Balanced Task Sharing Method for Frequency-Based Multi-Agent Patrolling Teams

July 2018

Vourchteang SEA

Balanced Task Sharing Method for Frequency-Based Multi-Agent Patrolling Teams

July 2018

Waseda University
Graduate School of Fundamental Science and Engineering
Department of Computer Science and Communications Engineering,
Research on Intelligent Software

Vourchteang SEA

Abstract

Ongoing advances on autonomous mobile robots have been evident in the last couple of decades. The patrolling problem with a team of cooperative agents, in particular, has received much focus. This has made multi-agent (multi-robot) patrolling problem receive growing attention from many researchers over the past few years due to its wide range of potential applications. In realistic environment, e.g., security patrolling, each location has different visitation requirement according to the required security level. The difference in visiting frequency generally causes imbalanced workload among agents, leading to inefficiency. Therefore, a patrolling system that can take into account the non-uniform visiting frequency is needed in real-world applications.

Multi-agent patrolling, however, is not limited to patrolling real-world environments, yet they can be found in applications on several domains, such as continuous sweeping, security patrolling, surveillance systems, network security systems and games. This has motivated many researchers to focus their studies on multi-agent patrolling problem by proposing and implementing plenty of methods from simple to sophisticated ones. Unfortunately, most of the studies do not consider when each location to be patrolled has different visitation requirement, and many of them are not able to balance the workload among all patroller agents.

In this study, we first described the area partitioning method that reflects the differences in robot specifications and the characteristics of partitioned areas. Then, by generalizing this method, we proposed a frequency-based area partitioning method for balanced workload in multi-agent patrolling team. Our proposed work aims at partitioning a given area so as to balance agents' workload by taking into account the

different visiting frequencies and then generating route inside each allocated sub-area. We formulate the problem of frequency-based multi-agent patrolling and propose its semi-optimal solution method, whose overall process consists of two steps – graph partitioning and sub-graph patrolling. Our work improve the traditional k-means clustering algorithm by formulating a new objective function for graph partitioning and combining it with simulated annealing, which is one of the useful tools for operations research, for sub-graph patrolling. Experimental results illustrated the effectiveness and reasonable computational efficiency of our approach.

Acknowledgements

It is a great pleasure to mention all the people who have encouraged, motivated and supported me during my PhD study. I would like to take this opportunity to thank the following people who stay behind this achievement.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Toshiharu Sugawara, for his significant and continuous support, constructive advices and precious guidance throughout my degree which enabled me to complete this research work successfully. He has always been very helpful and cooperative. I am really grateful for his patience, motivation, kindness and understanding, and especially for his time spent on discussing and accurately reviewing all the research work of this thesis. This research would not be successfully achieved without his guidance and support.

With much respect, I also would like to express my sincere thanks to my examiners, Prof. Tatsuo Nakajima and Prof. Kazunori Ueda, for their insightful discussions and invaluable comments and feedback which help improve the quality of this thesis.

My appreciation also goes to all my laboratory members, especially Mr. Ayumi Sugiyama, who has always provided invaluable discussions, comments and support not only for my research activities but also for my living in Japan. I feel so lucky to have such a good-environment laboratory as well as nice and generous lab mates.

My sincere thanks also go to all the staff in the department of computer science and engineering for their kindly assistance and friendly behavior towards the students.

I also would like to acknowledge the financial support from the Japanese government for providing me the MEXT Scholarship. Without

their financial support, I would not be able to fulfill my dream in pursuing my study in Japan.

From the bottom of my heart, I would like to thank my boyfriend for everything he has done so far, especially for always trying his best for me. Besides my parents, he is the only person who is always by my side through ups and downs, whom I can share both my sorrow and happiness, and who always tries to cheer me up whenever I feel down. Words will never be enough to say how much his unconditional love and care despite the distance really mean to me. I am so blessed to have him as the biggest gift in my life.

Last but not least, I would like to gratefully and deeply thank my parents for their endless love, warm care and continuous support throughout my PhD study. They are the strongest motivation for me to overcome all the difficulties I had faced during these three academic years. I dedicate this thesis to my parents.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

The evolution of computer technologies, nowadays, has been dramatically increasing day by day and has been used in many application domains including the field of robotics and automation. Since robots can often perform tasks well in a variety of environments, the demand for robotic applications has been increasing which makes the popularity of robotic automation growing across a wide range of sectors. This has led to the massive of software applications in the field of robotics combining computer and sensor technologies.

Robotics are an extension of machinery that has some forms of information processing linked to the powerful computers or controllers. This means that robots can be made generic and programmed to do different tasks, or respond to changes in the environment to better complete their tasks.

The advantages of robotics have become more noticeable as industrial robotics technology has been widespread over 50 years since the first industrial robot, namely Unimate (as shown in Fig. 1.3), was put into use in 1961. About 90% of the robots today are installed and operating in the industrial robotics sector in factories [33]. According to the report from the Robotics Industry Association (RIA), the number of industrial robots, approximately 140,000, were in use in the U.S. in 2004. Furthermore, the International Federation of Robotics (IFR) also reported

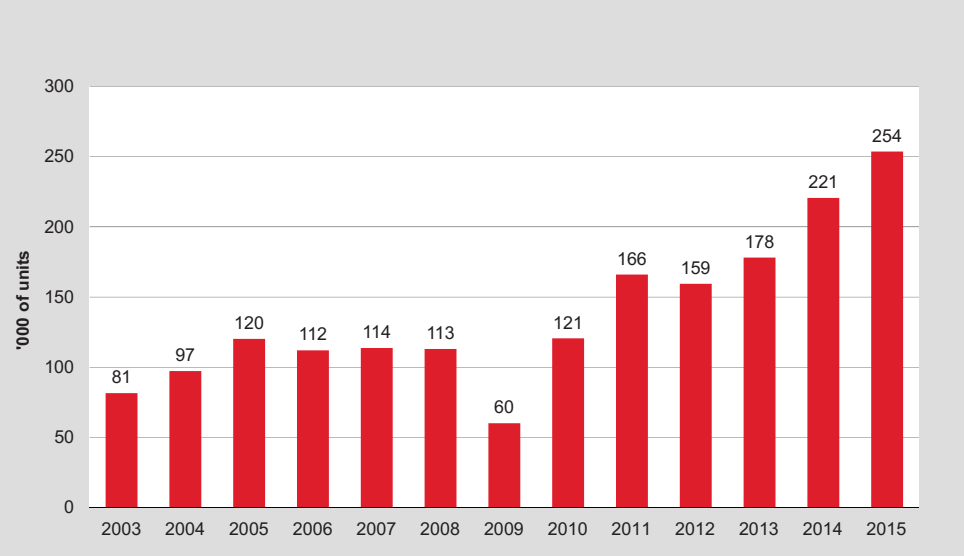

FIGURE 1.1: Estimated worldwide annual supply of industrial robots (extracted from [38])

that the annual supply of industrial robots had risen over time between 2003 and 2015 based on the increasing role of robots in improving the production lines and other business activities, as shown in Fig. 1.1. However, the economic and financial crisis during 2008-2009 lowered the amount of robot supplied around the world to about 60 thousand units before the sales began to sharply increase to 254 thousand units in 2015. Robots are now also used in laboratories, exploration sites, research and development facilities, energy plants, hospitals, warehouses and outer space. Similarly, Fig. 1.2 indicates the estimated supply of industrial robots in different industries during the period of 2014-2016 where automotive industry and electronics share major proportion of the total supply. Some robots can work in places and situations that are difficult for human, such as nuclear plant search, interplanetary exploration and disaster relief, while others can fulfill the daily functions such as cleaning and security patrolling robots.

Robots has been becoming more user friendly, intelligent and most importantly affordable due to the recent development in the field of robotics [70]. It is obvious that this advancement is the major reason for the existence of robots in various industries ranging from industrial

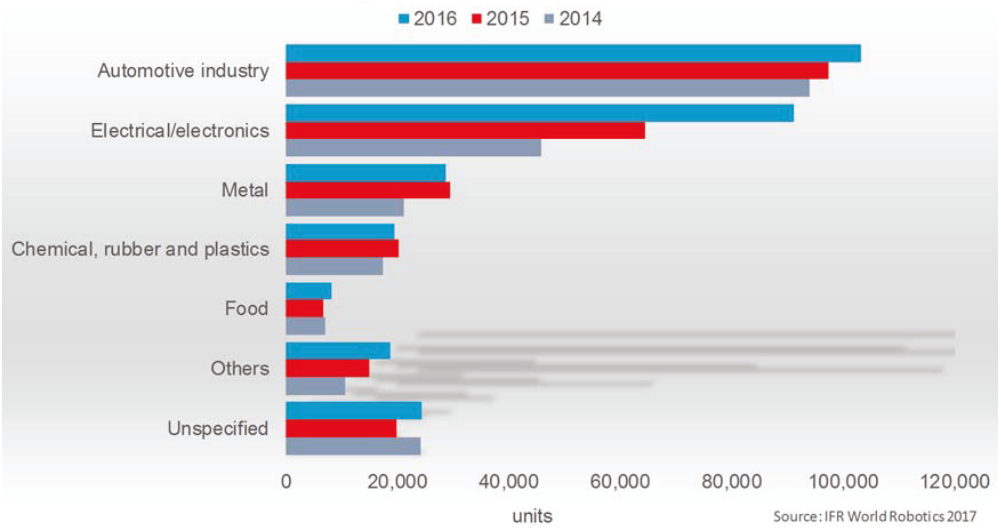

FIGURE 1.2: Estimated annual supply of industrial robots at year-end by industries worldwide 2014-2016 (extracted from [40])

(manufacturing industry) to medical field (health care service). Robots can handle heavy and high-risk tasks which ensures the safe working environment. Furthermore, they have been intentionally used for improving the productivity, and saving time and money.

Robots are used in the medical field for complex surgeries as those surgeries cannot be done by human, i.e., prostate cancer surgery. In particular, they can precisely reach and fit where human ' s hands cannot, allowing greater accuracy, flexibility and control. Moreover, other benefits of robotic are less invasive procedures resulting in less post-operative pain and risk of infection for the patients [70]. In addition, robots are now being used in the chemical industry and can, for example, deal with chemical spills in a nuclear plant, which would otherwise pose a major health concern.

The benefits of using robotics can be categorized into four main classifications [30, 33]:

- *Quality Assurance Improvement.* This focuses on quality, accuracy, or precision. As the nature of human beings, workers are less likely to enjoy doing tasks repetitively. Therefore, their concentration levels tend to decline over time. This leads to costly errors in business and sometimes can cause serious injury to the staff

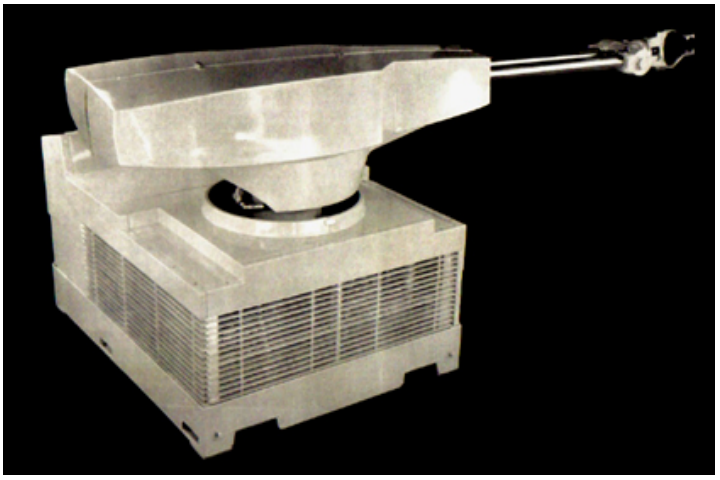

FIGURE 1.3: The first industrial robot, Unimate (extracted from [59])

members. Robot automation gets rid of these risks by accurately and successfully producing products with standardized quality. Having more products with high standard to be manufactured allows the enterprises to broaden various business possibilities. For example, industrial robots appear mostly as the form of a robotic arm working in the major production line of manufacturing industries. This kind of robotic arms are identical to the robots shown in Fig. 1.3 and 1.4, where the first figure shows the first invented industrial robot, namely Unimate; the latter figure demonstrates the modern industrial robotic arm. According to the aforementioned role of robots in maintaining the quality of products, a robotic arm can easily handle repetitive tasks with great precision leading to the improvement and consistency of product quality. This fact is also applicable to a number of production activities including welding and assembling process.

- *Cost Effectiveness.* The efficiency and speed improvement of industrial robots has been the result of the mechanical nature of the equipment and the computerized control which lead to a higher productivity than human labor. Robots are able to work non-stop on a repetitive-cycle tasks unless it is programmed to stop. There will be no lunchbreaks, holidays, sick leave or shift time assigned to robotic automation. This eliminates the risk of repetitive strain

injury (RSI)[1]. This productivity increment at a lower cost is obviously beneficial for the manufacturers.

- *Ability to Work in Hazardous Environments.* This is the most appealing benefit of robotics utilization. Robots have the ability to perform a number of tasks in a place where it is unsafe (too dangerous), unstable, too exposed to toxins, or inhospitable for humans. For instance, the spray painting tasks affect negatively to people inhaling the paint fume, but not to the robots. This also includes such daunting tasks as defusing bombs and such dirty tasks as cleaning sewers. Moreover, robots remain active by continuously performing tasks without getting any harm even in a situation where a high level of chemicals exist. Therefore, robotic automation can be used in every place where human safety is a huge concern.

- *Freedom from Human Limitations Like Boredom.* The absence of boredom postulates the greater precision and quality of production. Furthermore, a robotic arm can perform the task non-stop or with occasional downtime due to scheduled maintenance. Robots also do not eat or get sick like human does which is obviously an absolute advantage of using robotics.

Aside from the advances in the field of robotics and automation, the multiple robots' context has gained more popularity comparing to the single-robot context. In recent years, the use of Multi-Robot Systems (MRS) has become apparent for several application domains such as exploration, surveillance, and even search and rescue. The main reason for using these MRS is that they provide convenient solution in term of cost, performance, reliability, efficiency, and human exposure reduction [15].

The use of MRS is generally believed by researchers to hold several advantages over single-robot systems [8, 14, 24]. The most common motivations for developing multi-robot system solutions in the real-world applications are that [35, 63]:

[1]RSI is related to the pain felt in the upper part of the body such as wrist, forearm, elbow, shoulder, back or neck caused by repetitive movement and overuse of muscles and tendons. Certain activities that increase the risk of RSI are lifting heavy objects, doing the same activity over a long period of time without rest, working in an awkward position, etc. [72].

FIGURE 1.4: Industrial robotic arm (extracted from [89])

1. A single robot cannot adequately deal with task complexities;
2. The task is inherently distributed composed of sub-systems, which are physically and geographically separated;
3. Building several resource-limited robots is much easier than having a single powerful robot;
4. Multiple robots cause enhanced productivity as they can solve problems and complete some tasks faster using sub-tasks parallelism or because of the spatial distribution of the individual robot;
5. The initialization of multiple robots increases robustness and reliability of the whole system through redundancy as multi-robot teams provide redundancy so that the failure of a single robot does not completely stop the whole task from being performed.

Motivated by the above significant advantages, multi-robot system has become an active research field in robotics for many years. This has made many researchers focus their studies and researches on different issues using multiple robots such as path planning or graph exploration, communication, negotiation, area partitioning, cooperation/coordination, map building, autonomous navigation, self-localization, and obstacle or collision avoidance.

The generalization of the multi-robot systems is a multi-agent systems (MAS), which is a computerized system composed of multiple interacting intelligent agents within a given/unknown environment. MAS consists of entities (i.e., computer programs, robots, or humans) that are each specialized for a certain task. They cooperate to achieve the ultimate goal, yet individually they are also able to do some tasks. MAS tend to find the best solution for their problems without any intervention. The systems also tend to prevent propagation of faults, self-recover and be fault tolerant, mainly due to the redundancy of components.

The study of MAS is generally concerned with the advancement and analysis of sophisticated artificial intelligence (AI) problem-solving and control architectures for both single-agent and multi-agent systems.

1.2 Motivation

Although a lot of advancement has been done in the MAS, yet there are still many challenging issues that are remaining and need to be solved. These issues include coordination, cooperation control, path planning, collision avoidance, communication among robots, navigation, area

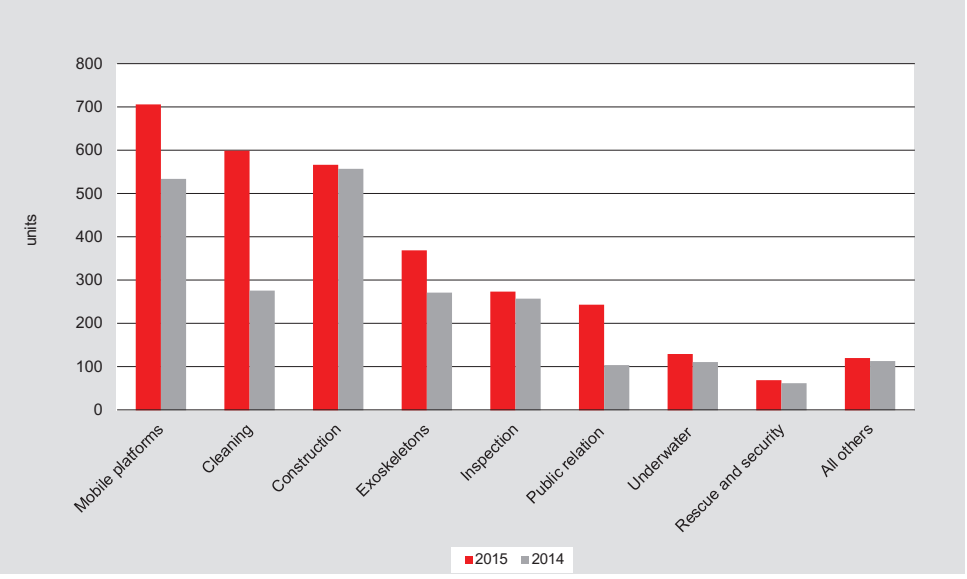

FIGURE 1.5: Service robots for professional use. Sold units 2015 and 2014 (continued) [39]

partitioning, task allocation, etc. Among all these issues, the cooperation/coordination of multiple agents is an important issue in the field of robotics.

Continuing advances in computer science and robotics have led to applications for covering large areas that require coordinated tasks by multiple control programs including robots. One of the hottest applications that requires coordination and cooperation between a group of agents is the cleaning/sweeping robots. If we look inside that, we can see the cooperation of multiple cleaning robots has now become very crucial, since effective cooperative cleaning of multiple robots can improve the working quality and reduce the time for cleaning by sharing tasks. To be more emphasized, a report from IFR in 2016 about service robots also indicates a more than double increase in the demand for cleaning robots in 2015 comparing to last year statistic as shown in Fig. 1.5. Moreover, this report also claims that the sales of robots for household usage is predicted to be extremely high between 2016 to 2019 which is demonstrated in Fig. 1.6. This upsurge may be resulted from the realization of the benefit of service robots in household's daily activities.

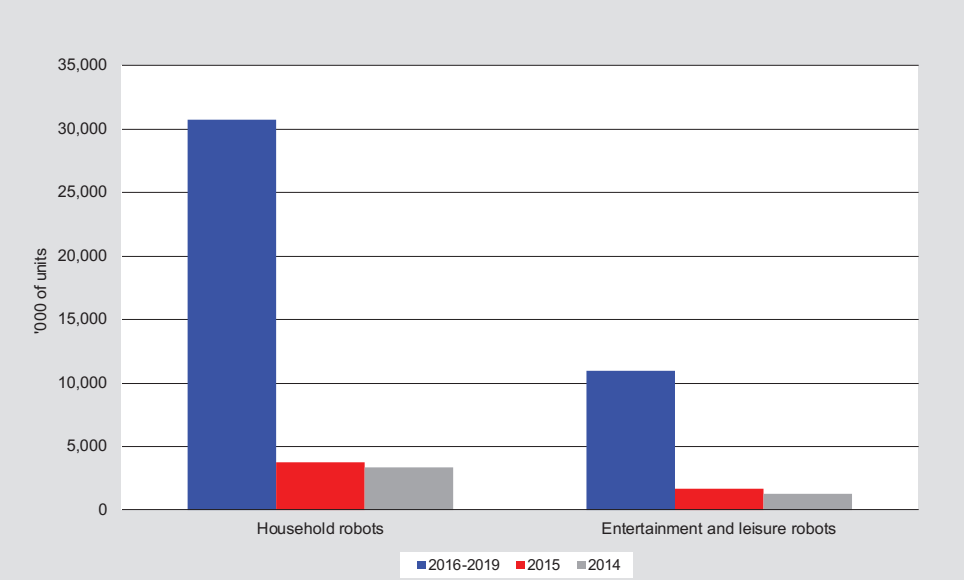

FIGURE 1.6: Service robots for personal/domestic use. Units sales forecast 2016-2019, 2015 and 2014. (extracted from [39])

In addition, the control for cleaning is also applicable to other sweeping tasks such as security patrolling and area search.

Besides coordination and cooperation, an area partitioning context has also been apparent in many research work [2, 5, 26, 41, 44, 67], and thus a coordinated area partitioning method for cooperative sweeping robots is needed for continuous cooperative tasks. Therefore, this has motivated us to firstly deal with the methods for cooperation/coordination of multiple agents, which are control programs of robots, using examples of cleaning tasks by multiple robots.

However, as the study of cleaning robots has some limitations due to its specific type of application, we have changed it to a more general context which can be applied to more applications. In this step, we have focused our study on a multi-agent patrolling problem (MAPP), which is a more generalized problem of the multi-agent cleaning/sweeping task. Although the advances on autonomous mobile robots have been evident in the last few decades, the patrolling problem with a group of agents, in particular, has received much attention. This is due to the fact that multi-agent (multi-robot) patrolling is not limited to patrolling real-world environments, yet they can be found in applications on several domains, such as continuous sweeping, security patrolling, surveillance systems, network security systems and games. In other words, patrolling can be applicable and beneficial in any domain characterized by the need of systematically visiting a set of given points [73]. These all above are our motivations for conducting the research experiments in this thesis.

