behavior*, 21(5):404–420, 2013.

[28] Debora Di Caprio, Ali Ebrahimnejad, Hamidreza Alrezaamiri, and Francisco J Santos-Arteaga. A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights. *Alexandria Engineering Journal*, 61(5):3403–3415, 2022.

[29] F John Dian. An indoor environmental visualization system (ievs). In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0098–0102. IEEE, 2019.

[30] Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Towards decentralized coordination of multi robot systems in industrial environments: A hierarchical traffic control strategy. In *2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 209–215, 2013.

[31] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[32] Jiajun Duan, Di Shi, Ruisheng Diao, Haifeng Li, Zhiwei Wang, Bei Zhang, Desong Bian, and Zhehan Yi. Deep-reinforcement-learning-based autonomous voltage control for power grid operations. *IEEE Transactions on Power Systems*, 35(1):814–817, 2020.

[33] Matthew G. Earl and Raffaello D'Andrea. A decomposition approach to multi-vehicle cooperative control. *Robotics and Autonomous Systems*, 55(4):276–291, 2007.

[34] Marco Faroni, Alessandro Umbrico, Manuel Beschi, Andrea Orlandini, Amedeo Cesta, and Nicola Pedrocchi. Optimal task and motion planning and execution for multiagent systems in dynamic environments. *IEEE Transactions on Cybernetics*, pages 1–12, 2023.

[35] Alberto Finzi and Andrea Orlandini. Human-robot interaction through mixed-initiative planning for rescue and search rovers. In *Congress of the Italian Association for Artificial Intelligence*, pages 483–494. Springer, 2005.

[36] Giovanni Fusco and James M Coughlan. Indoor localization for visually impaired travelers using computer vision on a smartphone. In *Proceedings of the 17th International Web for All Conference*, pages 1–11, 2020.

[37] Carlos E Galván-Tejada, Laura A Zanella-Calzada, Antonio García-Domínguez, Rafael Magallanes-Quintanar, Huizilopoztli Luna-García, Jose M Celaya-Padilla, Jorge I Galván-Tejada, Alberto Vélez-Rodríguez, and Hamurabi Gamboa-Rosales. Estimation of indoor location through

magnetic field data: An approach based on convolutional neural networks. *ISPRS International Journal of Geo-Information*, 9(4):226, 2020.

[38] Claudio Gariazzo, Armando Pelliccioni, and Maria Paola Bogliolo. Spatiotemporal analysis of urban mobility using aggregate mobile phone derived presence and demographic data: A case study in the city of rome, italy. *Data*, 4(1):8, Jan 2019.

[39] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2021.

[40] Mose Gu, Jiyong Uhm, and Jaehoon Paul Jeong. A particle filter-based indoor positioning system using an rssi heatmap. , pages 1094–1095, 2022.

[41] Amir Arslan Haghrah and Sehraneh Ghaemi. Pyit2fls: A new python toolkit for interval type 2 fuzzy logic systems, 2019.

[42] Steven Hanna. Transport and dispersion of tracers simulating covid-19 aerosols in passenger aircraft. *Indoor Air*, 32(1):e12974.

[43] Marcel Harmon and Josephine Lau. The facility infection risk estimator™: A web application tool for comparing indoor risk mitigation strategies by estimating airborne transmission risk. *Indoor and Built Environment*, 0(0):1420326X211039544, 0.

[44] Andrew Howard, Lynne E. Parker, and Gaurav S. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *The International Journal of Robotics Research*, 25(5-6):431–447, 2006.

[45] Chenn-Jung Huang, Yu-Wu Wang, Heng-Ming Chen, Han-Wen Tsai, Jui-Jiun Jian, Ai-Lin Cheng, and Jia-Jian Liao. Application of cellular automata and type-2 fuzzy logic to dynamic vehicle path planning. *Applied Soft Computing*, 19:333–342, 2014.

[46] Adomas Ivanovas, Armantas Ostreika, Rytis Maskeliūnas, Robertas Damaševičius, Dawid Połap, and Marcin Woźniak. Block matching based obstacle avoidance for unmanned aerial vehicle. In *International Conference on Artificial Intelligence and Soft Computing*, pages 58–69. Springer, 2018.