1.3 Problem Statement

Since we mainly focus our study on multi-agent patrolling problem, the problem in this study is *"how to partition a given area based on the visitation requirement of each location in a balanced manner by multiple agents in a distributed and autonomous manner by taking into account the differences in the given areas, hardware specification, exploration algorithms and so on."*. The visitation requirement here refers to the number of time a patroller agent is required to visit or patrol a particular location in a given area/environment, namely *frequency of visit* or *visiting*

frequency. In other words, how patroller agents can cluster a given graph so as to balance the workload of each agent by taking into account the required visiting frequency to each node. In patrolling problem, the patrolled area is described as a graph, where a node represents a location to be patrolled/visited, and an edge represents a path between nodes along which agents move. As each location in realistic environment, e.g., security patrolling, has different visitation requirement according to the required security level, a patrolling system with non-uniform visiting frequency is preferable in real-world applications.

More specifically, we intend to address the problems remaining from the previous studies as follows:

1. Most of previous studies did not consider the patrolling problem with non-uniform visitation requirement where the required frequencies of visit to each location are different, but we consider this requirement in our study. Thus, we do not focus on the exploration algorithms; we assume such algorithms are given to all agents, and agents partition the responsible areas so that their workloads are balanced if they explore on the basis of the given algorithms.

2. The difference in visiting frequency generally causes imbalanced workload among patroller agents, leading to inefficiency. Thus, an effective clustering algorithm that can overcome this problem is needed.

3. Most of prior works could generate the route for patrolling in each allocated sub-area, yet the route generation inside each sub-area based on non-uniform visitation requirement was not taken into account.

All the above three problems have not been solved at the same time by any research. Therefore, the main goal of our study is to overcome these three aforementioned problems.

1.4 Contributions

Our study makes three contributions for the multi-agent patrolling problem as follows:

1. **The model of a frequency-based patrolling problem for multi-agent system**: it is a model where the required frequency of visit to each location is not uniform, that is, the frequency of visit can be high or low based on the realistic environment to be patrolled. This model is well-suited to the real-world applications, e.g., security patrolling, where each location has different visitation requirement or risk status according to the required security level.

2. **Frequency-based area partitioning method for balanced workload**: we developed an effective and scalable clustering algorithm for periodically visiting locations based on their visitation requirements by formulating a new k-means based approach for multi-agent patrolling system. The main objective is to balance the workload among all patroller agents, in which the visitation requirement for each location is non-uniform.

3. **Frequency-based sub-area patrolling method**: we generated the route for each agent to patrol in its allocated region, in which the cost of visiting all locations is minimized by taking into account the difference in each location's visiting frequency.

1.5 Dissertation Outline

The remaining chapters of this dissertation are organized as follows:

- **Chapter 2** provides the basic background knowledge of several approaches needed to better understand this thesis.

- **Chapter 3** reviews the related papers or work that are most relevant to our work and those that are relevant to the multi-agent patrolling problem.

- **Chapter 4** describes the area partitioning method for continuous cooperative task, using a cleaning task as an example. This chapter includes the model of agent and environment, the proposed area partitioning method with learning of dirty areas and obstacles in environments for cooperative sweeping task, the experimental setting and results, the discussion, and the summary.

- **Chapter 5** presents the area partitioning method for multi-agent patrolling task in more general framework. This chapter explains the problem formulation, the proposed frequency-based area partitioning and sub-area patrolling for balance workload in multi-agent patrolling system, the experimental evaluation, and the summary of the proposed work.
- **Chapter 6** concludes our study and gives some outlines for future direction.

Chapter 2

Preliminaries

This chapter provides the background of the approaches relevant to the main problems of this thesis. First, we present the general description of the traveling salesman problem which is a classic algorithm mainly focusing on an optimization problem. Next, a simple heuristic, namely greedy algorithm, is introduced since it is a straightforward algorithm for solving the optimization problems. Finally, we review a simulated annealing algorithm, which is one of the metaheuristic algorithms used for solving the combinatorial optimization problems, i.e., the traveling salesman problem.

2.1 Traveling Salesman Problem

The reason why we introduce the traveling salesman problem (TSP) in this chapter is that our problem is similar to the multiple traveling salesman problem (mTSP), which is a generalization of the classical traveling salesman problem and which will be explained later. The only difference between our problem and the mTSP is that in mTSP, a number of cities have to be visited by m salesmen whose goal is to find m tours with minimum total travel where all the cities must be visited exactly once, while in our problem, each location in a patrolled area must be visited based on the required frequency of visit.

2.1.1 Standard Traveling Salesman Problem

The traveling salesman problem, called TSP as aforementioned, is a classical integer programming and well-known combinatorial optimization problem [32]. Simply speaking, there are n cities where the distances between pairs of cities are known. The main goal is to minimize the total distance in which a salesman must visit each city exactly once and then return to the starting city, simply called depot. The distance from city i to j is represented by d_{ij}, such that it is measured by the cost of travel between the two cities. This travel cost can be given in a unit of length, time, or currency value.

The TSP can be formulated as a (fully-connected) undirected graph, $G = (V, E)$, in which this problem is assumed to be symmetric, where $d_{ij} = d_{ji}$. The cities are represented by a set of nodes $V = \{1, 2, \ldots, n\}$, and $E = \{(i, j) : i, j \in V, i \neq j\}$ is a set of edges denoting the paths between cities. Each edge $e_{ij} \in E$ consists of an associated weight denoted by the distance between node i and j, d_{ij}. Generally, as this is the case of planar problem where the positions of all nodes are points with coordinates (x, y), d_{ij} is the Euclidean distance from point i to j, represented by $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

2.1.2 Assignment Formulation of TSP

The formulation of TSP can be classified into two types, where the former formulation is considered in an open tour in which the salesman does not return to the starting city; the latter is regarded in a closed tour in the extent to which the salesman returns to the starting city. In this context, we are going to mention only the closed tour, whose assignment formulation is defined as:

$$Minimize \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{2.1}$$

Subject to:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall\, j \in N \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall\, i \in N \tag{2.3}$$

$$x_{ij} \in \{0, 1\}, \ \forall (i, j) \in N \tag{2.4}$$

$$\begin{aligned} u_i + u_j + (n-1)x_{ij} \leq n-2, \quad & 2 \leq i, j \leq n, i \neq j, \\ & \forall (i, j) \in E, \\ & 1 \leq u_i \leq n-1, \end{aligned} \tag{2.5}$$

where u_i and u_j denotes the visiting rank of city i and j in order respectively, and $u_1 = 0$. Both u_i and u_j are non-negative integers. Equations 2.1 to 2.4 define the associated assignment problem, while Eq. 2.5 specifies the subtour elimination constraint (SEC). The SEC is introduced to ensure that the tour is feasible, such that no subtours (loops without a depot) exist during the tour [71]. Despite this simple mathematical formulation, the TSP is not easy to solve as it is regarded as NP-hard problem.

2.1.3 Multiple Traveling Salesman Problem

A generalized variation of the TSP is the multiple traveling salesman problem, called mTSP as mentioned above. Simply stated, mTSP consists of m salesmen and n cities whose goal is to obtain m tours with a minimized total cost of travel. This problem is the same as the classic TSP, where every city is visited exactly once. Moreover, the salesmen must visit at least one city, and all of them return to the starting city. As the TSP is NP-hard problem, mTSP is also NP-hard.

2.1.4 Assignment Formulation of mTSP

The mTSP is formulated using integer linear programming formulation [55]. We initially define a decision variable which is a binary variable, denoted by:

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is used in the tour} \\ 0 & \text{otherwise} \end{cases}$$

Then, the assignment formulation of the mTSP is similar to that of the TSP with a few extra constraints, given as:

$$Minimize \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} \tag{2.6}$$

Subject to:

$$\sum_{j=2}^{n} x_{1j} = m \tag{2.7}$$

$$\sum_{j=2}^{n} x_{j1} = m \tag{2.8}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad j = 2, \ldots, n \tag{2.9}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 2, \ldots, n \tag{2.10}$$

$$\text{subtour elimination constraint} \tag{2.11}$$

$$x_{ij} \in \{0, 1\}, \ \forall (i, j) \in E, \tag{2.12}$$

where (2.9), (2.10) and (2.12) are the usual assignment constraints, (2.7) and (2.8) ensure that exactly m salesmen depart from and return back to node 1 (the depot). Even though constraint (2.8) is implicitly understood by (2.7), (2.9) and (2.10), it is introduced here for the purpose of complete ness. Constraint (2.11) is used to prevent the subtours as mentioned in the above TSP formulation.

2.1.5 Applications of mTSP

Main applications

The mTSP is most applicable to various routing and scheduling problems as these problems require the involvement of multiple salesmen. According to [55], several applications that are most popular in the literature are presented as follows:

- School bus routing problem

- Printing press scheduling problem
- Crew scheduling problem
- Interview scheduling problem
- Hot rolling scheduling problem
- Mission planning problem

Connection with other problems

Aside from the aforementioned applications, we can also link mTSP to other problems, i.e., mTSP is used to find balanced workload among salesmen in the study of [62]. Moreover, in the work of [55], the authors employed mTSP approach to deal with a workload scheduling problem by incorporating some additional constraints. Likewise, the mTSP-based for an overnight security service problem is presented by [13, 45], which is related to the task assignment for security guards to monitor a given location set based on their capacity and working hours.

2.2 Greedy Algorithm

2.2.1 Definition

A greedy algorithm is typically a simple, easy-to-implement and problem-solving heuristic for solving an ***optimization problem***. An optimization problem is a problem in which a given set of inputs are required to be minimized regarding to some constraints or conditions on the set of solutions. This problem is assumed to have n inputs as a set of candidates, $C = \{c_1, c_2, \ldots, c_n\}$, which are required to obtain a set of solutions, S, where $S \subseteq C$. A ***feasible*** solution, $c \in S$, is obtained when any subset of C satisfies the given constraints. Then, a feasible solution that satisfactorily meets the minimized or maximized condition of a predefined objective function is called an ***optimal*** solution.

The greedy algorithm generally consists of the following components:

1. A set of candidates (or input), C, which is used to generate a solution that can be a set of nodes or edges in a graph, in this study.

2. A set of solutions, S, which is a set of selected candidates from C that are considered and chosen by the greedy method to reach an optimal solution.

The decision strategy of the greedy technique makes a choice at any step without considering the future consequence, and once a choice has been rejected, it will never be reconsidered. In many cases within a reasonable period of time, this strategy may produce a local optimal solution that is often close to the global optimum solution.

Simply stated, this algorithm has several outstanding characteristics such that it is considered as a straightforward and efficient method in solving a problem. Although it does not always yield optimal solutions, it does for many other cases. Moreover, the best choice at the moment will be chosen which is hopefully expected to achieve the overall global optimum.

Algorithm 2.2.1: Pseudocode of standard greedy algorithm

Input : A set of candidates, C
Output: A set of solutions, S

```
1: S ← ∅
2: while (S is not completed) and (C ≠ ∅) do
3:     Choose the best currently available element c from C
4:     if By adding c to S, the condition is satisfied then
5:         Add c to S
6:     end
7: end
8: return S
```

2.2.2 Process of Greedy Algorithm

Considering the shortest path in a graph, basically, there exist a couple of main steps in computing the greedy approach. The first step involves the process of sorting all edges in the graph to find the nearest node to the current one. In the second step, this method will choose the shortest edge and add it to the solution set. Finally, the second step will be repeated if the maximum number of edges are not reached. To be more concise,

the simple structure of the greedy approach is demonstrated in Algorithm 2.2.1.

2.3 Simulated Annealing

Simulated annealing (SA) is a randomized local search technique that is used to prevent the process of optimization from getting trapped in a local minimum. The SA algorithm was originally developed and inspired by the process of annealing in metal work. The annealing refers to a process in which a solid is heated and then slowly cooled until its structure is eventually frozen at a minimal energy state [23].

Although SA algorithm may not produce a perfect solution, it at least can find a good solution for optimization problem. If our purpose is to deal with minimization or maximization, SA would be an ideal solution to tackle this problem. A good example to which SA can be applied is the traveling salesman problem, where the salesman is required to visit a set of cities so as to minimize the total cost of its tour. The SA has been successfully applied and adapted to give an approximate solution for the TSP.

2.3.1 Advantages of SA

There are a number of advantages in using SA. First, it can deal with highly nonlinear or stochastic problems. Second, it is a flexible optimization method that has the ability to reach global optimums, which is highly suitable for large combinatorial optimization problems. Third, it is quite adaptable as it does not restrictively depend on any property of the model. Fourth, SA is considered as a robust technique since it can get rid of trapping in the local minima. Last, this algorithm can generate a reasonably good solution for many combinatorial problems.

2.3.2 Basic Procedures of SA

It is further of importance to illustrate the basic procedures of the SA algorithm so as to get intuitive understanding on how this algorithm works, which consists of the following steps:

1. Starts with an initial solution that is randomly generated, and a high initial temperature. This solution is the current and best solution.
2. Generate a new solution randomly based on the current solution.
3. Compute the relative change in cost which is the difference between the current and new solution.
4. If the relative change in cost is less than or equal to zero, the new solution is accepted as the best solution.
5. Otherwise, the new solution is accepted in accordance with the acceptance probability that is decided on the basis of the relative change in cost and the current temperature.
6. The temperature is then reduced based on the cooling ratio function.
7. Repeat steps 2 to 6 until the stopping criterion is met. This criterion can be satisfied when the minimum (final) temperature is reached, or the number of iterations exceeds the maximum number of iterations.

In addition, Fig. 2.1 simply shows the process of SA algorithm in a flow chart structure.

2.3.3 Formulation of SA

The basic notations of the main parameters used in SA can be identified as follows:

- s_0: an initial solution
- T_0 : an initial temperature
- s: the current solution
- s': the new solution
- c: a cost function denoting the total cost/distance of a solution
- δ: a relative change in cost between s and s'
- α: a cooling rate

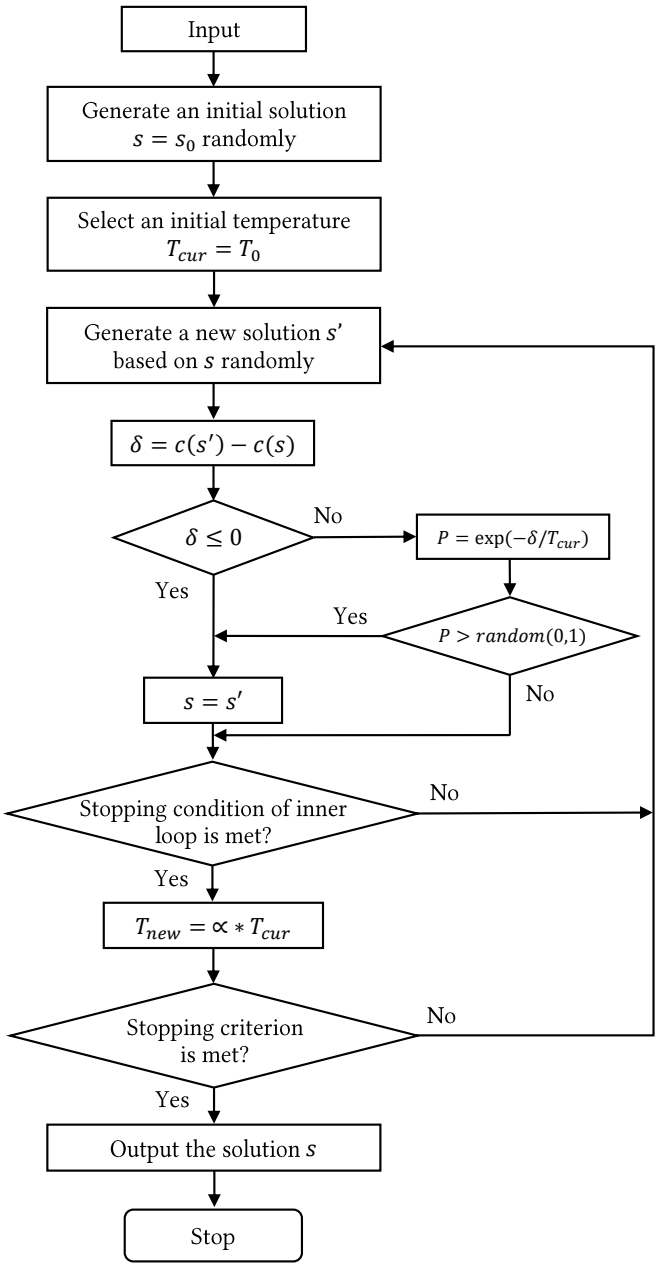

FIGURE 2.1: The flow chart of SA

- T_{cur} : the current temperature
- $T_{new} = \alpha * T_{cur}$: a cooling ratio function, where $0.8 < \alpha < 0.99$.

- $P(\delta)$: an acceptance probability function that determines the probability of choosing the worse solution, where $P(\delta)$ is calculated by:

$$P(\delta) = e^{-\frac{\delta}{T_{new}}} \tag{2.13}$$

Chapter 3

Literature Review

3.1 Review of Research Related to This Work

3.1.1 Continuous Cooperative Task

There have been a number of researches applying agents, which are software programs for autonomously generating robot activities, to cleaning and patrolling problems using single or multiple robots. For example, Ahmadi et al. [4] proposed a patrolling method where an agent is assigned to move around the areas to search for the events that happen with different probability. Yet, the authors did not study the case of collaborative movement with multiple agents. Kurabayashi et al. [46] proposed another patrolling method, called a centralized off-line method. This method consists of a single server that is able to generate the entire route for a sweeping task. Then, the route is fragmented into sub-areas for the agents to patrol so as to obtain the minimal working time of each agent. Yoneda et al. [88] proposed a distributed method in which agents autonomously decide their search/exploration strategies in a multi-robot sweeping problem using reinforcement learning. Sampaio et al. [75] proposed the gravity-based model in which the locals that were not visited for a long time have the stronger gravity, and thus, agents tend to visit such locations for uniform patrolling. Unlike our method, these

methods are based on the assumption that agents traverse a shared area along different routes or with different exploration algorithms.

Luo et al. [50] proposed a cooperative sweeping strategy of complete coverage path planning for multiple cleaning robots in a time-varying and unstructured environment. It used biologically inspired neural networks, and each cleaning robot treats other robots as moving obstacles. This approach is capable of autonomously planning collision-free cooperative path for multi-robot in unstructured environment. However, they did not discuss how robots divide their responsible cleaning areas for a balanced performance. Sugiyama et al. [82, 83] proposed the control method for coordinated cleaning tasks with learning of such information. However, their method did not segment the area but selected appropriate path-planning strategies for moving around in the shared environment.

Portugal et al. [67] introduced a multi-level subgraph patrolling algorithm based on balanced graph partition for efficient multi-robot patrolling in a known environment. Hert et al. [37] tried to partition the environment into n equal size parts. Bast et al. [9] also tried to partition the area into equal size parts but with the additional condition that the parts do not have any acute angles. Unlike ours, these works use heuristics for good partitioning rules, such as parts that are equal-size or without acute angles. Jager et al. [41] partitioned the environment into polygonal regions. Each agent requests to clean a region, and others respond to it when they have done it. This will result in an unpredictable area partitioning because while a robot is requesting a polygon, it does not consider the whole region that it has to sweep. Thus, this method is suitable only for single-agent sweeping applications.

We can formalize the patrolling problem from more theoretical perspective. For example, Chevaleyre et al. [18] formalized the patrolling problem by considering the case of traveling salesman problem with multiple agents. The authors then tried to compare the number of cycle in which agents move and the route division methods to find the minimal length of routes. Elmaliach et al. [26] proposed an algorithm that finds the shortest Hamiltonian cycle in grids, which are used to patrol in the areas and to spread the agents there evenly. These methods also assume that robots move in the shared areas. Furthermore, most of these studies

did not consider the case of agents visiting the locations at different frequencies.

Another approach is to partition the area into subareas so that agents can divide the labor. Based on their previous work [4], Ahmadi et al. [5] introduced the extended method by including multiple patrolling agents in which the responsible region of individual agent is segmented based on the exchange of boundary information. Furthermore, agents visiting a boundary region more frequently tend to take charge of the region. Voronoi-based techniques are also another method that do not require graph descriptions of the environments, such as [12, 22, 77]. However, these require computational costs that limit their applicability.

Bio-inspired computation models are also used to cover the areas. Ranjbar-Sahraei et al. [69] introduced the indirect communication using pheromone-based stigmatic communications to identify the regions that should be covered. McCaffery [56] proposed the graph partitioning algorithm using the foraging behaviors. The resulting subgraphs are allocated to agents so that they cover the whole environment. Elor et al. [27] introduced a segmentation method based on the integration of the ant pheromone and balloon models for covering a region. That is, a region is divided into sub-regions which are then allocated to different agents. The authors assumed that each sub-region is a balloon, which is the pressure that indicates the size of the sub-region. Then, the use of pheromone communication model allows an undirected exchange of the pressure values. However, the differences in agent performance and the environmental characteristics are not considered in these methods, thus the region is likely to be divided into equal-size sub-regions. Furthermore, the implementation of pheromone communications in decentralized multiple-robot applications is not trivial. Kato and Sugawara [44] proposed the method, *performance-based partitioning* (PBP), for partitioning a given area so that agents keep the environment evenly clean by performing the cleaning task in a balanced manner by taking into account these differences although it is not a bio-inspired approach. However, unlike our method, it assumes that agents have knowledge about what areas are easy to be dirty in the environment, but providing this knowledge in advance is difficult because it depends on many factors.

Our work is different from the others because we focus on how agents identify their RA autonomously based on the complexity of the shapes of obstacles and the characteristics of environment so that they can share the tasks in a balanced manner.

3.1.2 Patrolling Task

Multi-agent patrolling problems have been investigated and studied by many researchers, e.g., [6, 11, 17, 19, 36, 49, 51, 52, 54, 74, 80]. Initial researches [7, 18, 20] presented a theoretical analysis of various strategies for multi-agent patrolling systems and an overview of the recent advances in patrolling problems. Portugal et al. [67] proposed a multi-robot patrolling algorithm based on balanced graph partition, yet this paper did not consider when the required frequency of visit is not uniform. The same author, then, addressed a theoretical analysis of how graph partition and cyclic-based techniques perform in generic graphs [66]. A survey of multi-agent patrolling strategies can be found in [68], where strategies are evaluated based on robot perception, communication, coordination and decision-making capabilities.