[47] Lan Jiang, Hongyun Huang, and Zuohua Ding. Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge. *IEEE/CAA Journal of Automatica Sinica*, 7(4):1179–1189, 2020.

[48] Samane Kaviri, Ahmadreza Tahsiri, and Hamid D Taghirad. Coverage control of multi-robot system for dynamic cleaning of oil spills. In *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, pages 17–22. IEEE, 2019.

[49] Samane Kaviri, Ahmadreza Tahsiri, and Hamid D Taghirad. A cooperative control framework of multiple unmanned aerial vehicles for dynamic oil spill cleanup. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 43(6):289, 2021.

[50] Anirudh Krishna Lakshmanan, Rajesh Elara Mohan, Balakrishnan Ramalingam, Anh Vu Le, Prabahar Veerajagadeshwar, Kamlesh Tiwari, and Muhammad Ilyas. Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot. *Automation in Construction*, 112:103078, 2020.

[51] Zuohua Ding Lan Jiang, Hongyun Huang. Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge, 2020.

[52] Tae-Kyeong Lee, Sang-Hoon Baek, Young-Ho Choi, and Se-Young Oh. Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robotics and Autonomous Systems*, 59(10):801–812, 2011.

[53] Tae-Kyeong Lee, Sang-Hoon Baek, Se-Young Oh, and Young-Ho Choi. Complete coverage algorithm based on linked smooth spiral paths for mobile robots. In *2010 11th International Conference on Control Automation Robotics Vision*, pages 609–614, 2010.

[54] Tae-Kyeong Lee, Sanghoon Baek, and Se-Young Oh. Sector-based maximal online coverage of unknown environments for cleaning robots with limited sensing. *Robotics and Autonomous Systems*, 59(10):698–710, 2011.

[55] Xiting Liu, Banghui Lu, Jianwei Niu, Lei Shu, and Yuanfang Chen. Hmf: heatmap and wifi fingerprint-based indoor localization with building layout consideration. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 324–331. IEEE, 2016.

[56] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[57] Wenhao Luo and Katia Sycara. Voronoi-based coverage control with connectivity maintenance for robotic sensor networks. In *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 148–154. IEEE, 2019.

[58] Deepak Mallya, Sumanth Kandala, Leena Vachhani, and Arpita Sinha. Priority patrolling using multiple agents. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8692–8698, 2021.

[59] Ugnius Malūkas, Rytis Maskeliūnas, Robertas Damaševičius, and Marcin Woźniak. Real time path finding for assisted living using deep learning. *J. Univers. Comput. Sci*, 24(4):475–487, 2018.

[60] Varghese Mathai, Asimanshu Das, Jeffrey A Bailey, and Kenneth Breuer. Airflows inside passenger cars and implications for airborne disease transmission. *Science advances*, 7(1):eabe0166, 2021.

[61] Xu Miao, Hyun-Soon Lee, and Bo-Yeong Kang. Multi-cleaning robots using cleaning distribution method based on map decomposition in large environments. *IEEE Access*, 8:97873–97889, 2020.

[62] Xu Miao, Jaesung Lee, and Bo-Yeong Kang. Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments. *IEEE Access*, 6:38200–38215, 2018.

[63] Kanika Mittal, Amita Jain, Kunwar Singh Vaisla, Oscar Castillo, and Janusz Kacprzyk. A comprehensive review on type 2 fuzzy logic applications: Past, present and future. *Engineering Applications of Artificial Intelligence*, 95:103916, 2020.

[64] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, and et al. Human-level control through deep reinforcement learning, Feb 2015.

[65] Ghulam Murtaza, Salil Kanhere, and Sanjay Jha. Priority-based coverage path planning for aerial wireless sensor networks. In *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 219–224, 2013.

[66] Mohet Narang, Mehul Rana, Jai Patel, Steve D'souza, Princewill Onyechie, Cody Berry, Mohsen Tayefeh, and Ahmad Barari. Fighting covid: An autonomous indoor cleaning robot (aicr) supported by artificial intelligence and vision for dynamic air disinfection. In *2021 14th IEEE International Conference on Industry Applications (INDUSCON)*, pages 1146–1153, 2021.

[67] B. Nasirian, M. Mehrandezh, and F. Janabi-Sharifi. Efficient coverage path planning for mobile disinfecting robots using graph-based representation of environment. *Frontiers in Robotics and AI*, 8, 2021.