Chao et al. [16] presented a heuristic for the team orienteering problem in which a competitor starts at a specified control point trying to visit as many other control points as possible within a fixed amount of time, and returns to a specified control point. The goal of orienteering is to maximize the total score of each control point, while in our patrolling problem, the main goal is to minimize the difference in workload amongst all patroller agents. Sak et al. [73] proposed a centralized solution for multi-agent patrolling systems by presenting three new metrics to evaluate the patrolling problem. Popescu et al. [65] addressed the problem of multi-agent patrolling in wireless sensor networks by defining and formalizing the problem of vertex covering with bounded simple cycles (CBSC). This approach consequently considered polynomial-time algorithms to offer solutions for CBSC. Mao et al. [53] investigated multi-agent frequency based patrolling in undirected circle graphs where graph nodes have non-uniform visitation requirements, and agents have limited communication.

Elor et al. [27] introduced a novel graph patrolling algorithm, in which the region is divided into sub-regions that are allocated to each agent. However, this method partitioned the region into equal-size sub-regions. As the characteristic of the area is not always uniform, equal-size sub-areas are inappropriate. Elmaliach et al. [26] presented a centralized algorithm in a non-uniform grid environment which ensures optimal uniform frequency, such that every cell is visited with maximal and uniform frequency. However, grid-based representation has a limitation in dealing with partial-occluded cells, or covering close-to-boundary areas in the continuous spaces.

Sugiyama et al. [81] also introduced an effective autonomous task allocation method that can achieve efficient cooperative work by enhancing divisional cooperation in multi-agent patrolling tasks. This paper addressed the *continuous cooperative patrolling problem* (CCPP), in which agents move around a given area and visit locations with the required and different frequencies for given purposes. However, this paper did not consider area partitioning and was implemented in a 2-dimentional grid space.

The most relevant work to ours is the work of Karimov et al. [43], which introduced a new hybrid clustering model for k-means clustering, namely HE-kmeans, to improve the quality of clustering. This proposed model integrated *particle swarm optimization*, *scatter search* and *simulated annealing* to find good initial centroids for k-means. Another relevant work is from Ogston et al. [61], which proposed a decentralized clustering method by extending the traditional k-means in a grid pattern. These two approaches could produce a good quality of clustering. However, they did not consider when the frequencies of visit to each location are different. As the frequencies of visit in the real-world environment are not always uniform which makes the clustering imbalanced, a clustering method that can take into account the non-uniform frequency of visit and at the same time tries to balance the workload amongst all patroller agents is preferable for realistic applications. Our proposed method, thus, aims at dealing with these requirements

3.2 State of the Art of Multi-Agent Patrolling Problem

This section describes some research papers that are not directly related with this work. The goal of describing these work is to give an overview of the techniques, paradigms or methods used so far to solve the multi-robot patrolling problems.

3.2.1 Cyclic Patrolling Model

Elmaliach et al. [26] tackled the problem of generating patrol paths for a group of mobile robots inside a continuous target area. This target area is divided into a grid which is associated with a terrain that takes into account directionality and velocity constrains. Therefore, the terrains and the terrain grids considered in that work are directionally non-uniform. In these types of terrains, each point is associated with a cost which depends on the direction in which robots can travel. As a consequence, robots have velocity limitations which depend on both the terrain and the traveling direction.

In that work, a patrolling model called *Cyclic* is developed to generate a cyclic patrolling path that visits every point in a given area exactly once. A path with these characteristics is called a Hamilton cycle [64]. The cyclic patrolling model uses a spanning tree coverage method to find the Hamilton cycle required to patrol the terrain. The terrain could have more than one Hamilton cycle. The cost of all these cycles is the same when the terrain is uniform. However, the opposite is true when the terrain is non-uniform. In that case, the cyclic patrolling model must select the minimal circular path of minimal cost which is called minimal Hamilton cycle. A minimal Hamilton cycle is a circular path that visits all points exactly once in the terrain with the lowest cost. The maximal uniform frequency in the terrain is guaranteed by selecting this minimal Hamilton cycle, *i.e.*, each point is covered with the same optimal frequency. This nature of cyclic patrolling model suggests to Elmaliach et al. to propose a criterion based on frequency optimization to evaluate multi-robot patrolling models. Note that the patrolling model presented in that work assumes that a topological

representation of the whole patrolling environment is available. However, to assume that such representation is known a priori, it is not appropriate because there are several situations in which such assumption is not true.

Once a cycle is obtained, cyclic patrolling model assigns an initial position to each robot from which they start to patrol the terrain. This assignation considers the minimization of the maximal distance traveled by every robot from its current position to the assigned position. This is done to allow robots to arrive at their initial positions in the minimal time. These initial positions are points distributed uniformly along the Hamilton cycle path. As a result, the distance between every two consecutive robots is the total weight of the cycle divided by the number of robots. This consideration yields an equal distance between every two consecutive robots. Finally, cyclic patrolling model instructs all robots to patrol along this cycle in equidistant relative positions. Clearly, the manner in which robots patrol the terrain makes the solution developed by Elmaliach et al. [26] completely deterministic, and therefore predictable. The solution is predictable because robots follow the same cycle over and over again. Indeed, if all robots start to patrol in the same point, all of them will visit the same point in the same time. Moreover, the criterion proposed in that work suggests that all the points of the terrain will be visited at the same period of time. Therefore, this criterion makes more predictable the behavior of robots and for security purpose, a predictable solution is not appropriate. This is because an intruder, no matter how intelligent, can easily deduce how a point of the patrolling environment, or even worse the whole patrolling environment, is protected. The intruder can then use this information to plan an attack.

On the other hand, Elmaliach et al. [26] claimed that the cyclic patrolling model is robust in the sense that the uniform frequency of the multi-robot patrolling task is achieved as long as one robot continues working properly. In this sense, if one robot fails, the other robots simply divide the circular path considering the number of robots minus one. However, the patrolling model purposed in that work depends on a central and explicit coordinator scheme. A centralized solution has a couple of disadvantages, including lack of scalability in protecting the number of locations and its likelihood to be influenced by a single-point failure because of its unique control point. Moreover, centralized, predefined and

fixed schemes are not suitable for security applications in some situations such as dynamic patrolling environments, huge graphs and patrolling environments where regions have different priorities. In fact, adding or removing new nodes requires the generation of new patrol paths.

3.2.2 MSP Patrolling Model

Portugal et al. [67] presented a patrolling model called *Multilevel Subgraph Patrolling*, or simply *MSP*. This patrolling model uses a balanced graph partitioning method to divide the patrolling environment into regions with the same dimension according to the number of robots used to protect the patrolling environment. Nevertheless, no partitioning is needed when only one robot is used, and a patrolling scheme for the whole graph is implemented. This method provides partitions from two up to eight balanced graph regions. Every region is represented by a sub-graph extracted from the topological representation of the global patrolling environment. Each of these regions is assigned to a robot that follows a local patrolling route which depends on the sub-graph topology. The patrolling model for generating this patrolling route typically searches for Euler and Hamilton circuits and paths. Euler circuits and paths are paths that visit all the edges of the graph exactly once. The difference between Euler circuits and paths is that the former start and end on the same vertex, while the latter do not. The Hamilton circuits and paths visit all the graph nodes exactly once, and only the Hamilton circuits begin and end on the same node. The search for these circuits and paths have the disadvantage that it is hard to find them. Besides, most of the graph do not have them.

If the optimal Euler or Hamilton circuits and paths do not exist, the patrolling model searches for the longest paths and non-hamiltonian cycles. The longest path starts and ends in vertices with only one neighbor, also called one-degree vertices or leaf vertices. In this case, the patrolling model builds a list with all the leaf vertices of the graph. From this list, the start vertex and the end vertex are selected and the patrolling model searches for a longer path. This step is performed several times with different start and end vertices. Finally, the best path found by the patrolling model is selected, i.e., the longest path. Non-hamiltonian

cycles are selected only when they have at least half of the vertices of a graph; if not, the patrolling route remains the longest path. Since the longest path and the non-hamiltonian cycle do not contain all vertices of the graph, the procedure includes such vertices to complete the patrolling route. Then, ultimately inverse path procedure is used to return to the starting vertex of the route when is required. This path has the same vertices that the original path but in an opposite direction.

In that work, Portugal and Rocha claimed that tracking the path of all the robots and predicting better regions to be intruded in the patrolled environment are even more difficult. Nevertheless, an intruder does not need to know the paths of all robots to perform an attack. The intruder can attack the infrastructure only by knowing the path of one robot. Although robots follow their own patrolling cycle, this cycle is deterministic and therefore predictable. Additionally, the fault-tolerance mechanism of that patrolling model depends on a central coordinator which recalculates the paths without considering the robot that has failed. However, if the central coordinator fails, the fault-tolerance mechanism fails too. Note that similar to cyclic patrolling model, the patrolling model presented in that work assumes the availability of a topological representation of the entire patrolling area. However, as aforementioned, this assumption is not always appropriate.

Single cyclic and MSP patrolling models demonstrated the effectiveness of the patrolling models that implement solutions based on cycles and partitioning [18, 58]. The suitable performance of those patrolling models can be explained by their centralized coordinator scheme [7].

3.2.3 Adaptive Solutions

Sempé et al. [79] proposed a reactive and adaptive patrolling model to solve the multi-robot patrolling problem. To manage this problem, the patrolling environment is divided into zones which are called regions. The whole patrolling environment is represented by a graph in which each region is a vertex and the edges represent connections between adjacent regions. In that patrolling model, robots share a virtual patrolling environment which is used to propagate the visiting value of each region

among them. This visiting value represents the time that a given region has not been visited by any robot. The higher the visiting value of a region, the higher the time that such regions remains unvisited. Therefore, this patrolling model is based on a descent gradient method in which the robots are driven by the propagated visiting values to the least visited regions. Once a region is visited, its visiting value is dropped to zero. In that work, the authors take into account that robots must gather information for a given region which takes a time called *visit duration*. Another robot constraint that is considered in that work is the energy management, e.g., robots need to charge their batteries. The patrolling model presented in that work is evaluated carried out simulated experiments and real-world experimentation with three pioneer 2DX robots.

3.2.4 Negotiation Methods

Almeida et al. [7] tackled the patrolling problem with negotiations methods. To this end, the patrolling environment is represented by a graph. Initially, each robot receives randomly a set of vertices of this graph to patrol in the beginning of a simulation. Note that this set could have separate or close vertices. In this context, robots aim at getting a set of vertices as close as possible to minimize the time between two visits to the same node and increase their utility. The utility function of robots only considers the distance between vertices. To fulfill this requirement, robots offer through auctions the vertices that cannot be visited within a reasonable amount of time. Robots that receive such offer are called *bidders*. The bidders verify whether they can trade the offered node by bidding a node from their own set. In the case of several bids, the auctioneer must choose the best bid and make the deal with the bidder. The best bid represents the nearest vertex from the other vertices in the set of the auctioneer. By using this mechanism, Almeida et al. presented six market-based multi-robot patrolling models. These patrolling models differ in the manner in which robots perform their auctions. There are three differences in auctions. Firstly, the auctions are either one or two shots or rounds. Secondly, the utility function of the auctioneer determines the value of node on auction

that is, for example, a private value. Finally, the bidder does not know the bid of others which is called sealed-bid.

Menezes et al. [58] presented other negotiator patrolling models and compared them with the ones described by Almeida et al. The mechanism used in both works is the same with five variations introduced by Menezes et al. First, vertices assigned randomly at the beginning of the simulation are neighbor vertices instead of global vertices. Second, an algorithm called insertion sort was used to determine which node should be auctioned. Third, the behavior of robots can be self-interest or cooperative. A cooperative robot trade one node by another that decreases its utility if such exchange increases the utility of the group. Fourth, robots avoid offering always the same vertices by selecting randomly a node from their own set of vertices every specific time. Finally, robots can offer up to two vertices to other robots, i.e., they can exchange two vertices by other two, two vertices by one, or one-by-one.

The comparison carried out in that work showed that the centralized patrolling model developed by Chevaleyre [18] performs better than the negotiation patrolling models in almost all cases of study. However, the negotiation-based patrolling models have characteristics to highlight such as distribution, reactivity, adaptability, scalability and stability.

3.2.5 Swarm Intelligence Optimization

Swarm intelligence optimization, generally speaking, is a bio-inspired paradigm that mimics the mechanisms of the ants. In this paradigm, the ants have the ability to use the patrolling environment as a shared memory. This is done by dropping and sensing pheromones which define information in a temporary way due to the evaporation process and establish an indirect communication system. The individual behaviors performed by the ants allow the developing of decentralized patrolling models.

Glad et al. [29] proposed a patrolling model based on this paradigm to address the patrolling problem. In that work, the patrolling environment is not known in advance and is represented with a grid. Each robot has a local perception of this patrolling environment which is used to mark and choice an action to move. The number of robots used in that work to

perform patrolling tasks in the patrolling environment change dynamically. The patrolling model presented in that work is called Exploration Vertex Ant Walk (EVAW). EVAW is a pheromone-based patrolling model which relies on the basics of other two patrolling models, namely, EVAP [21] and VAW [84].

In EVAP and VAW, robots self-organize and each of them reaches a stable cycle. This fact is due to the local behavior of robots which is similar in both patrolling models. This behavior is based on a digital pheromone model in which pheromones are represented as numbers. The values of these pheromones decrease over time to simulate the evaporation process of biological pheromones. To perform this process the patrolling environment evaporates pheromones with rate ρ. The remaining value of a pheromone represents the time elapsed since the last visit to the cell related to such pheromone. Robots can perceive and move only between neighboring cells. This neighborhood is represented by the four adjacent cells of the actual position. Moreover, robots perform two actions when they visit a cell of the grid. First, they move to the next cell according to the negative gradient of the pheromone by choosing in the surrounding neighborhood the cell with the minimum value. Thus, the agents necessarily choose the one which has not been visited for the longest time. Second, they drop a pheromone in the actual cell. Even though EVAP and VAW are similar, they differ in two aspects. The first difference relates to the information of the dropped pheromone. In EVAP, robots drop a pheromone of quantity Q_{max}, whereas in VAW the dropped information is the date of the visit. As a result, in VAW robots must have synchronized time counters and start at the same time with counter $t = 0$. The second difference relates to the order in which the operations move and drop are performed. In EVAP, robots drop the pheromone and then move, whereas the opposite is true in VAW. With this subtle difference EVAP favors exploration in the multi-robot case. On the other hand, VAW time computation is easier to manipulate. EVAW uses the order of operations of EVAP and the math formulae of VAW.

Wagner et al. [84] presented an enhanced version of VAW. In that patrolling model, robots use pheromones made up a pair (μ, τ) in which μ is the number of visits to the cell so far, and τ is the last time that the cell was visited. In the single agent case, Wagner et al. proved that when a Hamiltonian cycle is reached, the ant repeats it forever.

Ahmadi et al. [5] proposed a patrolling model based on negotiations to solve the task called *Continuous Area Sweeping*. In a continuous area sweeping task, a group of robots must repeatedly visit all points in a fixed area possibly with non-uniform frequency. This task is closely related to other two tasks called *security sweep* [42] and *sweeping* [46]. Clearly, continuous area sweeping and patrolling are the same tasks. However, the research articles that tackled the continuous area sweeping task are not included in the surveys of the literature related to the patrolling tasks. This could be caused by the use of different words even though the task is the same. On the other hand, in that work the authors extend a single-robot patrolling model [4] to the multi-robot case. To this end, the overall dynamic area is partitioned among robots and each one sweeps its part of the patrolling environment using the single-robot area sweeping method. The area is dynamic because it is considered factors such as addition of new robots, robot malfunctions, change in robot speeds or changing distribution for event appearances. That work is tested with simulations and implemented on physical robots.

Finally, Lauri et al. [48] introduced a patrolling model based on ant colony optimization (ACO). The patrolling model presented in that work is combined with an evolutionary algorithm technique. This combination allows that several ant colonies compete to find out the best multi-robot patrolling strategy dispersed efficiently over a graph. That patrolling model performs two stages to achieve the previously specified goal. In the first stage, the evolutionary algorithm is used to find the most distant vertices of a graph. In the second stage, the ACO patrolling model carries out the patrolling tasks [47].

3.2.6 Reinforcement Learning

Machine learning techniques such as reinforcement learning can be used to coordinate the actions of a group of robots when such coordination depends upon the topology of the environment. This is because reinforcement learning allows an automatic adaptation of the robots to the environment.

Santana et al. [76] investigated the creation of adaptive robots that learn to patrol using reinforcement learning techniques. In that work, the

patrolling task was defined by adopting an abstract representation of the terrain as a graph. For the single-robot case, the reinforcement learning framework is defined over the theory of markov decision processes. In this theory, robots act according to some policy which represents the probability of choosing an action from a state. This selection aims at maximizing a long-term performance criterion which is defined as a sum of a discounted reward. The local reward used in that work depends only on the idleness of the node currently visited by the robot. Because of this such reward does not assume anything about the whole environment. In order to include the edges of different length into this reward is used a discrete-time finite semi-markov decision process framework. These frameworks can be solved through the use of an algorithm called Q-Learning.

The extension of this patrolling model to the multi-robot case is based on a concept called *individuals learners*. An individual learner solves a collective optimization problem by solving local optimization ones. Two reward models are used to solve these optimization problems in the multi-robot case. In the first model, called *Selfish Utility*, robots do not help to maximize the rewards of the other robots. In the second one, called *Wonderful Life Utility*, robots received penalties when they compete for the idleness of the same node. On the other hand, two communications schemes were developed to tackle the non-determinism produced by the multi-robot case. In the first communication scheme, called *Black-Box*, robots communicate by placing flags every time that they visit a node. In the second one, called *Gray-Box*, robots communicate by flags their intentions upon actions.

Preliminary results showed that the architecture that uses the Selfish Utility model and the Gray-Box communication scheme obtained the best performance. The comparison between that architecture and previous ones showed that the former performs better than the later in 80% of the cases of study. Besides these results, the architectures presented in that work are distributed and adaptable.

3.2.7 Traveling Salesman Problem

Chevaleyre [18] proposed several strategies to solve the multi-robot patrolling problem by using cycles and closed-paths. In that work, the

territory to be protected is depicted by an undirected graph. A closed-path is a path represented by a list of vertices that start and end in the same node and cover the edges of a graph possibly more than once. A graph could have more than one closed path. Among these paths, the smallest one that cover all vertices of the graph is the best solution. The closed-path with these characteristics is called cycle. A cycle is calculated as the optimal solution for the well-known traveling salesman problem (TSP). Thus, for the single-robot case, a cyclic strategy consists in traveling along the calculated cycle indefinitely.

To extend the single-robot cyclic strategies to the multi-robot case, the robots are distributed along the smallest closed-path. The distance between robots is the same for all of them. In the multi-robot case, besides of the TSP strategies, the author studied strategies based on partitioning. To this end, the territory is partitioned into several regions, and each robot is assigned to patrol inside a single region. The experimental results of that work showed that the cyclic strategies based on TSP perform better than the partition-based strategies. In the literature, the patrolling model of Chevaleyre is referred to as *Single Cycle*. Finally, another contribution of that work is a theoretical analysis of the patrolling problem [20].

Chapter 4

Area Partitioning Method for Multi-Agent Continuous Cooperative Tasks

4.1 Introduction

The development of computer science and technologies, nowadays, has been dramatically increasing day by day and has been used in many application domains including the field of robotics and automation. Since robots can often perform tasks well in a variety of environments, the demand for robotic applications has been growing. This has led to the massive of software applications in the field of robotics combining computer and sensor technologies. Some robots are able to work in places and situations that are inconvenient and dangerous for humans, such as nuclear plant search, interplanetary exploration and disaster relief, while others can perform the daily functions, e.g., cleaning/sweeping and security surveillance robots. The cleaning and patrolling robots are examples of the hottest applications. In particular, the cooperation of multiple cleaning robots has now become very crucial because effective cooperative cleaning of multiple robots can improve the working quality and reduce the time for cleaning by sharing tasks. The control for cleaning is also applicable to other sweeping tasks, such as security patrolling and area search. This has made many researchers focus their

studies on multiple cleaning robots in different issues, such as path planning (or graph exploration), area partitioning, map building, autonomous navigation, self-localization, and obstacle/collision avoidance.

The real-world environments where agents operate are diverse, so it is almost impossible to design a system by completely anticipating the environmental characteristics in the design stage. For example, in the cleaning task, there are a number of locations where dirt may tend to easily accumulate, and these locations depend on many factors such as the shape of the environment and the locations of furniture and fixtures. Similarly, in security applications, the locations near entrances, near windows and around safes should be kept more secure than other locations. This means that agents for cleaning or security have to visit each location with non-uniform frequency in a given area based on the characteristics of environment. Furthermore, the agents may be most-advanced or old models and may have been developed by different makers; this means that they have different hardware/software capabilities, and thus, exhibit different levels of performance. Therefore, the agents must cooperatively work by considering these differences so as to complete the tasks in a more efficient and balanced manner.

Regarding the cleaning and security tasks, two conventional approaches are used to implement the patrolling activities in both coordinated and cooperative ways. Agents in the first approach work together by sharing and cleaning the given area in a coordinated manner. For instance, either different cleaning algorithms or visitation cycles can be applied to ensure that the agents are able to cover the entire area [18, 46, 88]. Another strategy for this approach is for the agents to move around the area in formation (e.g., [3, 25, 57]). However, in these approaches, an agent's strategy affects the other's, and this interaction makes cooperation complex. In the second approach, agents partition a given area into sub-areas, such that each agent is in charge of each allocated sub-area [4, 27]. However, it is non-trivial to fairly perform a division in the latter approach; if the characteristics of the area are non-uniform and the agents have different capabilities, the responsible sub-area for each agent should not be equal to achieve the balanced workload.

Therefore, this work proposes a new approach which allows the agents to fairly and autonomously allocate their tasks based on their capabilities and each subarea's characteristics. If new agents are added, agents autonomously reconfigure their subareas through coordinated interaction over time. The main concept is that each agent keeps its recently-visited location and calculates its *expansion power* that expresses the remaining capacity/power when it believes its responsible area has almost been cleaned or patrolled, which depends upon the degree of task completion, such as the expected amount of dirt remaining in its subarea in cleaning tasks and the number of important locations to keep them secure. Next, it negotiates with the neighboring agents to readjust their responsible sub-areas so that they can balance the cleanliness of the whole area. However, it is difficult to identify in advance which agents that have different hardware/software capabilities perform better in the environment and what areas are easy to be dirty. Thus, our study aims, using a cleaning task application, at the proposal of coordination method for area partitioning without this kind of knowledge.

Kato and Sugawara [44] proposed the method, called *performance-based partitioning* (PBP) along this line, but they did not examine whether the method could reflect the differences between agents' algorithms into the area partitioning. Furthermore, it assumed that knowledge about what areas are easy to be dirty and where obstacles are was given to all agents. However, providing this knowledge in advance is difficult because the easy-to-dirty areas depend on many factors such as locations of objects, intake/exhaust vents, doors and windows in the environments. Furthermore, the locations and shapes of obstacles differ in individual environments and may change, it is not easy to accurately specify their information in advance. Thus, we eliminated this assumption and extended the previous method by adding (1) the learning capability to agents for identifying easy-to-dirty areas and (2) the function to find and maintain the locations of obstacles through their operations [78]. We also show the detailed results with extensive experiments in this chapter.

The work in this chapter is an extended approach based on the previous studies [44, 87, 88]; thus, the model and problem description are based on those proposed in these papers.

4.2 Model and Problem Description

We will describe the models of environment and agents, and then state our problem addressed in this chapter.

4.2.1 Models of Agent and Environment

An agent here is a control program installed on a portable cleaner robot capable of autonomously deciding its actions and sending/receiving messages. We assume that agent has a map (graph) of the area, which may generally be unknown. This assumption can be made in this study, thanks to the fact that previous studies [31, 34, 85] have already proposed a number of algorithms for generating a map, identifying agents' locations, and avoiding collisions. We also apply this assumption in this work, for our study mainly focuses on area partition which is autonomously learned by the agents to obtain a balanced task division.

Let $A = \{1, \ldots, n\}$ be a set of agents. The agents move around the area which is represented by a connected graph with obstacles, $G = (V^+, E, O)$, where V^+, E and O ($\subset V^+$) denote the sets of nodes, edges, and obstacles, respectively. A node in O is called the *obstacle node*. In general, we assume that a number of obstacles, $\{O_i \mid O_i \subset V^+ \text{ for } 1 \leq i \leq k, O_i \text{ and } O_j \text{ are disjoint, and } O_i \text{ is the connected set}\}$ exist in the environment and we define $O = O_1 \cup \cdots \cup O_k$. The edge that connects nodes $v_i, v_j \in V^+$ is denoted by $e_{i,j}$. We then define a discrete time with a unit called a *tick*. An agent moves between nodes in $V = V^+ \setminus O$ and cleans each node it visits. Without imposing further restrictions on the problem, the length of an edge in E is assumed to be one which allows an agent to move along an edge from a node to another and clean the visited node in one tick. However, it cannot move to any node in O. We assume that $V \setminus O$ is connected, i.e., for $\forall v, w \in V$, at least one path from v to w consisting of only non-obstacle nodes exists.

4.2.2 Model of Dirt Accumulation

We represent the degree to which dirt is easy to be accumulated per tick at node $v \in V$. The *amount of accumulated dirt* at v at time t, $L_t(v)$, is initially defined as $L_0(v) = 0$ for $\forall v \in V$ and is updated by:

$$L_t(v) \leftarrow \begin{cases} L_{t-1}(v) + 1 & \text{with probability } p_v \\ & \text{(a piece of dirt is} \\ & \text{accumulated at } t) \\ L_{t-1}(v) & \text{otherwise,} \end{cases} \tag{4.1}$$

where event probability p_v ($0 \leq p_v \leq 1$) is called the *dirt accumulation probability* (DAP) for v. Yet, if node v has been visited by an agent at time t; then node v is cleaned, so $L_t(v) = 0$. Note that agent i cannot know the actual value of $L_t(v)$ except the current position, v_t^i.

Each agent has a *responsible area* (RA) which it tries to keep clean. Particularly, the connected subgraph $G_t^i = (V_t^i, E_t^i)$ represents the RA of agent i at t, where $V_t^i \subseteq V$ and $E_t^i = \{e_{i,j} \in E \mid v_i, v_j \in V_t^i\}$. We assume that $v_{base}^i \in V_t^i$ and V_t^i, and V_t^j are disjoint for i, j ($\in A$ and $i \neq j$). The size of each agent's RA, $|V_t^i|$, can be changed to uniformly keep the area clean in a cooperative manner.

4.2.3 Performance Measure

The purpose of cleaning tasks is to minimize the amount of pieces of dirt in the environment without neglecting them. Hence, we use the *sum of the amount of remaining dirt* in the entire area at specific time intervals as the performance measure of the agents' collective tasks. This is defined as:

$$D_{t_s,t_e} = \frac{\sum_{v \in V} \sum_{t=t_s}^{t_e} L_t(v)}{t_e - t_s}, \tag{4.2}$$

where positive integers t_s and t_e represent the starting and ending times of the interval, respectively. The smaller performance value D_{t_s,t_e} is, the better agents can keep the area clean. Thus, agents aim at minimizing this value.

Although D_{t_s,t_e} is an important measure, we also consider the balanced task allocation for cooperative cleaning in which agents that can handle

more work take part of the RAs of other agents that are busy and/or that have less efficient exploration algorithms. Thus, it is of importance to focus on the sizes of the RAs, V_t^i, to investigate whether or not an efficient agent could do more work in a larger RA, and calculate the amount of remaining dirt in i's RA, which is denoted by D_{t_s,t_e}^i. D_{t_s,t_e} and D_{t_s,t_e}^i are often denoted by D and D^i if there is no confusion. Note that balanced task allocation does not necessarily mean equal size of V_t^i.

The proposed probabilistic model of dirt accumulation can also be modified for other patrolling domains such as surveillance. For example, the important locations that require high-level security, such as around safes and entrances/exits correspond to the dirty areas, thus they have higher probabilities, p_v. Furthermore, we can change these probabilities in accordance with time of day. So, for example, agents can visit the important locations more frequently during nighttime hours.

4.2.4 Battery Consumption and Charge

Let B_{max}^i be a positive integer representing the maximal battery capacity of agent i at time t. Similarly, let $b^i(t)$ denote the remaining battery power. We further assume that a constant amount of power per tick, B_{drain}^i, is consumed by agent i when it moves around. Thus, $b^i(t)$ is updated using

$$b^i(t+1) \leftarrow b^i(t) - B_{drain}^i \tag{4.3}$$

every tick. Therefore, M_i is the *maximum running time* that agent i can continue to operate at most $\lfloor B_{max}^i / B_{drain}^i \rfloor$ ticks. Moreover, agent i charges its battery at its charging base, $v_{base}^i \in V$. The required time for a fully charge which begins at time t, $T_{charge}^i(t)$, is proportional to the battery consumption, defined as:

$$T_{charge}^i(t) = k_{charge}^i (B_{max}^i - b^i(t)), \tag{4.4}$$

where $k_{charge}^i (> 0)$ is the proportionality factor indicating the speed of charge. The full-battery agents start moving around and performing the cleaning tasks in their RAs. Before the battery becomes empty, they always return to their charging bases so as to recharge their batteries.

This *cleaning cycle* is followed and repeated to keep clean the allocated areas.

For every node $v \in V_t^i$, agent i calculates the minimal capacity of battery required to return to i's charging base v_{base}^i, called the *potential*, in which $\mathcal{P}(v)^i$ represents the potential of v for i and is defined as:

$$\mathcal{P}(v)^i = d(v, v_{base}^i) \cdot B_{drain}^i, \tag{4.5}$$

where $d(v, v_{base}^i)$ is the length of shortest path within the RA of i. Since we assume that agents have the knowledge of G in advance, they are able to identify the shortest path using A* or Dijkstra's algorithm. We set a condition in which agent i can safely move to the neighbor node v at time t if

$$b^i(t) \geq \mathcal{P}(v)^i + d(v_t^i, v).B_{drain}^i, \tag{4.6}$$

where v_t^i denotes the current node that i is located. This condition implies that if the next node is safe, agent i will move to that node; otherwise, it will return to its charging base along the shortest path and recharges.

4.3 Extended Performance-Based Partitioning Method

We describe the proposed *extended performance-based partitioning* (ePBP) method, which fairly partitions the given area by taking into account the performances of the individual agents and the characteristics of the area. In our proposed method, we assume that agents have information of V^+ and E but do not know (1) the set of the DAP of nodes, $\{p_v | v \in V\}$ nor (2) the set of obstacles, O (initially agents assume that $O = \emptyset$). Therefore, agents with ePBP concurrently learn the DAPs of their RAs to see which locations in the RAs are easy to become dirty, and the set of obstacles while they decide and negotiates the responsible area with other agents.

4.3.1 Area Partitioning

4.3.1.1 Expansion Power

Although agents do not know the values of p_v for $\forall v \in V$, if agents estimate the values of p_v for $\forall v \in V$, i can estimate $L_t(v)$ using the expected amount of accumulated dirt on v, which is calculated by:

$$E(L_t(v)) = p_v^i \cdot (t - t_v^i),$$

where p_v^i is the estimated value of p_v by learning of the dirt accumulation in i, and t_v^i is the most recent time when i visited and cleaned node $v \in V_t^i$; if node v is never visited by i, t_v^i is then regarded as the time by which v was included in its RA, G_t^i. How i calculates p_v^i is explained in Section 4.3.3. We also define $L_t(V_0) = \sum_{v \in V_0} L_t(v)$ and $E(L_t(V_0)) = \sum_{v \in V_0} E(L_t(v))$ for a set of nodes $V_0 (\subset V)$.

When agent i returns to its charging base at a specific time t, its *expansion power* for the current RA will be calculated. Intuitively, it expresses how efficiently i could have covered the current RA during the latest cleaning cycle. The expected amount of accumulated dirt in each RA at time t is initially computed using

$$E(L(G_t^i)) = \sum_{v \in V_t^i} E(L_t(v)) = \sum_{v \in V_t^i} p_v^i \cdot (t - t_v^i). \tag{4.7}$$

Then, i calculates the expansion power $\xi(i,t)$ of i at time t which is also known as the inversion of the expected value, denoted by:

$$\xi(i,t) = E(L(G_t^i))^{-1}. \tag{4.8}$$

If $E(L(G_t^i)) = 0$, $\xi(i,t)$ is set to a sufficiently large number. The computation of expansion power is reserved until the next calculation.

4.3.1.2 Expanding of Responsible Areas

We consider the cleaning cycle of each agent begins by leaving its base node with full battery to clean each RA using its own exploration algorithm. Each agent will decide to expand its RA in case that it realizes the RA

has been mostly cleaned. Agent i will make the decision based on the expected amount of accumulated dirt in its RA at a certain future time, $E(L_{t_0+\gamma}(G^i_{t_0}))$, when i leaves from v^i_{base} at time t_0, where γ ($\leq M_i$) is a positive integer. Furthermore, i also stores the number of visited nodes, $N_{vis}(t)$, and the amount of vacuumed dirt, $N_d(t)$, at t ($> t_0$) during the current cleaning cycle, which started from t_0. Then, the agent will try to expand its current RA, V^i_t, if the following conditions are satisfied.

$$N_{vis}(t) \geq R_1 \cdot |V^i_t| \tag{4.9}$$

$$N_d(t) \geq R_2 \cdot E(L_{t_0+\gamma}(G^i_{t_0})), \tag{4.10}$$

where $0 \leq R_1, R_2 \leq 1$, and $0 \leq \gamma \leq M_i$ are the parameters used by agents to determine whether or not they have mostly cleaned the current RA. The parameter γ is introduced to specify the expected future amount of dirt in the RA due to the continuous accumulation of dirt while the agents move around. Of course, agents may compute $E(L_t(G^i_t))$ every time and can use it in Condition (4.10) instead of $E(L_{t_0+\gamma}(G^i_{t_0}))$. However, we use $E(L_{t_0+\gamma}(G^i_{t_0}))$ in the following experiments to avoid the frequent calculations of the expected value. These conditions indirectly reflect both the capabilities of the agent's hardware and the quality/performance of the exploration algorithms. Agents with a simple algorithm cannot effectively move around the area (for example, the agents may visit the same nodes many times and/or may skip some nodes). Agents that can move more quickly have a sophisticated exploration algorithm, or have a large-capacity battery can more easily satisfy two conditions and thus are likely to expand their RAs.

Note that a larger R_1 and R_2 make agents more conservative about expanding their RAs. There is a trade-off between conservativeness and eagerness: eager agents with a small R_1 and R_2 try to expand their RAs even if their RAs are not clean enough while conservative agents will avoid expanding their RAs even if they are able to do so, and the adjacent agents have smaller expansion powers. We will discuss this in Section 4.4.2.6.

4.3.1.3 Expansion Strategy

When Conditions (4.9) and (4.10) are fulfilled, agents realize that they can perform their tasks in a larger area. Therefore, they will start an *area expansion trial* (AET), in which they try to cover other nodes that are not covered by other agents or are in the RAs of busier agents. In the AET, we have taken into account two factors. The first factor is the distances from their bases because visiting only far nodes may reduce the performance of both agents and the whole system. The second one is the frequent failures of expansion in a certain direction operated by unbusy agents in which we attempt to avoid.

When agent i finds that Conditions (4.9) and (4.10) are satisfied at time t during its cleaning cycle, an AET process begins comprising two parts, where the first part involves the case that i identifies the nodes to be included in its RA using the *expansion strategy*; the latter part involves the negotiation of i with neighbor agents to determine which agent should be responsible for the identified nodes with the assumption that part of I^i is in the RAs of the neighbor agents.

The expansion strategy tries to include the boundary nodes closer to the charging base. Agent i starts with defining its current RA boundary, which is denoted by $B(V_t^i) \subset V$. For instance, Fig. 4.1 represents the environment G in a grid graph. The set of white color nodes with bold lines is denoted by V_t^i, and the set of the light blue and orange colored nodes is called the boundary, $B(V_t^i)$. i chooses the set of k_{inc} nodes, $I_{inc}^i (\subset B(V_t^i))$,

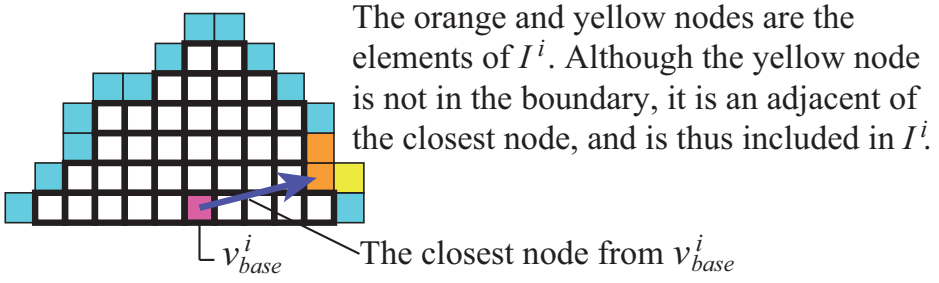

FIGURE 4.1: Expansion strategy (the nearest boundary expansion)

The squares with bold lines denote the current RA. The light blue and orange squares are boundary of sub-area, while the pink one is the base node.

that are not in I^i_{avoid} and are considered as the nearest nodes from v^i_{base}, for positive integer k_{inc}. After that, it will define I^i as the nodes in I^i_{inc} and their neighboring (north, south, east, and west) nodes that are not in both V^i_t and in I^i_{avoid}. In Fig. 4.1, for instance, the orange and yellow nodes express I^i when $k_{inc} = 1$ and $I^i_{avoid} = \emptyset$. If $I^i = \emptyset$, the AET process ends, and nodes are not added to i's RA.

If one of the adjacent agents can afford to clean a larger RA, an attempt to take nodes from its RA may fail. To get rid of frequent failures of the AET, i stores the nodes that it failed to take into I^i_{avoid}. At the same time, it does not choose those nodes as elements of I^i in the next k_{avoid} times of AET. Note that I^i_{avoid} is initially set to $\emptyset$, and k_{avoid} is a positive integer.

4.3.1.4 Negotiation for Expanding Responsible Areas

The negotiation process begins after agent i identifies I^i to determine which nodes in I^i should be included in its RA according to the following steps:

(1) The revision of the RA:
V^i_t is set to $V^i_{t-1} \cup I^i$.

(2) The sending of request message for area expansion:
i reports I^i based on its current expansion power $\xi = \xi(i,t)$.

(3) The acceptance/rejection of area expansion request:
Assume that a request message for area expansion has been sent from agent i to j at time t. If $V^j_t \cap I^i = \emptyset$, j does nothing. Otherwise, j compares j's expansion power, $\xi(j,t)$, with ξ which yields two possible conditions:

(3.1) First condition: if $\xi(j,t) \geq \xi$, agent j sends a rejection message with $V^j_t \cap I^i$ and $\xi(j,t)$ to i.

(3.2) Second condition: if $\xi(j,t) < \xi$, agent j sends an acceptance message with $V^j_t \cap I^i$ and $\xi(j,t)$ to i. Then, j revises its RA to $V^j_t = V^j_t \setminus I^i$.

(4) Expansion of responsible area:
In the case that i receives a rejection message from j, the nodes will

then be excluded from V_t^i and stored the information about those nodes into I_{avoid}^i with j's expansion power. They will not be included in I^i in the next k_{avoid} times of AET. This process will help the agents avoid frequent failures.

Agent i continues performing its cleaning task in the current RA during the above message exchanges. AET is supposed to be invoked only once per cleaning cycle even if i has enough battery to continue so as to avoid excess expansion. Yet, of course, we can omit this restriction.

4.3.2 Identifying the Location of Obstacles

We assume that agent i can detect obstacles using sensors (e.g., touch/sonar/infrared sensor, proximity sensor and camera) and in this paper, i can detect a node of obstacle by hitting it using touch sensor which is the simplest way. Agent i starts moving from its charging base v_{base}^i along the path generated by an exploration algorithm. It then memorizes the nodes that it cannot move which is defined as *block node* O^i, whose initial value is the empty set. Then, when i hits a node of an obstacle during the cleaning process, it adds them into O^i. Furthermore, if the elements in O^i surround other nodes, these are the part of the obstacles. Thus, they are added into O^i. This enables i to recognize which nodes are the parts of obstacle. After their RAs changed or O^i was revisited, agents recalculated the shortest distance between nodes in the RAs when they arrive at their charging bases.

4.3.3 Learning of Dirt Accumulation Probabilities

To identify which nodes are easy to become dirty in the RAs, agent i learns p_v^i for $\forall v \in V_t^i$, which are the estimated values of the DAPs of V_t^i. First, when node v is added in $V_{t_v}^i$ at time t_v, i initializes as $p_v^i = 0$ and the last time when i visited v, $t_{LV}^i(v)$, is set to t_v.

Right after i has vacuumed up dirt at node v at time t, i calculates the interval, $I_t^i(v)$, between the current and the last time visited v:

$$I_t^i(v) = t - t_{LV}^i(v). \tag{4.11}$$

Then, the DAP of v is estimated by $L_t(v)/I_t^i(v)$. However, the reliability of such an estimated value depends on the length of interval, $I_t^i(v)$. Thus, we introduce the variable learning rate, $\alpha(x)$, which weighs the obtained probability according to the length of the interval, and p_v^i is updated as:

$$p_v^i = (1 - \alpha(I_t^i(v)))p_v^i + \alpha(I_t^i(v))\frac{L_t(v)}{I_t^i(v)}. \tag{4.12}$$

Then $t_{LV}^i(v)$, is set to t. The learning rate function $0 < \alpha(x) < 1$ in Eq. (4.12) is monotonically increasing and is defined as the linear function with the upper bound:

$$\alpha(x) = \max(\delta x, \alpha_{max}) \tag{4.13}$$

in the experiments below, where $0 < \delta \ll 1$ is the gradient of the learning rate, and $0 < \alpha_{max}$ is the upper bound.

4.4 Experimental Evaluation

4.4.1 Experimental Setting

We evaluated the ePBP method by clarifying its performance and features in a variety of situations using two environments for the simulation, as illustrated in Fig. 4.2. G, which is the cleaning area, is a 51×51 grid. Node v is expressed by (x, y), where $-25 \leq x, y \leq 25$. Four agents $A = \{a_1, a_2, a_3, a_4\}$ move around G starting from their charging bases v_{base}^i $(i = 1, 2, 3, 4)$. The set of obstacles, O, is empty if nothing is stated.

The DAPs for all nodes are shown in the figure, where parameters p_l, p_m, and p_h are described as:

$$p_l = 2 \cdot 10^{-6}, p_m = 2 \cdot 10^{-5}, p_h = 2 \cdot 10^{-4}. \tag{4.14}$$

The dirt in the first environment (Env. 1) uniformly accumulates, while the second environment (Env. 2) consists of areas where dirt more easily accumulate. These areas in Env. 2 are represented by the square regions, p_h and p_m, where the red region is specified by $(-20, -20)$ and $(-10, -10)$, and the blue region by $(5, -5)$ and $(15, 5)$, so the sizes of these regions are

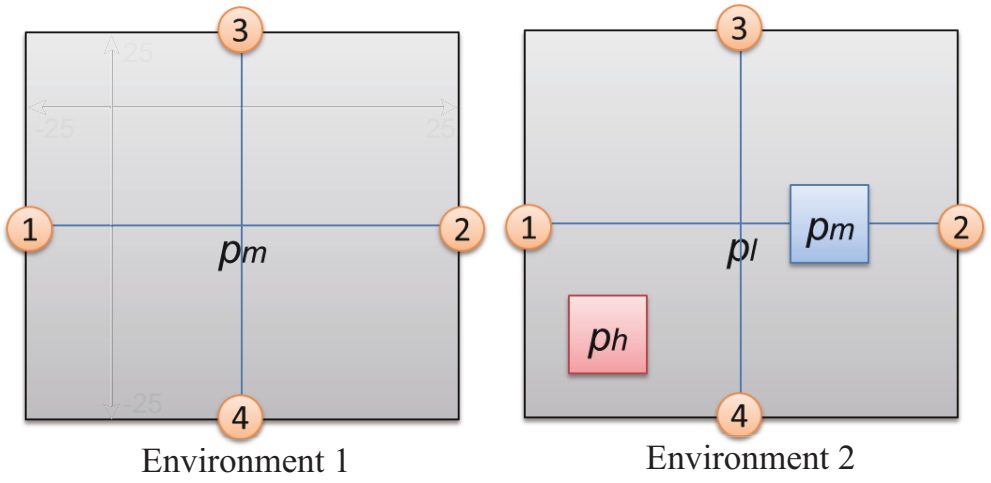

FIGURE 4.2: Experimental environments

121. The numbers with circle represent the charging bases' locations, e.g., the charging base of a_1 is at $(-25, 0)$. Furthermore, the subarea whose DAP is p_h in Env. 2 is considered to be an *easy-to-dirty* subarea. Note that since the DAP in Env. 1 is p_m, so Env. 1 is dirtier than Env. 2.