[68] Behnam Nasirian, Mehran Mehrandezh, and Farrokh Janabi-Sharifi. Efficient coverage path planning for mobile disinfecting robots using graph-based representation of environment. *Frontiers in Robotics and AI*, 8:4, 2021.

[69] Ibidun C Obagbuwa, Aderemi O Adewumi, and Ayodele A Adebiyi. A dynamic step-size adaptation roach infestation optimization. In *2014 IEEE International Advance Computing Conference (IACC)*, pages 1201–1205. IEEE, 2014.

[70] Ibidun Christiana Obagbuwa and Ademola Philips Abidoye. Binary cockroach swarm optimization for combinatorial optimization problem. *Algorithms*, 9(3), 2016.

[71] Joon Seop Oh, Yoon Ho Choi, Jin Bae Park, and Y.F. Zheng. Complete coverage navigation of cleaning robots using triangular-cell-based map. *IEEE Transactions on Industrial Electronics*, 51(3):718–726, 2004.

[72] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2681–2690, 2017.

[73] Lynne E. Parker. Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance, 2000.

[74] Fabio Pasqualetti, Joseph W. Durham, and Francesco Bullo. Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. *IEEE Transactions on Robotics*, 28(5):1181–1188, 2012.

[75] Gert-Joost Peek and Erik Louw. A multidisciplinary approach of railway station development: A case study of 's-Hertogenbosch. In Frank Bruinsma, Eric Pels, Piet Rietveld, Hugo Priemus, and Bert Wee, editors, *Railway Development*, Springer Books, chapter 7, pages 125–143. December 2008.

[76] Janusz Poliński and Krzysztof Ochociński. Impact of covid-19 on the functioning of passenger rail transport. *Problemy Kolejnictwa*, Z. 190:103–124, 2021.

[77] KJ Poornaselvan, T Gireesh Kumar, and Vinodh P Vijayan. Agent based ground flight control using type-2 fuzzy logic and hybrid ant colony optimization to a dynamic environment. In *2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 343–348. IEEE, 2008.

[78] Federico Pratissoli, Beatrice Capelli, and Lorenzo Sabattini. On coverage control for limited range multi-robot systems. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9957–9963. IEEE, 2022.

[79] Wei Qin, Zilong Zhuang, Zizhao Huang, and Haozhe Huang. A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. *Computers Industrial Engineering*, 156:107252, 2021.

[80] Bijan Ranjbar-Sahraei, Gerhard Weiss, and Ali Nakisaee. A multi-robot coverage approach based on stigmergic communication. In *Multiagent System Technologies: 10th German Conference, MATES 2012, Trier, Germany, October 10-12, 2012. Proceedings 10*, pages 126–138. Springer, 2012.

[81] Yadollah Rasekhipour, Amir Khajepour, Shih-Ken Chen, and Bakhtiar Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1255–1267, 2017.

[82] Yongli Ren, Flora Dilys Salim, Martin Tomko, Yuntian Brian Bai, Jeffrey Chan, Kyle Kai Qin, and Mark Sanderson. D-log: A wifi log-based differential scheme for enhanced indoor localization with single RSSI source and infrequent sampling rate. *Pervasive Mob. Comput.*, 37:94–114, 2017.

[83] KSA Riyadh. Anovel prototype model for swarm mobile robot navigation based fuzzy logic controller. *International Journal of Computer Science Information Technology (IJCSIT)*, 11(2), April 2019.

[84] Guillaume Sartoretti, Yue Wu, William Paivine, T. K. Satish Kumar, Sven Koenig, and Howie Choset. Distributed reinforcement learning for multi-robot decentralized collective construction. In Nikolaus Correll, Mac Schwager, and Michael Otte, editors, *Distributed Autonomous Robotic Systems*, pages 35–49, Cham, 2019. Springer International Publishing.

[85] Leonardo Setti, Fabrizio Passarini, Gianluigi De Gennaro, Pierluigi Barbieri, Maria Grazia Perrone, Massimo Borelli, Jolanda Palmisani, Alessia Di Gilio, Prisco Piscitelli, Alessandro Miani, and et al. Airborne transmission route of covid-19: Why 2 meters/6 feet of inter-personal distance could not be enough, Apr 2020.