We assume that all agents have the same batteries, and the specific battery configurations and their values are shown in Table 4.1. We defined these values in accordance with the specifications of an actual robot cleaner[1]. In addition, we also includes the value of parameters for selecting and controlling AET and the parameters used in the learning of DAPs in Table 4.2 and Table 4.3. We then stored, every 3600 ticks (which is the maximal cleaning cycle) up to 1,000,000 ticks, the sum of the amount of remaining dirt, D, the expansion powers $\xi(a_i, t)$ calculated when the agents returned to their base, and the sizes of the RAs, $|V_t^i|$. The experimental results given below are the average values of 100 trials. These results are compared with those of a conventional partitioning method [27], whose given area is divided by the agents into equal-size subareas based on the comparison of the current sizes of their RAs. We call it the *balloon method* [27] hereafter.

[1]In our experiments, one tick is about 4 seconds, the velocity is 0.25 m/s, the maximum operation time is 1 hour, and each agent's battery takes 3 hours at maximum to fully charge.

TABLE 4.1: Battery configurations

Parameters	**Value**
Maximal battery capacity: B^i_{max}	900 ticks
Battery consumption per tick: B^i_{drain}	1 hour
Time to charge: k^i_{charge}	3 hours
Maximum running time: M_i	900 ticks
Time for fully charge: $T^i_{charge}(t)$	2700 ticks if the battery is empty
Maximum length of a cleaning cycle (for all agents)	3600 ticks

TABLE 4.2: Parameters used in AET

Parameters	**Value**
R_1	0.7
R_2	0.7
γ	300 ($= M_i/3$)
k_{inc}	15
k_{avoid}	17

TABLE 4.3: Parameters used in the learning of DAPs

Parameters	**Value**
δ	0.0001
α_{max}	0.5

We conducted four experiments. In the first experiment (Exp. 1), we compared cleaning performance and examined how the environments were divided in accordance with the environmental characteristics. The second experiment (Exp. 2) investigated how the ePBP could reflect the difference in algorithms of exploration. In the third experiment (Exp. 3), we introduced the agents with the enhanced battery to know how hardware differences affected the RA partitioning. Finally, we added a number of obstacles into the environments to investigate how the ePBP method decided the RAs by reflecting the obstacles, especially a intricately-shaped obstacle, in the fourth experiment (Exp. 4).

4.4.2 Experimental Results

4.4.2.1 Algorithms for Exploration in Experiments

Agents move around the RAs by using certain exploration algorithms and to verify that the proposed PBP method can determine the RAs by taking into account the differences in algorithm performance. We assume that the agents use one of three exploration algorithms described below. Because the focus in the experiments is on area partitioning for division of labor, these algorithms are quite simple and non-intelligent; improvement of exploration algorithm out of scope, but agents can use more effective algorithms in our framework.

With the *random exploration* (RE) algorithm, agent i randomly selects target node v from V_t^i and then moves to v along the shortest path from the current node. After reaching the node, i randomly selects another node, i.e., it iterates this select-and-move action.

With a simple depth-first search, *directed depth-first exploration* (DDFE) algorithm, i chooses the first targeted node, $v \in V_t^i$, whose expected amount of accumulated dirt $E(L_t(v))$ is the largest when it leaves v_{base}^i, moves to it along the shortest path, and pushes the node on top of its stack. After that, it randomly selects one of the adjacent nodes excluding a previously visited one, moves to it, and pushes the node on top of its stack. This process is iterated as long as i can select an unvisited node. Then, if i cannot select it, i moves back to the previous node by popping the top node from its stack and backtracking one step. It again tries to select another unvisited node. Finally, i will return to it base node v_{base}^i after it returns to the first chosen node. Although [44] used the *(random) depth-first exploration* (DFE) algorithm that is also a depth-first search simpler than DDFE, we did not use it here. DDFE relies on the learned DAPs , so it is better to see the effect of the DAP learning on the performance.

The DDFE algorithm is better than the RE one since an agent using RE may visit the same nodes many times but one using DFE does not visit the same node in a cleaning cycle except when backtracking. Note that agents using these algorithms move to only safe nodes, as we mentioned

in Section 4.2.1. If they find that the next node is not safe, they directly return to their base nodes via shortest paths.

4.4.2.2 Performance of Cleaning and Sizes of RAs

For the purpose of Exp. 1, we compare the sum of the amount of remaining dirt, D, in two environments. We assumed that all agents used the DDFE exploring algorithm. We also examined the PBP method in Exp. 1 to investigate the differences in performance between the PBP (the DAPs were given) and the ePBP (the DAPs were learned) methods. The results are plotted in Fig. 4.3. The average values of $D = D_{t_s,t_e}$ observed between $t_s = 800,000$ and $t_e = 1,000,000$ in Env. 1 and Env. 2 and the improvement ratios of the ePBP method to the conventional method are also listed in

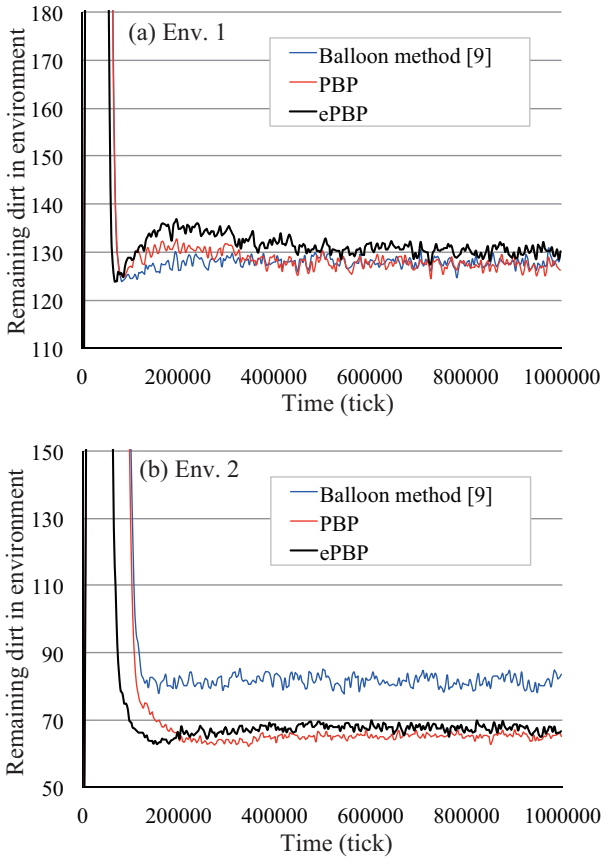

FIGURE 4.3: Amount of remaining dirt, D, using PBP and ePBP in Exp. 1

Table 4.4. For Env. 1, it is reasonable that the area divisions are equal in size because the DAPs $\{p_v\}_{v \in V}$ are constant. Hence, the differences between the PBP, ePBP, and conventional methods were small although ePBP exhibited slightly lower performances (the improvement ratio was -1.80% in Table 4.4). In Env. 2, the ePBP and PBP methods resulted in a much smaller D than the conventional method, and the improvement ratio was 17.40% (Table 4.4), because the area is partitioned based on the environmental characteristics. We can also observe that the ePBP and PBP exhibited the almost identical performance in both environments (Fig. 4.3) although agents with the ePBP were not given the values of DAPs.

We investigated how the RAs expanded and were partitioned depending on the PBP and ePBP methods over time in Env. 2; the results are plotted in Fig. 4.4. Note that the results for Env. 1 are omitted because Env. 1 is uniform, so they partitioned the equal-size RAs. However, Env. 2 has two easy-to-dirty subareas, so the equal-size partitioning is inappropriate. First, Fig. 4.4 (a) and (b) indicates that the sizes of RAs of a_i, $|V_t^i|$, were quite indifferent between the PBP and ePBP methods in Env. 2. With both methods, agents a_1 and a_4 had their bases located near the easy-to-dirty subarea; thus the sizes of their RAs are relatively smaller than those of the others (note that p_h is 10 times larger

Table 4.4: Average values of remaining dirt between 800,000 and 1,000,000 ticks.

		Conventional Method	ePBP Method	Improvement Ratio (%)
Exp. 1	D_{t_s,t_e} in Env. 1	127.9	130.2	-1.80
	D_{t_s,t_e} in Env. 2	81.6	67.4	17.40
Exp. 2	D_{t_s,t_e} in Env. 1	171.0	165.7	3.10
	D_{t_s,t_e} in Env. 2	106.4	81.1	23.78
Exp. 3	D_{t_s,t_e} in Env. 1	92.8	85.6	7.76
	D_{t_s,t_e} in Env. 2	58.2	44.0	24.40
Exp. 4	D_{t_s,t_e} in Env. 1	131.4	134.5	-2.35
	D_{t_s,t_e} in Env. 2	85.6	68.3	20.12

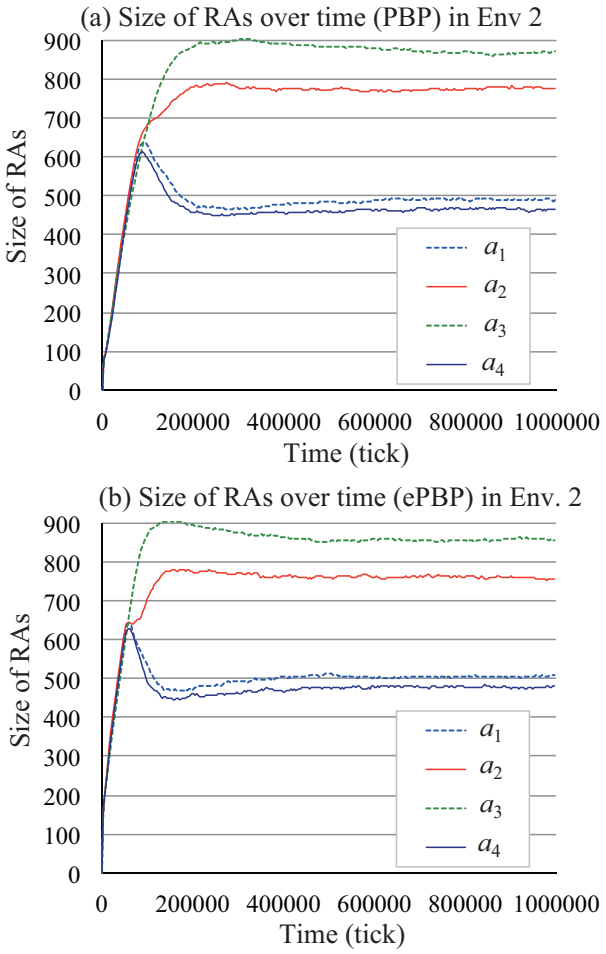

FIGURE 4.4: Sizes of RAs, $|V_t^i|$, in Exp. 1

than p_m). The RA of a_3 was the largest because the area near its charging base rarely got dirty.

Figures 4.3 and 4.4 indicate that the convergences were slightly faster when agents adopted the ePBP method. Because they initially believed that the environment was uniform and had no easy-to-dirty subareas, they tried to extend their RAs to proactively clean the wider areas.

Figure 4.5 plotted the amount of remaining dirt, D^i, in Env. 2 when agents adopted the conventional or ePBP method. It shows that the differences in D^i were quite smaller in the ePBP method than those in the conventional method; this is the result of better partitioning of RAs for balanced work by taking into account the characteristics of Env. 2. Note that Fig. 4.5 (b) indicates that the values of D^i did not become identical.

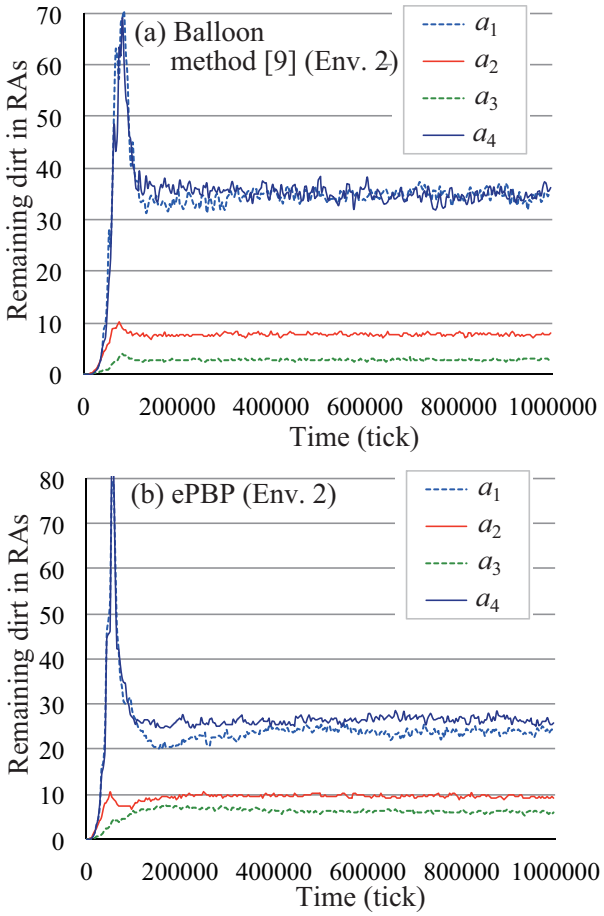

FIGURE 4.5: Remaining dirt in $|V_t^i|$ in Exp. 1

The main reason is that when they charged (maximally, 2700 ticks), the amount of dirt increased, especially, in the nodes whose DAP were high. Actually, in Env. 1, D^i converged to an identical value in all experiments below. We will show this fact in Exp. 2 in the next section because its experimental setting was more diverse than that of Exp. 1.

4.4.2.3 Effect of Different Exploration Algorithms

In Exp. 2, we gave agents two different exploration algorithms, RE and DDFE, described in Section 4.4.2.1; agents a_1 and a_2 used RE and a_3 and a_4 used DDFE. The subarea near the charging bases for a_1 and a_4 would be the dirtiest although RE is less effective than DDFE. In realistic situations, the agents using a better algorithm should be allocated to the dirtier areas.

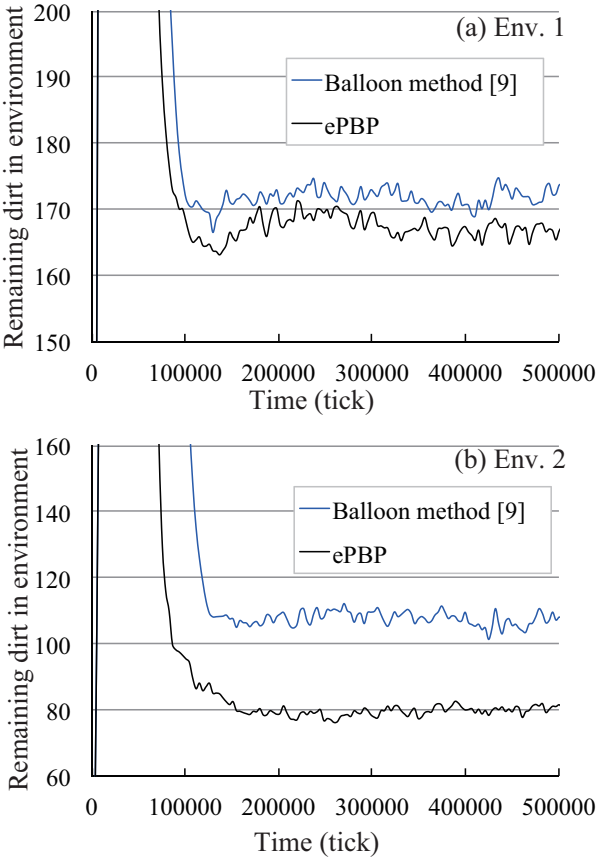

FIGURE 4.6: Amount of remaining dirt, D, in Exp. 2

We did not do this because we wanted to clarify the effect of the differences in algorithms and environments on performance and RA partitioning.

Figure 4.6 is the set of graphs showing the amount of remaining dirt, D, in Envs. 1 and 2 over time. We also listed the average value of D between 800,000 and 1,000,000 ticks in Table 4.4. These data indicate that the ePBP method could clean more effectively, especially in Env. 2 like Exp. 1, than the conventional method.

We also plotted the sizes of RAs of a_i with the ePBP method in Envs. 1 and 2 in Fig. 4.7. In Env. 1, agents with the ePBP method autonomously divided their RAs in accordance with their exploration algorithms. In Env. 2, by adding the easy-to-dirty subareas, the tendency was more notable; for example, a_3 and a_4 used the DDFE, but a_3 had no dirty subareas near its charging base, so the size of its RA became 1000 nodes approximately. In contrast, a_4 had the small RA that was smaller

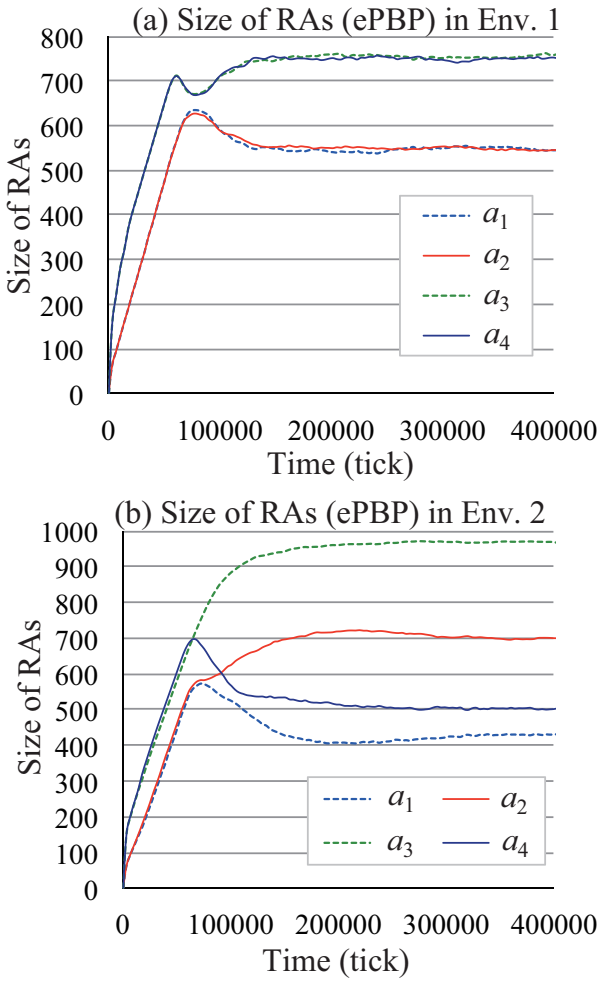

FIGURE 4.7: Sizes of RAs, $|V_t^i|$, in Exp. 2

than a_2's RA. The graphs in Fig. 4.8 show the amount of remaining dirt in RAs, D^i (for $i = 1, 2, 3, 4$), when agents adopted the conventional method (Fig. 4.8 (a) and (c)) and the ePBP method (Fig. 4.8 (d) and (d)). We can find that the proposed ePBP method could clean the RAs more evenly in both environments. Note that in Env. 2 (Fig. 4.8 (d)), the values of D^1 and D^4 were relatively larger although agents tried to divide the RAs for balanced work. This reason is identical to the case in Exp. 1; the amount of dirt increased in the nodes whose DAP were high when they charged. Note again that p_h is 10 times larger than p_m.

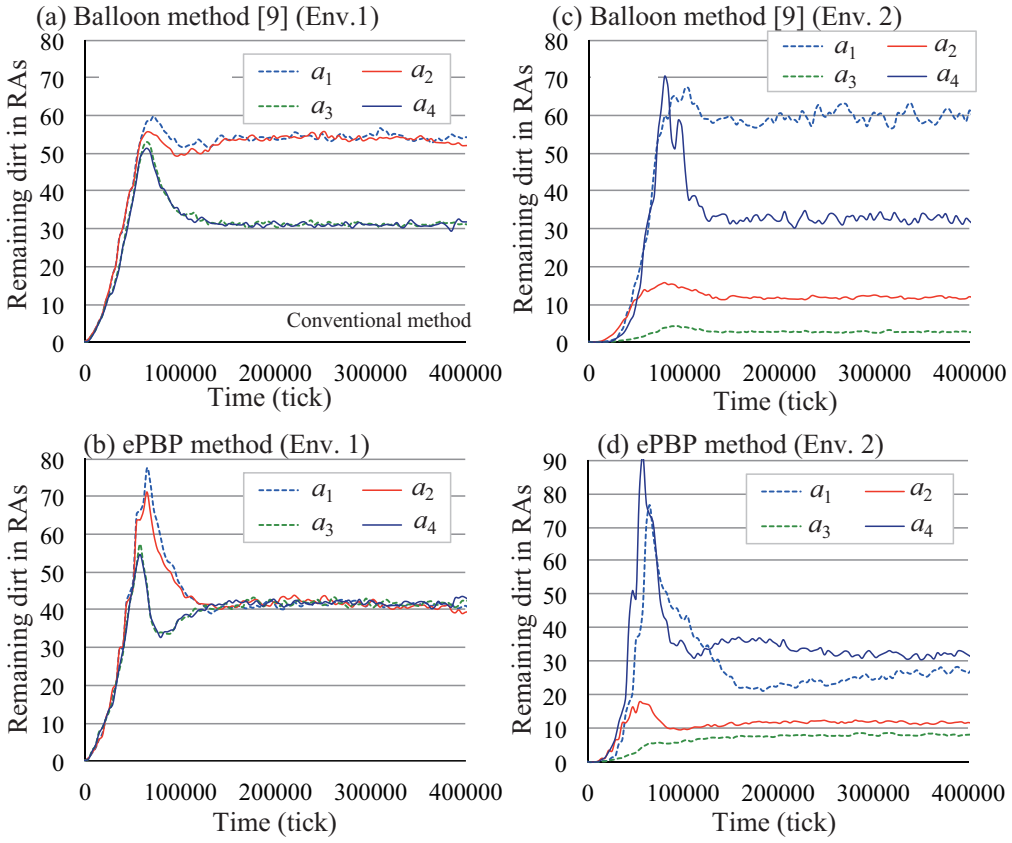

FIGURE 4.8: Remaining dirt in $|V_t^i|$ in Exp. 2

4.4.2.4 Effect of Hardware Difference

We conducted Exp. 3 to see the effect of hardware difference, more specifically different capacities of batteries, on the sizes of RAs and on the performance of cleaning. We assumed that a_1 and a_2 had the same battery in the previous experiments, but a_3 and a_4 had a better (long-life) battery that is specified as $B^3_{max} = B^4_{max} = 1800$ (and other battery specifications k^3_{charge}, k^4_{charge}, B^3_{drain}, and B^4_{drain} are identical to other's batteries). Other experimental setting was identical to that of Exp. 1.

Figure 4.9 (a) and (b) shows the amount of dirt remaining in the environments over time. The average value of D between 800,000 and 1,000,000 ticks is also listed in Table 4.4. They indicate that the ePBP method could outperform the conventional method due to better allocations of RAs as shown in Fig. 4.10 (a) and (b), which shows the sizes of RAs allocated to agents, a_i in Envs. 1 and 2. Figure 4.10 (a) indicates that a_3

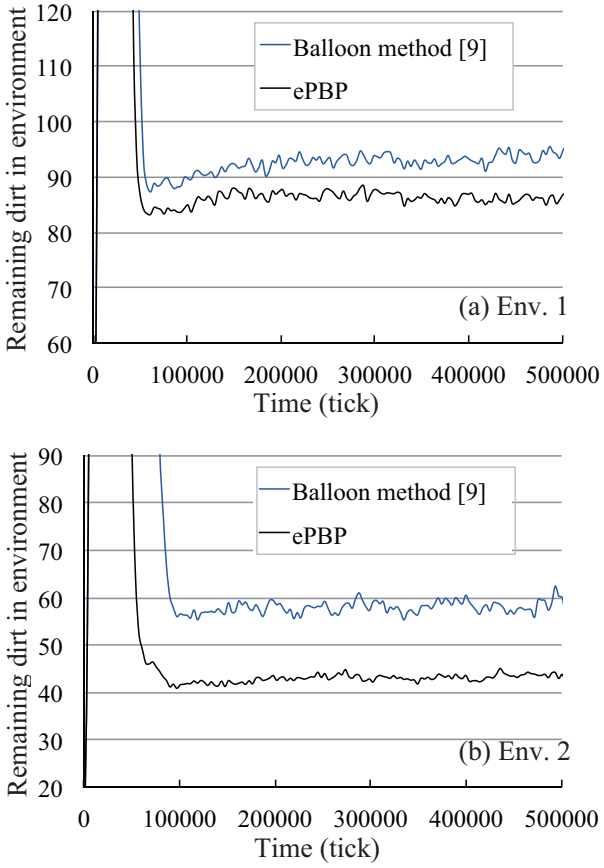

FIGURE 4.9: Amount of remaining dirt, D, using conventional method and ePBP in Exp. 3

and a_4 had larger sizes of RAs than those of a_1 and a_2 in accordance with their battery capacity specifications. In Env. 2, Figure 4.10 (b) exhibits more interesting curves: until 5,000 ticks, the sizes of RAs were similar to those in Env. 1. After that, because the agents began to include the dirtier subareas in their RAs and to learn the DAPs, agents changed the sizes of their RAs by reflecting the environment and the battery capacities shortly. Thus, the RA of a_4, for example, became smaller although it has better battery, and conversely, the size of a_2's RA became larger. Note again that p_h is 10 times larger than p_m.

Figure 4.11 (a) and (b) shows the amount of remaining dirt in RAs in Env. 2. We omitted the graphs in Env. 1 but found that the ePBP method could keep clean uniformly in Env. 1 due to the balanced work allocations. It also made the difference in remaining dirt between RAs smaller in Env. 2

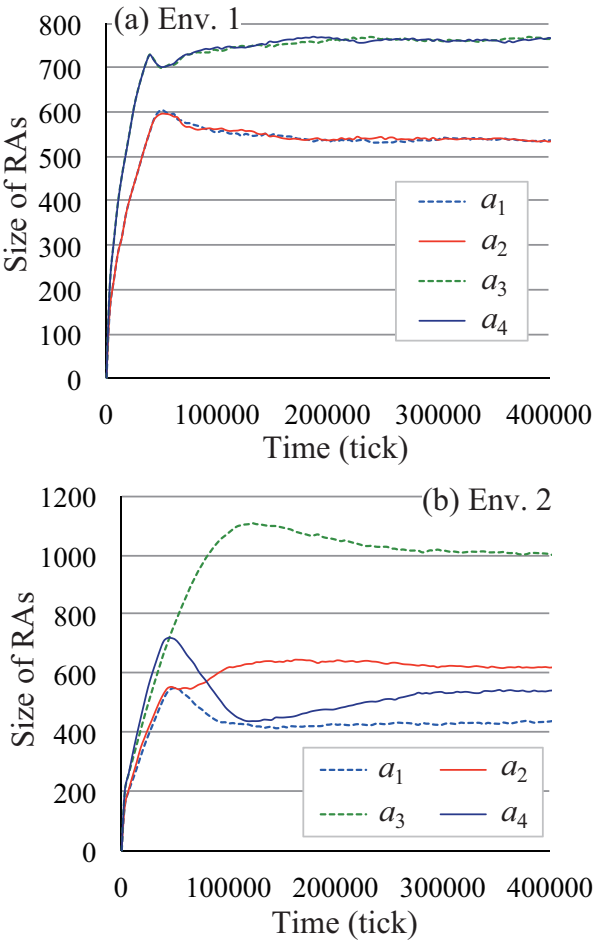

FIGURE 4.10: Sizes of RAs, $|V_t^i|$, in Exp. 3

(Fig. 4.11 (a) and (b)); however, there are still differences between them because the difference was caused by the increase of dirt in the easy-to-dirty subarea during battery charge.

4.4.2.5 Balanced RA Allocations with Obstacles

In Exp. 4, we investigated how existence of obstacles and their shapes affected their sizes of RAs. For this purpose, we put three obstacles into Envs. 1 and 2 with different shapes, including square, rectangular and E-shape, as shown in Fig. 4.12. These environments are referred to as Env. 3 and Env. 4, respectively. We add the E-shaped obstacle since it is slightly complicated and some extra time is required to clean its neighbors. The square obstacle is specified by $(-18, -3)$ and $(-13, 2)$, while the rectangular

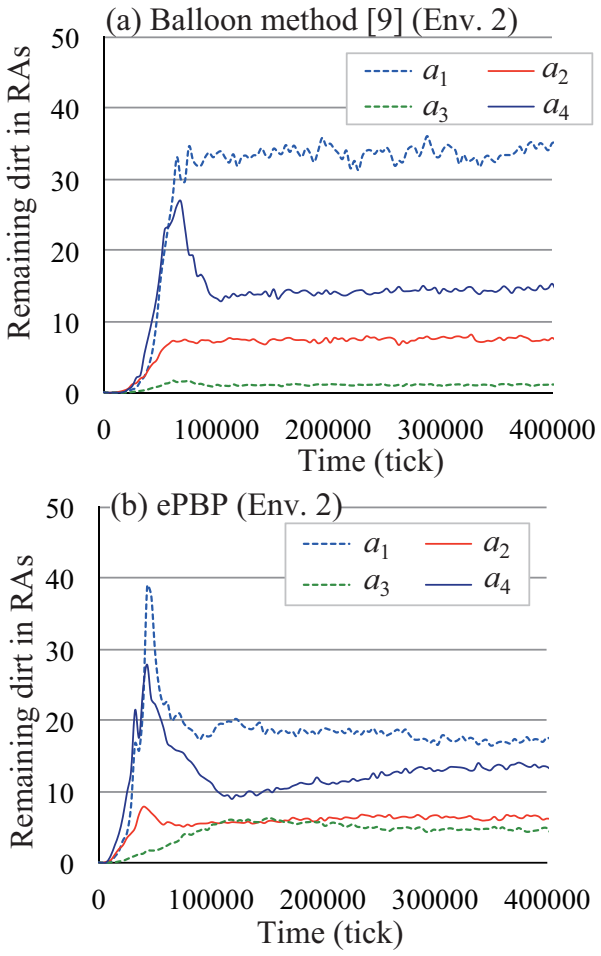

FIGURE 4.11: Remaining dirt in $|V_t^i|$ in Exp. 3

is specified by $(13, -6)$ and $(18, 3)$. The size and location of the E-shape obstacle is shown in Fig. 4.12. Note that the rectangular obstacle partly overlapped the dirtier subarea whose DAP is p_m.

When a number of obstacles exist in the environment, we could observe the slightly different phenomenon. Figure 4.13 presents how remaining dirt, D, varied overtime (until 1,000,000 ticks) in Envs. 3 and 4. We also listed the improvement ratios of D between 800,000 and 1,000,000 ticks in Table 4.4. Figure 4.13 and Table 4.4 indicate that the ePBP method left slightly more dirt in Env. 3 than the conventional method as in Exp. 1, although the ePBP outperformed the conventional method in Env. 4. Because Env. 3 is uniform except the three obstacles which hindered for the learning of the DAPs, the ePBP could not learn

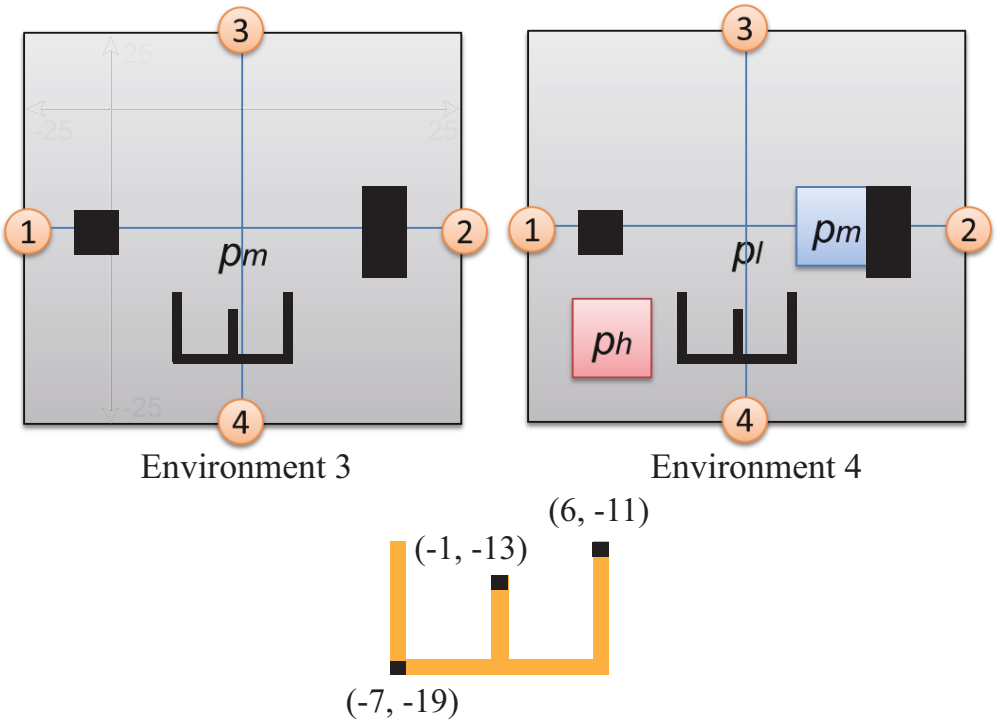

FIGURE 4.12: Experimental environments

the DAP values efficiently. Figure 4.13 also shows that the values of D almost converged around 5,000 ticks. However, if we look at Fig. 4.13 (a) and (b) more carefully, the values of D decreased very slowly after that.

Figure 4.14 represents the size of RAs of agent a_i in both Envs. 3 and 4 using our proposed method. Note that the sizes of RAs excluded the nodes occupied by obstacles. Figure 4.14 (a) indicates that agents autonomously divided the areas on the basis of only the existence and the shapes of the obstacles since Env. 3 is uniform. For example, a_4 had the E-shaped obstacle that is more complex than others, and it took more ticks to reach the areas inside the E-shared obstacle. This results in the smaller a_4's RA than others. The RA of a_2 was also smaller because it had the rectangular obstacle which took slightly longer time to reach the nodes in the opposite side of the rectangle from the a_2's base, v^2_{base}. This situation is also similar for a_1 but the obstacle near v^1_{base} was smaller, so the RA of a_1 was relatively larger.

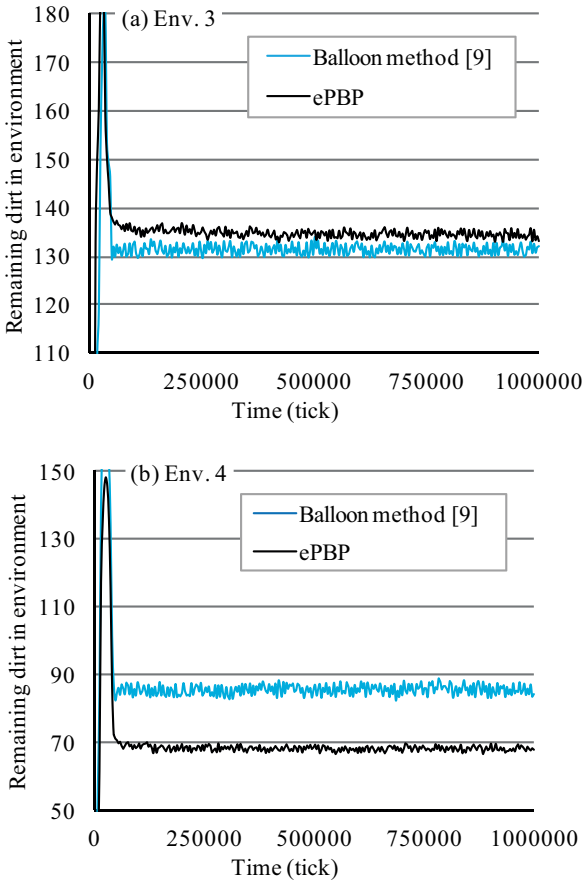

FIGURE 4.13: Amount of remaining dirt, D, in Exp. 4

On the other hand, because Env. 4 has a number of easy-to-dirty subareas, the area partition reflected both the obstacles and the characteristics of the environment. Figure 4.14 (b) indicates that because a_4 had both the E-shape obstacle and the easy-to-dirty subarea near the charging base, its RA was the smallest (about 400). In addition, the RA of a_3 was the largest (around 860), for there was neither easy-to-dirty subarea nor obstacles nearby its charging base. Of course, agents with the conventional method have equal-size RAs, thus the RA including the complex-shaped obstacle and easy-to-dirty region tended to have more remaining pieces of dirt.

Figure 4.15 (a) and (b) represents the amount of dirt left in the RAs in Env. 4 using the conventional and the proposed methods, respectively. Fig. 4.15 (a) indicates that the differences in the amount of remaining dirt

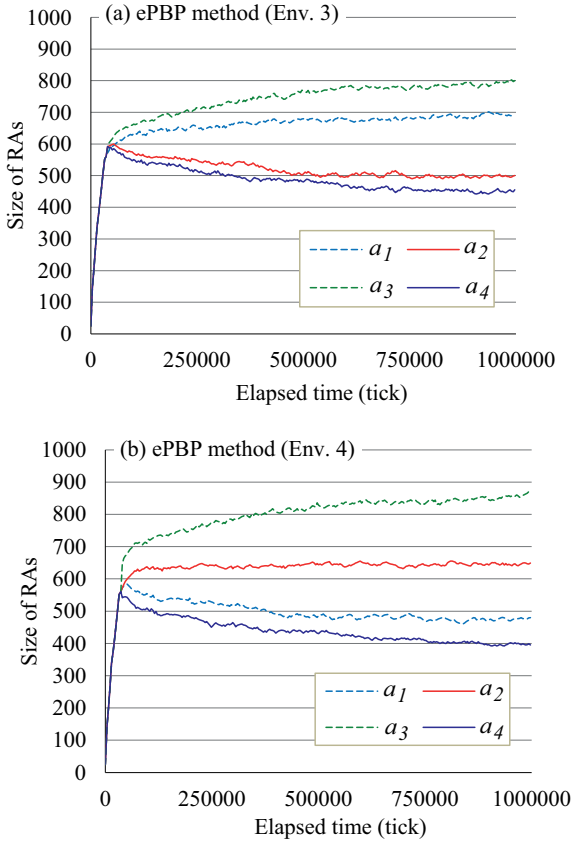

FIGURE 4.14: Sizes of RAs, $|V_t^i|$, in Exp. 4

in RAs, D^i $(i = 1, 2, 3, 4)$, were large but by using the proposed method, we can see from Fig. 4.15 (b) that agents could keep the values of D^i closer. This result shows that our proposed method could vacuum dirt in a more balanced manner.

We can observe two phenomena different from other experiments. First, if we compare the results of a_1 and a_4 in Figs. 4.14 (b) and 4.15 (b), we can see that the size of a_4's RA was smaller but the a_1's RA was dirtier. This indicates that because a_4 had E-shaped obstacle, a_1 cleaned the dirty subarea between v^1_{base} and v^4_{base} more than a_4.

Second, Fig. 4.14 obviously indicates that it took longer time to converge the sizes of RAs. We can consider two reasons for this (see also Fig. 4.4). First, agents required more time to reach and thereby learn the $DAPs$ of the regions in the opposite side of obstacles, especially another

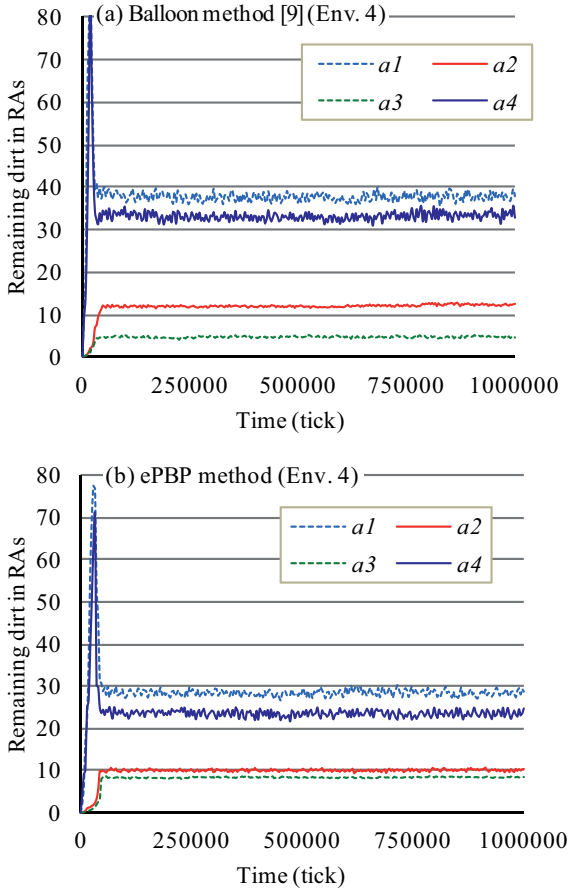

FIGURE 4.15: Remaining dirt in $|V_t^i|$ in Exp. 4

side of the E-shaped obstacle. In addition, the exploring algorithm used in this experiment was too simple to clean effectively such a complex region. Second, the existence of obstacles let the speed of expansion of RAs slower because agents first try to expand them to the nearest nodes. This discussion suggests the limitation of the proposed method; i.e., we have to improve the learning speed, and we will address this issue next time.

How environment is partitioned is shown in Fig. 4.16. Note that we selected this result of partitioning randomly from 100 experimental trials we conducted, and we could see that other partitioning looked similar. We can see in Fig. 4.16 that a_4 had the smallest RA because an E-shape obstacle is next to its charging base, and the dirty area whose probability is p_h is also in its RA. Particularly, this obstacle made the cleaning difficult, and

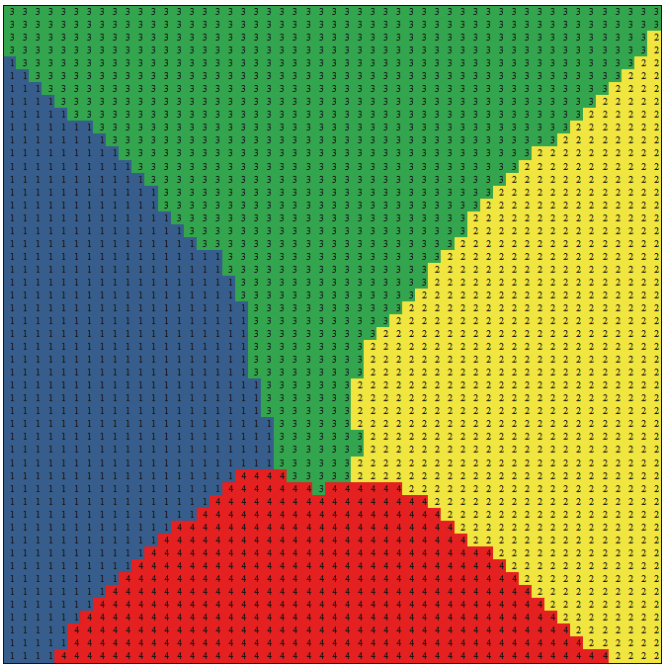

FIGURE 4.16: Shape of RA in Env. 4

a_4 needed to spend longer time. Thus, a_4 decreased its RA. However, the RA of a_3 was the biggest because there is no obstacle nearby its charging base nor the dirty areas.

4.4.2.6 Discussion

From the results of our experiments, we can say that the proposed ePBP method can effectively partition the area in accordance with the differences in the environment and the performances of the agents in a cooperative cleaning task. The agents that: (a) use more efficient algorithms, (2) have high-capacity batteries, and/or (3) are deployed in regions that are relatively simpler and cleaner can handle larger areas, and thus, they try to expand their RAs by acquiring nodes from busier agents. Furthermore, although the ePBP method does not assume the information of dirty areas, i.e., the values of DAPs , it exhibits the performance comparable to the PBP [44]. However, a few things need to be considered.

The first thing to consider is the effect of the parameters used. Parameters R_1, R_2, and γ, which are used in Conditions (4.9) and (4.10) specify the situations in which agents start an AET. If these parameters

are large, agents tend to expand their RAs only after they have sufficiently cleaned their current RAs. That is, they are conservative about expanding their RAs even if the adjacent agents lack the performance needed to clean their areas. If these parameters are small, the agents tend to start an AET even before their current RAs have been sufficiently cleaned, so AETs are started more frequently. This can result in frequent meaningless AETs. Parameter k_{inc} controls the number of nodes acquired in a single AET, and parameter k_{avoid} controls the number of fruitless AETs in which agents try to extend their RA towards the expense of agents with high expansion powers. The trade-off mentioned here is similar to the explore-or-exploit dilemma that occurs with learning algorithms. We think that the learning is needed to decide the values of these parameters: This is left to our future work.