[86] Luca Siligardi, Jacopo Panerati, Marcel Kaufmann, Marco Minelli, Cinara Ghedini, Giovanni Beltrame, and Lorenzo Sabattini. Robust area coverage with connectivity maintenance. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2202–2208, 2019.

[87] Peng Song and V. Kumar. A potential field based approach to multi-robot manipulation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 2, pages 1217–1222 vol.2, 2002.

[88] Xiaoyu Song, Yunlong Zhu, Chaowan Yin, and Fuming Li. A hybrid strategy based on ant colony and taboo search algorithms for fuzzy job shop scheduling. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 7362–7365. IEEE, 2006.

[89] Paolo Stegagno, Marco Cognetti, Lorenzo Rosa, Pietro Peliti, and Giuseppe Oriolo. Relative localization and identification in a heterogeneous multi-robot system. In *2013 IEEE International Conference on Robotics and Automation*, pages 1857–1864, 2013.

[90] Ethan Stump and Nathan Michael. Multi-robot persistent surveillance planning as a vehicle routing problem. In *2011 IEEE International Conference on Automation Science and Engineering*, pages 569–575, 2011.

[91] Herbert G. Tanner. Switched uav-ugv cooperation scheme for target detection. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3457–3462, 2007.

[92] Alessio Tardivo, Armando Carrillo Zanuy, and Celestino Sánchez Martín. Covid-19 impact on transport: A paper from the railways' systems research perspective. *Transportation Research Record*, 2675(5):367–378, 2021.

[93] Mahdi Tavakoli, Jay Carriere, and Ali Torabi. Robotics, smart wearable technologies, and autonomous intelligent systems for healthcare during the covid-19 pandemic: An analysis of the state of the art and future vision. *Advanced Intelligent Systems*, 2(7):2000071.

[94] Fabrizio Tavano, Riccardo Caccavale, Mirko Ermini, Eugenio Fedeli, Luca Ricciardi, Alberto Finzi, and Vincenzo Lippiello. Bioinspired artificial cockroach colony strategy combined with 2-type fuzzy logic for the priority-based sanitization of railway stations. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 359–374. Springer, 2023.

[95] Amee Trivedi, Kate Silverstein, Emma Strubell, Prashant Shenoy, and Mohit Iyyer. Wifimod: Transformer-based indoor human mobility modeling using passive sensing. In *ACM SIGCAS Conference on Computing and Sustainable Societies*, COMPASS '21, page 126–137, New York, NY, USA, 2021. Association for Computing Machinery.

[96] R. Vidal, O. Shakernia, H.J. Kim, D.H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002.

[97] Anh Vu Le, Balakrishnan Ramalingam, Braulio Félix Gómez, Rajesh Elara Mohan, Tran Hoang Quang Minh, and Vinu Sivanantham. Social density monitoring toward selective cleaning by human support robot with 3d based perception system. *IEEE Access*, 9:41407–41416, 2021.

[98] Di Wang, Hongbin Deng, and Zhenhua Pan. Mrcdrl: Multi-robot coordination with deep reinforcement learning. *Neurocomputing*, 406:68–76, 2020.

[99] Steven L. Waslander. *Unmanned Aerial and Ground Vehicle Teams: Recent Work and Open Problems*, pages 21–36. Springer Japan, Tokyo, 2013.

[100] Marcin Woźniak, Adam Zielonka, and Andrzej Sikora. Driving support by type-2 fuzzy logic control model. *Expert Systems with Applications*, 207:117798, 2022.

[101] Yonghao Yin, Dewei Li, Songliang Zhang, and Lifu Wu. How does railway respond to covid-19 spreading? – countermeasure analysis and evaluation around the world, 2020.

[102] Choujun Zhan, Chi K. Tse, Yuxia Fu, Zhikang Lai, and Haijun Zhang. Modeling and prediction of the 2019 coronavirus disease spreading in china incorporating human migration data. *PLOS ONE*, 15(10):1–17, 10 2020.

[103] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5872–5881, 10–15 Jul 2018.

[104] Bin Zhou, Xuemei Duan, Dongjun Ye, Wei Wei, Marcin Woźniak, Dawid Połap, and Robertas Damaševičius. Multi-level features extraction for discontinuous target tracking in remote sensing image monitoring. *Sensors*, 19(22):4855, 2019.