Finally, as shown in Exp. 4, the convergence became slower when the environment had a number of obstacles. When its shape was complex, like the E-shaped obstacle, in particular, agents could rarely reach recessed areas inside the complex obstacle due to a number of reasons, and this resulted in the inefficient learning. First, the exploring algorithm used in our experiments was so simple to explore such recessed areas. Second, more importantly, agents had no information about the DAP and initially assumed that such recessed areas were not so dirty, so there were no motives to move there. For example, if the recessed areas were easy-to-dirty, agents gradually learned it and visited there more often. However, in our experiment, the recessed areas were not easy to be dirty. This is also another issue that we should address in the future.

By using area expansion trial (AET), agents can adaptively expand their RAs. If the room is large, agents can expand their RA rapidly by adjusting the parameters used in the AET strategy. However, we cannot decide the maximum size of the cleaning area because it depends on the specifications and the number of agents. Note that most of the computational cost in our proposed method occurs in the calculation of the expansion power, and is $O(m)$, where m is the size of RA.

Our proposed method could partition the area/environment fairly and effectively by taking into account the characteristics of the environment and the capability of each agent. However, some additional issues such as map generation, path planning, identifying agents' locations, collision/obstacle

avoidance, compensation for imperfect communication and how to identify the appropriate number of agents for efficient cleaning exist for the real applications of cleaning/sweeping domains. In particular, although the appropriate number of agents depends on the agent's specifications, it is important to introduce some mechanism, which contribute both efficiency in the cleaning and energy saving, to decide the appropriate number of the cleaning robots. For example, if the room is very dirty, then the number of agents should be increased. Yet, if some agents are redundant, the number of agents should be reduced by improving their specifications. This issue should be solved and is our next future work.

In addition, when an area is connected with a very narrow path (e.g., the room with a small door/gate whose width is 1, through which only one agent can pass), agents cannot partition the whole area in a balanced manner. This is because the whole space in the room will be covered by only one agent whose base is close to the door, while another agent cannot; if the room is very large, the agent that is responsible for it will have to cover the whole room alone, leading to unbalanced task division between agents. This is one limitation of our work, yet we will extend our method to overcome this issue.

Our work is not restricted to only the cleaning application. We can apply it to other real-world applications such as the security patrolling. Agents in this problem domain must visit/monitor locations in environment at different frequencies. For instance, continuous cleaning and security patrolling agents have to control robots so that they frequently visit regions that easily accumulate dirt and those at high security levels. Thus, the cleaning task is just an example for our experiments described in Section 4.4.

4.5 Summary

We have introduced a decentralized area partitioning method for cleaning and patrolling tasks. This method tries to uniformly keep clean/secure the given environment by allocating areas of responsibility in accordance with the characteristics of the environment and the performance of the exploration algorithms. We first modeled the environment, the agents,

and the problem addressed here. Then, we explained the proposed method in which agents try to expand their responsible areas and negotiate with adjacent agents to decide which agents should clean the identified boundary nodes while they learn what areas are easy to accumulate dirt. Experimental results showed that our proposed method can fairly and effectively divide an area into subareas (responsible areas) by taking into account the efficiency and capability of each agent and the environmental characteristics. Finally, unbalanced tasks are resolved, and the tasks for agents are completely done in a more balanced and efficient manner.

We mainly focused on the cleanliness of floor whose purpose is to minimize the amount of remaining dirt left in the whole environment after each cleaning. We think that energy consumption of the cleaning robot is important, but it has not been considered yet and must be related with the appropriate control of the number of agents and their operating time.

We can consider a number of future work to make our method practical as discussed in Section 4.4.2.6. Although applying our method to a new room relies on other methods to create the map of environment as discussed in Section 3.1, we believe that combining our method and a map creation seems better for actual application. Additionally, we plan to find a way to appropriately control the parameter values to enable more autonomous and intelligent activities and to speed up the convergence.

Chapter 5

Graph-Based Area Partitioning Method for Multi-Agent Patrolling Tasks

5.1 Introduction

Continuing advancement in the field of autonomous mobile robots has been apparent within the last few decades. The patrolling problem with a team of agents particularly has gained much attention. Patrolling refers to the act of continuously moving around and visiting the relevant areas or important points of an environment, with some regularity/at regular intervals, in order to protect, navigate, monitor or supervise it. A group of agents is usually required to perform this task efficiently as multi-robot systems are generally believed to hold a number of advantages over the single-robot ones. The ability of multi-robot system in providing solutions for real-world applications and dealing with task complexities has motivated and made many people prefer developing this system to developing a single-robot system [28].

Multi-agent (multi-robot) patrolling, however, is not limited to patrolling real-world areas, yet they can be found in applications on several domains, such as continuous sweeping, security patrolling, surveillance systems, network security systems and games. In other words, patrolling can be beneficial in any domain characterized by the need of

systematically visiting a set of predefined points [73]. For example, in many cases of real police works, there are services with human such as electronic security services [1]. The benefits of those systems are the cost-effectiveness against labor costs, and because it is monitored by sensors, visual overlook and human error are less likely to occur [86]. However, most of current studies assume that the frequency of visit to each node/location is uniform, yet in the realistic applications, the frequencies of visit differ; for example, in security patrolling, each location has different visitation requirement or risk status according to the required security level.

We divide multi-agent patrolling task into three steps: how to partition the work into a number of sub-tasks, how to allocate the individual sub-task to one of the agents and how to select the visiting sequence for each agent. We call them the *partition*, *allocation* and *sequencing* problems, respectively. In this work, we assume *homogeneous* agents that have the same capability and use the same algorithms. This assumption makes the allocation problem simple, and thus, we only consider the algorithms for partitioning and sequencing. The combination of the partition algorithm and the sequencing algorithm is referred as a *strategy*.

In this work, we will model the problem of patrolling as a problem of visiting vertices in a graph with *visitation requirement* by dividing it into a number of clusters. The visitation requirement of a location (vertex) refers to the number of times or how often a patroller agent is required to visit/patrol it in a certain interval of time. Then, after clustering nodes in this graph, each agent is responsible for patrolling the allocated cluster, and its nodes must be visited to meet the visitation requirement, namely *frequency of visit*[1]. In the partitioning step, we applied k-means based algorithm as a clustering algorithm by modifying its objective function and the initialization of centroids so as to make it fit to our problem. Our goal in this step is to cluster a given graph so that the potential workloads of individual clusters are balanced, which means trying to balance the workload amongst all agents. Moreover, the sequencing step addressed how to select the route (sequence of nodes) for each agent in its allocated

[1]The words *visitation requirement* and *frequency of visit* have the same meaning in this context. For the sake of clarity and specification, we refer to the visitation requirement as the *frequency of visit*.

cluster with a minimized cost. We used the *simulated annealing* (SA) here as an algorithm to find the sequences of nodes because our problem is similar to the multiple traveling salesman problem (mTSP), which is a generalization of the well-known traveling salesman problem (TSP) as mentioned in [10], and SA is often used to find the acceptable solutions due to the fact that SA is considered to be a flexible meta-heuristic method for solving a variety of combinatorial optimization problems. The difference between our problem and mTSP is that in mTSP, a number of cities have to be visited by m-salesman whose objective is to find m tours with minimum total travel, where all the cities must be visited exactly once, while in our problem, all the locations in a patrolled area must be visited to meet the visitation requirement. We believe that our model of partitioning and sequencing with the frequency of visit to each node is more fit to realistic environment.

5.2 Problem Formulation

This study aims at proposing solutions for muti-agent patrolling under visitation requirement constraints, namely *multi-agent frequency-based patrolling problem*[2], by trying to balance the workload amongst all patroller agents and then minimizing the cost for patrolling. First, we formulate our problem in this section.

Let $G = (V, E)$ be a complete graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes, and $E = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is a set of edges. The patrolled area is described as a graph G, where a location $v_i \in V$ is represented by its (x, y) coordinates in the 2D plane, and thus, E contains $\frac{n\times(n-1)}{2}$ edges. In our patrolling problem, a node represents a location to be patrolled/visited, and an edge represents a path between nodes along which agents move. Let $A = \{1, 2, \ldots, m\}$ be a set of agents, and $m = |A|$ denotes the number of agents patrolling graph G, where $m < |V|$.

Each edge in G has its associated cost which is a traveling distance. Because nodes in G are points of $\mathbb{R}^2$, the distance between a pair of nodes is the Euclidean distance between two spatial coordinates $v_i \in V$ and $v_j \in V$

[2]This refers to any patrolling problem by a group of agents that take into account the visitation requirement of each individual location in the real-world environment.

denoted by $\|v_i - v_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where (x_i, y_i) and (x_j, y_j) are the coordinates of nodes v_i and v_j respectively.

In the general multi-agent patrolling problem, a team of m agents patrols an area represented by a complete graph, $G = (V, E)$. Thus, there are n nodes to be patrolled and $|E|$ possible paths for m agents to move.

Definition 5.1. Each node (location) in graph G has its associated visitation requirement, simply called *frequency of visit*. Let $f(v_i) \in \mathbb{Z}^+$ be the frequency of visit to each location in G. Agents have to visit node v_i at least $f(v_i)$ times in a given interval of time.

Definition 5.2. A graph C is a subgraph of a graph G if its vertex set $V(C)$ is a subset of the vertex set $V(G)$, that is $V(C) \subseteq V(G)$, and its edge set $E(C)$ is a subset of the edge set $E(G)$, that is $E(C) \subseteq E(G)$.

Definition 5.3. Let route $s = \langle v_1, v_2, \ldots, v_\ell \rangle$, $\forall v_i \in V$ be a sequence of nodes each agent has to visit each cluster.

Then, the length of route s is defined and denoted by:

$$\ell en(s) = \sum_{i=1}^{\ell-1} \|v_i - v_{i+1}\| \tag{5.1}$$

In the patrolling process, an agent tries to find a route with a minimum length. The *route* is defined as the selected path in a subgraph C, which is allocated to an agent to patrol.

Definition 5.4. Multi-agent frequency-based patrolling problem (MAFPP) is specified by (G, f, A), where $G = (V, E)$ is a graph, $f : V \to \mathbb{Z}^+$ is the frequency of visit, and A is the set of m agents. The goal is to find m (connected) subgraphs, $C_1, \ldots, C_m$, of G and the routes in all subgraphs, such that each agent has to visit/patrol a node in each subgraph based on the real-world visitation requirement of each location in a balanced manner and that the length of route in each subgraph is minimized.

The MAFPP consists of two main steps – graph partitioning and subgraph patrolling. Firstly, we partition a patrolled area represented by a graph G into k disjoint clusters (subgraphs), $C = \{C_1, \ldots, C_m\}$, and then

allocate cluster C_i to agent i. The main goal is to cluster G based on the required frequency of visit, where each node is visited at least $f(v_i)$ times by taking into account the condition from Eq. 5.4, in a balanced manner, such that the expected workload of each cluster is not much different from one another.

Let W_{C_s} be an *expected workload* of each agent in its allocated cluster, which is defined by:

$$W_{C_s} = \sum_{v_i, v_j \in C_s} \frac{f(v_i)\|v_i - v_j\|}{|C_s| - 1}, \tag{5.2}$$

where $|C_s|$ is the number of nodes in each cluster. Intuitively, the expected workload here refers to an estimated amount of work a patroller agent has to do if it generates the shortest (or near-shortest) path, which is the estimated total cost/length agent i has to patrol in its allocated cluster/region, not the actual cost. We used this as a metric to evaluate the clustering performance of our proposed method in Section 5.3. If the value of W_{C_s} for all patroller agents are not much different from one another, we can conclude that the overall workload amongst all agents is considered to be balanced.

After obtaining clusters from the first step, the next goal is to generate a route for each agent to patrol in its allocated cluster based on the required frequency of visit to each node.

For all agents in A, let $O(s, v_i)$ be the number of occurrence of node v_i in route s. Thus, the following condition is satisfied.

$$\begin{cases} O(s, v_i) > 0, & \text{if } v_i \in s \\ O(s, v_i) = 0, & \text{otherwise} \end{cases} \tag{5.3}$$

Then, for $\forall v_i \in C_k$, the route s_k must satisfy the following condition:

$$O(s_k, v_i) \geq f(v_i), \tag{5.4}$$

where s_k is the generated route in C_k for agent k, because clusters $(C_1, \ldots, C_m)$ are disjoint.

Let $S = \{s_1, \ldots, s_m\}$ be a set of routes, and thus m routes must be generated for all m agents to patrol G. Then, the multi-agent frequency-based patrolling problem is to find m routes, such that each node is visited at least $f(v_i)$ times and that the length of total routes is the shortest. Thus,

the objective function, R, is to minimize the sum of all routes, denoted by:

$$\begin{aligned} R(s_1, \ldots, s_m) &= min \sum_{k=1}^{m} \ell en(s_k) \\ \text{subject to: } & \sum_{k=1}^{m} O(s_k, v_i) \geq f(v_i), \forall v_i \in V \end{aligned} \tag{5.5}$$

Because C_k is disjoint and independent, and the shortest route in C_k is generated independently so that it meets the requirement of frequency of visit, the cost $R(s_1, \ldots, s_m)$ in Eq. 5.5 is identical to the sum of the cost of routes, $(s_1, \ldots, s_m)$. Therefore, our goal is to minimize:

$$\begin{aligned} R(s_1, \ldots, s_m) &= \sum_{k=1}^{m} min\ \ell en(s_k) \\ \text{subject to: } & \sum_{k=1}^{m} O(s_k, v_i) \geq f(v_i), \forall v_i \in V \end{aligned} \tag{5.6}$$

5.3 Proposed Method

Our proposed method is divided into two main steps: graph partitioning and sub-graph patrolling. As mentioned in Section 5.2, because we improved the well-known unsupervised traditional k-means clustering algorithm by taking into account the non-uniform visitation requirement for each location, we called our proposed method an *improved frequency-based k-means*, namely IF-k-means.

5.3.1 Graph Partitioning

Clustering refers to the process of partitioning or grouping a given set of patterns into disjoint clusters, $C = \{C_1, C_2, \ldots, C_m\}$. This step describes how agent could cluster a given graph, G, by taking into account the different frequency of visit to each node as well as balancing the workload of each cluster. We implemented k-means based clustering algorithm by modifying its objective function and centroids initialization so as to make it suit our problem. Each data point is interpreted as a node in a complete graph G, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes as

mentioned in Section 5.2. The main goal is to partition V into m disjoint clusters by taking into account the required frequency of visit to each node. We denote $C = \{C_1, C_2, \ldots, C_m\}$ as its set of clusters, and $c = \{c_1, c_2, \ldots, c_m\}$ as a set of corresponding centroids.

Simply speaking, k-means clustering is an algorithm to classify or to group the objects based on attributes/features into m number of groups, where m is a positive integer number. The grouping is done by minimizing the sum of square of distances between data points and the corresponding cluster centroids [60].

The traditional k-means clustering algorithm aims at minimizing the following objective function, which is a squared error function denoted by:

$$J = min \sum_{i=1}^{n} \sum_{s=1}^{m} \sum_{v_i \in C_s} \|v_i - c_s\|^2$$

$$\text{subject to: } C_1 \cup \ldots \cup C_m = C$$
$$C_i \cap C_j = \emptyset, \forall\ 1 \leq i, j \leq m,\ i \neq j,$$

$$\text{where } c_s = \frac{1}{|C_s|} \sum_{v_i \in C_s} v_i\ ,$$

m is the number of clusters, and c_s is the corresponding cluster centroid.

We modified the objective function of the above traditional k-means so as to apply our problem framework with frequency of visit. This method is called IF-k-means, and its objective function is denoted by:

$$Q = min \sum_{i=1}^{n} \sum_{s=1}^{m} \sum_{v_i \in C_s} f(v_i)\|v_i - c_s\|^2$$

$$\text{subject to: } C_1 \cup \ldots \cup C_m = C$$
$$C_i \cap C_j = \emptyset, \forall\ 1 \leq i, j \leq m,\ i \neq j, \tag{5.7}$$

$$\text{where } c_s = \frac{\sum_{v_i \in C_s} v_i \cdot f(v_i)}{\sum_{v_i \in C_s} f(v_i)},$$

$f(v_i)$ is the frequency of visit to node v_i, and $\|v_i - c_s\|$ is the Euclidean distance between v_i and c_s.

The aboved objective function, Q, indicates that we are trying to form clusters which produce the shortest distance (from nodes to the centroid of each cluster), and at the same time we consider the frequency of visit $f(v_i)$ in Q. Moreover, we incorporate $f(v_i)$ in the centroid calculation, c_s, to generate the better centroid placement for a more balanced workload division since centroid should be ideally located near nodes with high frequency of visit to minimize the total distance, and a cluster consisting of more nodes with high frequency of visit tends to be smaller in size.

The traditional k-means method has been shown to be effective in producing good clustering results for many practical applications. Although it is one of the most well-known clustering algorithm and is widely used in various applications, one of its drawbacks is the highly sensitive to the selection of the initial centroids, which means the result of clustering highly depends on the selection of initial centroids. Therefore, proper selection of initial centroids is necessary for a better clustering.

Thus, instead of placing the initial centroids randomly as in the traditional k-means, we place them on the nodes with the highest frequency of visit, $f(v_i)$, because a node with higher frequency of visit should have a shorter distance from its corresponding centroid than the node with lower frequency of visit to make the cluster balanced. By doing so, the required time for generating balanced clusters can also be reduced. Moreover, even if the nodes with high frequency of visit huddle together, we are still able to apply this idea as the program will then relocate the position of centroids accordingly based on the modified centroids function, and the clusters should remain balanced.

The difference between our IF-k-means and the classical k-means is that we incorporate $f(v_i)$ to both objective function and its constraint of the classical k-means in order to make the cluster balanced. Adding $f(v_i)$ to the objective function of the classical k-means makes the distance between node v_i and centroid c_s change causing the different size of clusters based on the visiting frequency to each node. It is also important to incorporate $f(v_i)$ into the calculation of the centroids to generate the weighted centroids function for producing a better centroids location for each cluster.

By implementing our IF-k-means, the clusters having more nodes with high frequency of visit tend to have smaller size comparing to those with lower frequency of visit. At each step of the clustering, the centroids move close to high-frequency nodes after the repeated calculation using our modified centroids function. Thus, without incorporating $f(v_i)$ to both objective function and its constraint, the inefficient clustering would happen due to the inefficient centroids placement.

Let T_{diff} be the *difference in workload* among all agents, where we define T_{diff} as follows:

$$T_{diff} = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j=1}^{m} |W_{C_i} - W_{C_j}|, \ i \neq j \tag{5.8}$$

Then, we define that the workload among all agents is considered to be balanced if it satisfies the following condition:

$$T_{diff} \leq M, \tag{5.9}$$

where $M \in \mathbb{R}^+$ is not so large positive number.

In this partitioning process of our proposed work, we calculated the expected workload of each cluster, W_{C_s}, by using Eq. 5.2. Then, we computed the difference in each workload, T_{diff}, by implementing the formula in Eq. 5.8. The process of our proposed IF-k-means algorithm is described as follows, and the pseudocode of how the algorithm works is illustrated in Algorithm 5.3.1.

(1) Sort all nodes in a descending order based on their frequencies of visit, and then add them into an array H.

(2) Randomly select k nodes from H consecutively, where k is the number of cluster.

(3) Place k initial centroids on the selected k nodes in G.

(4) Assign each node to the cluster that has the closest centroid, and then recalculate the centroids.

(5) Repeat step (4) until the centroids no longer move.

(6) Calculate the expected workload W_{C_s} and the difference in workload T_{diff} for each cluster using Eq. 5.2 and Eq. 5.8, respectively.

(6.1) If the value of T_{diff} satisfies the condition from Eq. 5.9 where $T_{diff} \leq M$, the clusters are accepted.

(6.2) Otherwise, go to step (2) again.

Algorithm 5.3.1: Pseudocode for improved frequency-based k-means (IF-k-means)

Input : $G = (V, E)$ and $f(v_i)$
k (number of clusters), where $k = |A|$
Output: $C = \{C_1, C_2, \ldots, C_k\}$

```
1: Sort V in a descending order based on the f(v_i) of each node
2: Add them into an array H
3: time = 1
4: Select k nodes from H[k(time − 1) + 1] to H[time ∗ k]
5: Place k initial centroids on selected k nodes in G
6: repeat
7:     Assign each node to the cluster having the closest
        centroid
8:     Recalculate(centroids)
9: until centroids no longer move
10: foreach cluster do
11:     Calculate expected workload, W_{C_s} using Eq. 5.2
12:     Calculate difference in workload, T_diff using Eq. 5.8
13:     if T_diff satisfies condition from Eq. 5.9 then
14:         Accept(clusters)
15:     else
16:         time = time + 1
17:         Go to step(4)
18:     end
19: end
```

Algorithm 5.3.2: Pseudocode for constructing an initial solution in SA

Output: $S_0 = \{v_{k_1}, v_{k_2}, \ldots, v_{k_L}\}$ based on condition (5.4), such that $k_i \neq k_{i+1}$

// function **distMatrix** return Euclidean distance between two nodes.
// k_i is the index of node v_i in cluster C_k.
// $O(S_0,$ *curNode*$)$ is an occurence of new *curNode* in S_0.

$S_0 = \emptyset$
Select a current node, *curNode*, randomly from V
Add *curNode* into S_0
while *(S_0 is not filled up)* and $(V \neq \emptyset)$ **do**
    Find the shortest distance from *curNode* to another node in V:
    $shortestDist = min($**distMatrix**$[curNode][j]$ for j in $V)$
    curNode = **distMatrix**[*curNode*].index($shortestDist$)
    Add new *curNode* to S_0
    if *($k_i \neq k_{i+1}$* is not satisfied) **then**
        Regenerate new *curNode*
    end
    if $f(v_i) \leq O(S_0,$ *curNode*$) \leq 2.f(v_i)$ **then**
        Remove new *curNode* from V
    end
end
return S_0

5.3.2 Sub-graph Patrolling

This step presents how agent selected the best route for patrolling in its allocated sub-region with the shortest length by taking into account the required frequency of visit to each location. The goal of this step aims at finding the shortest route for each patroller agent in its allocated cluster with a semi-optimal solution. Because our multi-agent frequency-based patrolling problem is considered to be one of the combinatorial optimization problems and our main purpose is to partition a given area so as to balance

the workload amongst all patroller agents, the optimal solution for the cost of visiting all nodes with their required frequency of visit is difficult due to the limited computational time, and thus, a semi-optimal solution is accepted in our work as a reasonable solution.