[105] K Zhou, A Leck Jensen, CG Sørensen, Patrizia Busato, and DD Bothtis. Agricultural operations planning in fields with multiple obstacle areas. *Computers and electronics in agriculture*, 109:12–22, 2014.

[106] Xing Zhou, Huaimin Wang, Bo Ding, Tianjiang Hu, and Suning Shang. Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. *Expert Systems with Applications*, 116:10–20, 2019.

[107] Ángel Madridano, Abdulla Al-Kaff, David Martín, and Arturo de la Escalera. Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications*, 173:114660, 2021.

Author's Publications

The list of the authors for the following articles is positioned in alphabetical order:

1. R Caccavale, V Calà, M Ermini, A Finzi, V Lippiello, and F Tavano. Multi-robot sanitization of railway stations based on deep q-learning. In 8th Italian Workshop on Artificial Intelligence and Robotics, AIRO 2021, pages 34–39, 2022.
2. R Caccavale, M Ermini, E Fedeli, A Finzi, V Lippiello, and F Tavano. A multi-robot deep q-learning framework for priority-based sanitization of railway stations. Applied Intelligence (2023).
3. R Caccavale, M Ermini, E Fedeli, A Finzi, E Garone, V Lippiello, and F Tavano. Combining Hierarchical Milp-Mpc and Artificial Potential Fields for Multi-Robot priority-based Sanitization of Railway Stations. In Fumitoshi Matsuno, Shun-ichi Azuma, and Masahito Yamamoto, editors, Distributed Autonomous Robotic Systems. Springer International Publishing, 2023.
4. Riccardo Caccavale, Mirko Ermini, Eugenio Fedeli, Alberto Finzi, Vincenzo Lippiello, and Fabrizio Tavano. Toward a heterogeneous multi-robot framework for priority-based sanitization of railway stations. In Agostino Dovier, Angelo Montanari, and Andrea Orlandini, editors, AIxIA 2022 – Advances in Artificial Intelligence, pages 387–401, Cham, 2023. Springer International Publishing

In the following article, I figure as the first author:

5. F Tavano, R Caccavale, M Ermini, E Fedeli, A Finzi, E Garone, V Lippiello, and L Ricciardi. Bioinspired artificial cockroach colony strategy combined with 2-type fuzzy logic for the priority-based sanitization of railway stations. In Advances in Practical Applications of Agents, Multi-Agent

Systems, and Cognitive Mimetics. The PAAMS Collection, volume 13955. Lecture Notes in Computer Science series, 2023

In the following list, there are other articles in which I participated during my Ph.D. studies:

6. Mirko Ermini, Giuseppe Cadavero, Fabrizio Tavano, Sebastiano Trigila, Luca Rea, Samuela Persia, Francesco G. Lavacca, Francesco D'Alterio; An innovative approach in interfacing Diagnostic Application protocols for the RFI Network with legacy communication stacks; 13th World Congress on Railway Research (WCRR), International Convention Centre Birmingham, United Kingdom, 6- 10 JUNE 2022;

7. Fabrizio Cerreto, Paola Pellegrini, Nathalie Botticchio, Rémy Chevrier , Fabrizio Tavano; Assessing self-organization algorithms for railway traffic: the selection of three case studies for the SORTEDMOBILITY research project; RailBelgrade 2023, 10th International Conference on Railway Operations Modelling and Analysis (ICROMA), Belgrade, Serbia, 25th – 28th 2023

8. Eugenio Fedeli, Franco Iacobini, Innocenzo Mungiello, Giuseppe Cadavero, Dario D'Avino, Fabrizio Tavano, Ivan Agostino and Stefano Meuti; Supervision of railway areas by satellite images, IRSC 2022 Congress, the International Railway Safety Council, October 16-21, 2022, Seville, Spain National journal papers

9. Paolo Cesario, Ivan Colla, Matteo Sciutto, Filippo Sugliano,David Gomez Casco, Fabrizio Tavano, Tiziano Cosso, Michal Falta, SEMOR: Safety Enhancement of Maintenance Operators in Railway worksite, Ingegneria Ferroviaria, Collegio Ingegneri Ferroviari Italiani (CIFI), accepted in date: December 2022