Algorithm 5.3.3: Pseudocode for route generation using SA

Input : Initial temperature, $T_0 = 1e + 10$
Final temperature, $T_f = 0.0001$
Cooling parameter, $\alpha = 0.95$

Output: S_{best}

1: Obtain initial solution $S_0 = \{v_{k_1}, v_{k_2}, \ldots, v_{k_L}\}$ from Algorithm 5.3.2
2: Set initial temperature: $T = T_0$
3: Cost function $C(S)$ is defined as $\ell en(s)$ in Eq. 5.1, where $C(S) = \ell en(s)$
4: Let current solution $S_{cur} = S_0$ whose cost is $C(S_{cur})$, and the best solution $S_{best} = S_0$ whose cost is $C(S_{best})$
5: **repeat**
6: Generate new solution S_{new} by randomly swapping two nodes in S_0 and get its cost $C(S_{new})$
if $k_i \neq k_{i+1}$ is not satisfied **then**
Regenerate S_{new} and $C(S_{new})$
7: **end**
8: Compute relative change in cost: $\delta = C(S_{new}) - C(S_{cur})$
9: Acceptance probability: $P(\delta, T) = exp(-\delta/T)$, where $T > 0$
10: **if** $\delta \leq 0$ or $P(\delta, T) > random[0, 1)$ **then**
11: $S_{cur} = S_{new}$ and $C(S_{cur}) = C(S_{new})$
12: **else if** $C(S_{new}) \leq C(S_{best})$ **then**
13: $S_{best} = S_{new}$ and $C(S_{best}) = C(S_{new})$
14: **end**
15: Compute new temperature: $T = \alpha \times T$
16: **until** $T < T_f$
17: return S_{best}, $C(S_{best})$

We use a simulated annealing here as a sequencing algorithm to find the shortest route, s_i, for patrolling. As our problem is a multi-agent patrolling problem, m routes will be generated in this step where $m = |A|$. However, in the multiple traveling salesman problem, in order to solve it in an easier and simpler way, a heuristic is formed to transform mTSP to TSP and then optimize the tour of each individual salesman. Because our problem is similar to the mTSP as mentioned in Section 5.1, we did the same by applying SA to each cluster to find the best route for each patroller agent in order to make the problem simpler.

Although the SA algorithm has been widely used in mTSP, we have modified and adapted it to our model with non-uniform frequency of visit to each node. The classical SA algorithm in mTSP generates the best solution/route such that each node must be visited exactly once, while our modified SA algorithm constructs the best route for each patroller agent based on the required frequency of visit, where each node is visited at least $f(v_i)$ times by taking into account the condition from Eq. 5.4. Furthermore, we have also modified the process of computing an initial solution in the SA by implementing a greedy approach instead of random approach to find an initial feasible solution. The computation of an initial feasible solution with the implementation of greedy strategy is described as follows, and the pseudocode of this process is shown in Algorithm 5.3.2.

(1) Let S_0 be an initial solution in SA, where S_0 is initially an empty set.

(2) Randomly select a current node, *curNode*, from V, and add it into S_0.

(3) Do the following steps if the number of element in S_0 is not equal to the number of node in V, and V has not yet become an empty set.

  (3.1) Find the shortest distance from the current node to another node in V.

  (3.2) Then, the new *curNode* is the one that has the shortest distance from the old *curNode*.

  (3.3) Add the new *curNode* into S_0.

(3.4) If the new *curNode* that has been added into S_0 is redundant with the previous one, regenerate the new *curNode* by going to step (3.1) again.

(3.5) If the occurence of *curNode* in S_0 is greater than or equal to its associated $f(v_i)$ and is less than or equal to its associated $f(v_i)$ multiplied by 2, remove the new *curNode* from V.

The process of how we applied SA to our model with non-uniform frequency of visit to find the shortest route is described as follows, and its process in pseudocode is illustrated in Algorithm 5.3.3.

(1) The initial solution $S_0 = \{v_{k_1}, v_{k_2}, \ldots, v_{k_L}\}$ is obtained from Algorithm 5.3.2

(2) Set the initial temperature, $T = T_0$, where $T_0 = 1e + 10$

(3) Set the final temperature, $T_f = 0.0001$ and the cooling parameter, $\alpha = 0.95$

(4) Let $S_{cur} = S_0$ and $S_{best} = S_0$ be the current solution and the best solution, respectively, whose cost can be calculated using Eq. 5.1 and are represented by $C(S_{cur})$ and $C(S_{best})$, respectively.

(5) Repeat the following steps until the stopping criterion is met (when the minimum temperature is reached).

(5.1) Generate the new solution, S_{new}, by randomly swapping two nodes in S_0 and compute its cost, $C(S_{new})$.

(5.2) If the two swapped nodes are redundant, go to step (6) again to regenerate S_{new} and recompute $C(S_{new})$.

(5.3) Compute the relative change in cost, δ, which is the difference between the cost of new solution and current solution.

(5.4) If δ is less than or equal to zero, the new solution is accepted as the best solution. Otherwise, the new solution is accepted based on the acceptance probability function.

(5.5) Decrease the temperature.

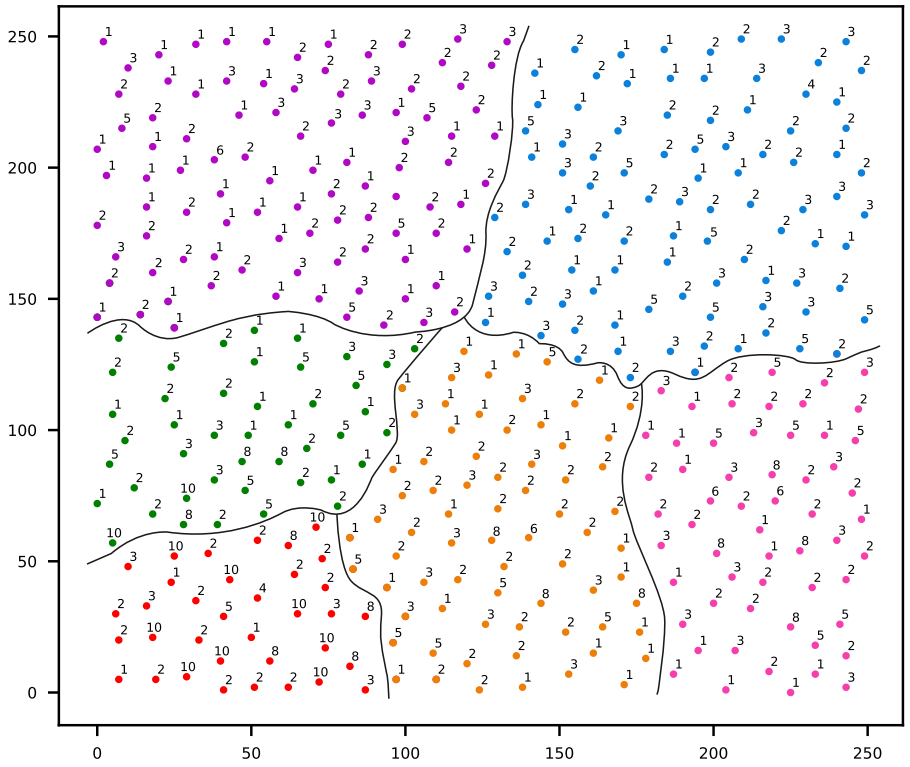

FIGURE 5.1: Clustering by proposed method with $n = 400, m = 6$

5.4 Experimental Evaluation

The proposed algorithms have been implemented in Python 3.5. All computational results are the averages of 20 trials, and are obtained on a personal computer with Intel(R) Core(TM) i5-6200U CPU @2.30 GHz processor and 8GB RAM running on Windows 10 64-bit. To run experiments, we generated the coordinates of all nodes whose ranges are $x \in [0, 250]$ and $y \in [0, 250]$ and their corresponding frequencies of visit $f(v_i)$, which are randomly distributed in the Euclidean space. We have tested our proposed method with different number of nodes and number of agents to see how well our algorithms can work when the number of nodes and agents increase respectively. In this work, we had run our experiments with 5 different number of nodes, $n = |V|$ is 200, 400, 600, 800 and 1000. We had also tried these with different number of agents, $m = |A|$ is 4, 6, 8 and 10. Moreover, we set $M = 10$ in our experiments.

From the best of our knowledge, if M is too small, the solution may not exist, and if it is too large, the solution is not acceptable because agents' works are imbalanced. Therefore, we have to define M according to the problem setting.

After running 20 experiments, we randomly plot the result of one experiment as shown in Fig. 5.1. Figure 5.1 presents the result of that plot

TABLE 5.1: Numerical results with $n = 400$ and $m = \{4, 6, 8, 10\}$

400 nodes				
No. of agent (m)	**Agent** (A)	**Workload** (W_{C_s})	**Cost of Route** ($\ell en(s_i)$)	**Difference** (T_{diff})
4	1	14030	14039	5.66
	2	14035	14046	
	3	14039	14048	
	4	14040	14049	
6	1	11635	11640	4.80
	2	11630	11638	
	3	11632	11641	
	4	11640	11647	
	5	11634	11642	
	6	11629	11635	
8	1	9330	9341	4.96
	2	9333	9342	
	3	9335	9340	
	4	9339	9345	
	5	9340	9351	
	6	9331	9343	
	7	9329	9335	
	8	9336	9344	
10	1	7136	7142	5.31
	2	7130	7140	
	3	7135	7139	
	4	7125	7134	
	5	7137	7145	
	6	7136	7147	
	7	7127	7138	
	8	7132	7141	
	9	7139	7146	
	10	7134	7143	

among 20 plots obtained from graph clustering using our proposed IF-k-means with $n = 400$ and $m = 6$, where the number on each node represents its required frequency of visit. According to Fig. 5.1, we could see that the sizes of all clusters are varied in accordance with the values of $f(v_i)$ in each cluster. Some clusters tend to have small size due to the existence of many values of high visiting frequency in their clusters, while others seem to have

TABLE 5.2: Numerical results with $n = 600$ and $m = \{4, 6, 8, 10\}$

600 nodes				
No. of agent (m)	**Agent** (A)	**Workload** (W_{C_s})	**Cost of Route** ($\ell en(s_i)$)	**Difference** (T_{diff})
4	1	21230	21238	5.66
	2	21236	21244	
	3	21240	21248	
	4	21232	21239	
6	1	18825	18834	5.86
	2	18823	18832	
	3	18827	18838	
	4	18829	18839	
	5	18838	18845	
	6	18826	18835	
8	1	16530	16536	5.00
	2	16537	16545	
	3	16531	16539	
	4	16529	16538	
	5	16535	16543	
	6	16540	16552	
	7	16532	16549	
	8	16528	16537	
10	1	14326	14330	7.02
	2	14328	14337	
	3	14333	14340	
	4	14336	14342	
	5	14324	14335	
	6	14338	14348	
	7	14323	14332	
	8	14329	14336	
	9	14321	14329	
	10	14334	14345	

bigger size because there are less high frequencies of visit in their clusters comparing to those with smaller size. This kind of phenomenon happened because we tried to balance the workload of each cluster. We, thus, say that our proposed clustering algorithm could effectively partition a given graph in a balanced manner.

TABLE 5.3: Numerical results with $n = 1000$ and $m = \{4, 6, 8, 10\}$

1000 nodes				
No. of agent (m)	**Agent** (A)	**Workload** (W_{C_s})	**Cost of Route** ($\ell en(s_i)$)	**Difference** (T_{diff})
4	1	33250	33259	6.16
	2	33260	33268	
	3	33262	33270	
	4	33259	33265	
6	1	30824	30834	7.40
	2	30835	30840	
	3	30841	30849	
	4	30837	30848	
	5	30840	30851	
	6	30842	30853	
8	1	28530	28537	5.46
	2	28538	28542	
	3	28535	28540	
	4	28540	28545	
	5	28545	28551	
	6	28537	28541	
	7	28542	28548	
	8	28536	28543	
10	1	26265	26273	8.02
	2	26273	26278	
	3	26268	26276	
	4	26255	26266	
	5	26260	26269	
	6	26256	26264	
	7	26254	26262	
	8	26268	26275	
	9	26264	26273	
	10	26270	26281	

To evaluate the effectiveness and performance of our proposed work, the expected workload (W_{C_s}), the cost of route ($\ell en(s_i)$) in each cluster and the difference in workload (T_{diff}) are listed in Table 5.1 to 5.3. These tables show the numerical results with the number of agents, $m = \{4, 6, 8, 10\}$ for 400, 600 and 1000 nodes respectively. All these tables demonstrate that the difference in workload always satisfied the condition in Eq. 5.9, where $T_{diff} \leq M$ and $M = 10$. Thus, if IF-k-means cannot find the route whose T_{diff} is less than 10, no solution is generated.

Furthermore, the results from all the tables also clarified that the cost of patrolling in each cluster, $\ell en(s_i)$, has the value which is not much different from its corresponding expected workload, W_{C_s}. This means that the sequencing algorithm in Section 5.3.2 produced a good result in term of generating the route for patrolling and minimizing the cost of each route. Therefore, we conclude that our proposed algorithms not only could balance the workload amongst all agents, but also could generate the patrolling route with a reasonable cost.

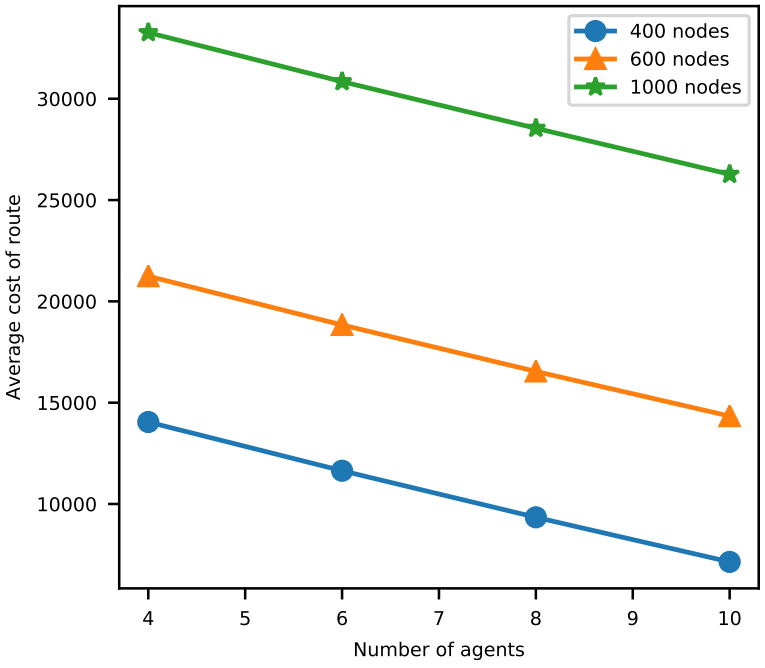

FIGURE 5.2: Average cost of route for each agent

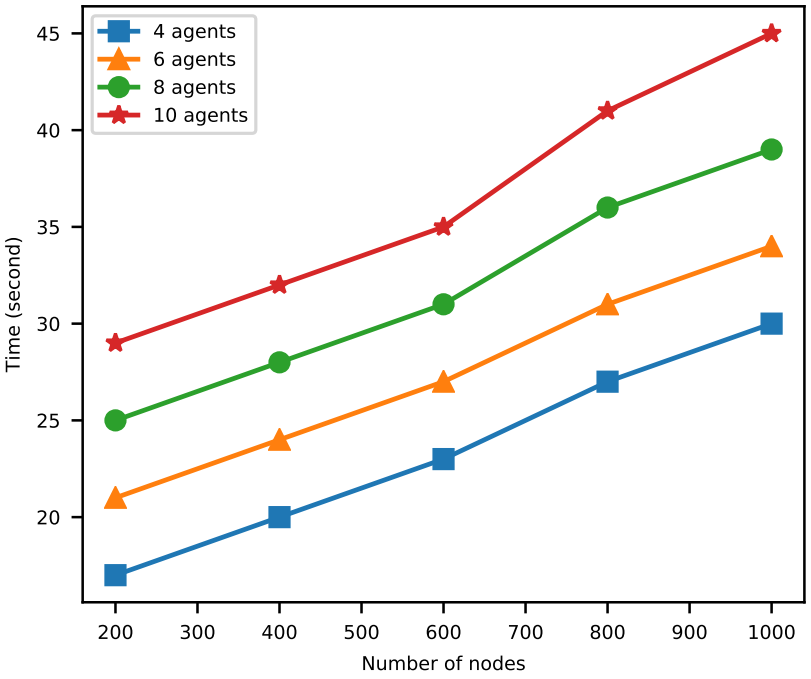

FIGURE 5.3: Computation time of proposed method

In addition, Fig. 5.2 shows the average cost of route for each agent with different number of agents ($m = \{4, 6, 8, 10\}$) and nodes ($n = \{400, 600, 1000\}$). According to Fig. 5.2, there exist the downward slopping trends of average route cost regardless of the number of nodes. This implies that the average cost of route declines because the more the number of agents are, the less the size of each agent's RA will be.

Besides the effectiveness of performing area partition and sub-area patrolling, we also considered the computation time as a significant factor to demonstrate the efficiency of our proposed work. Figure 5.3 indicates the computation time of our proposed method in second. According to Fig. 5.3, we could observe that the computation time increased linearly in accordance with the number of nodes and the number of agents. This shows that the proposed algorithms could be computed in a short amount of time, and thus, we conclude that our proposed method is computational efficiency.

5.5 Summary

A new frequency-based area partitioning method for balanced workload in multi-agent patrolling systems has been presented. This proposed work considered the non-uniform visitation requirement for each location, where its frequency of visit is high or low depending on the level of importance of that location. Because non-uniform visiting frequencies of all locations could affect the quality of clustering, the main goal of our work, thus, aims at balancing the workload of each cluster/agent so as to improve the workers morale. Besides the balance in workload, we also believe that computational cost plays a significant role in proving the effectiveness and computational efficiency of the proposed work. Experimental results demonstrated that our proposed method could effectively generate clusters of a given area regarding the non-uniform visitation requirements in a balanced manner and in a satisfied short amount of time.

A significant benefit of our work is the balanced task division for multi-agent patrolling task with the consideration of the real-world environment, where the visitation requirement of each location is not uniform. In realisic application (i.e., security patrolling), each location to be patrolled has different visitation requirement or risk status according to the required security level; thus, our work is well-suited to the real-world environment. However, we have not considered about the minimum time interval between the visits to a node that needs frequent patrolling, for there is a trade-off between trying to minimize the total cost (length) of route and trying to minimize the time interval between the visits for the frequent-visit node at the same time in this model, where the frequency of visit is not uniform. Therefore, we believe that incorporating the penalty function into our method will be an ideal solution to prevent the patroller agents from visiting nodes too often or too seldom; this will be our future work.

Chapter 6

Conclusion

6.1 Conclusion

We have presented decentralized and frequency-based area partitioning methods for balanced workload in multi-agent patrolling teams. Initially, we introduced a coordinated area partitioning method by autonomous agents for continuous cooperative tasks. In this work, we proposed an area partitioning method for cooperative cleaning robots in the environments with obstacles and with learning to identify the easy-to-dirty areas. Our study aims at coordination and cooperation by multiple agents, and we discuss it using an example of the cleaning task to be performed by multiple agents with potentially different performances and capabilities. We then developed a method for partitioning the target area on the basis of agents' performances in order to improve the overall efficiency through their balanced collective efforts. Agents autonomously decide in a cooperative manner how the area/task is partitioned by taking into account the characteristics of the environment and the differences in agents' software capability and hardware performance. During this partitioning process, agents also learn the locations of obstacles and the probabilities of dirt accumulation that express which areas that dirt tends to easily accumulate. Experimental evaluation demonstrated that even if the agents use different algorithms or have the batteries with different capacities resulting in different performances, and even if the environment is not uniform such as different

locations of easy-to-dirty areas and obstacles, the proposed method can adaptively partition the task/area among the agents with the learning of the probabilities of dirt accumulation. Thus, agents with the proposed method can keep the area clean evenly and effectively.

Although the aforementioned proposed work could yield better results in term of balanced task sharing, comparing to the conventional methods which assumed that the area is divided into equal-size subareas and/or the environmental characteristics are given in advance, this proposed study mainly focuses on the continuous cleaning/sweeping task which is somehow restricted in some other real-world patrolling applications.

Therefore, we have extended our work by introducing a more general method that can be applied to a number of realistic applications related to patrolling context. In this work, a frequency-based multi-agent patrolling model and its area partitioning solution method for balanced workload has then been proposed to deal with the above restricted scope problem as well as the real-world requirement of the visiting frequency to each location. This proposed work considered the non-uniform visitation requirement for each location, where its frequency of visit is high or low depending on the level of importance of that location. We formulated the problem of frequency-based multi-agent patrolling and proposed its semi-optimal solution method, whose overall process consists of two steps – graph partitioning and sub-graph patrolling. Because non-uniform visiting frequencies of all locations could affect the quality of clustering, the main goal of this work, thus, aims at partitioning a given area so as to balance agents' workload by taking into account the different visitation requirement. Then, another goal is to generate the route for each agent to patrol inside its allocated sub-area, such that the total cost of route is minimized. This proposed work is useful and preferable for the realistic environments, where the target area to be patrolled is not always uniform.

Besides the balance in workload, we also believe that computational cost plays a significant role in confirming the effectiveness and efficiency of the proposed work. Experimental results illustrated the effectiveness and reasonable computational efficiency of our approach. That is, our proposed method could effectively generate clusters of a given area regarding the non-uniform visitation requirements in a balanced manner and in a reasonable short amount of time.

6.2 Future Work

Regarding our future work, we intend to study the problem of multi-agent patrolling systems in a more realistic environment. Moreover, we plan to further extend our work by taking into account the corresponding minimum time interval between the visits to a node that needs frequent patrolling. Also, we attempt to incorporate the penalty function into our method in order to prevent the patroller agents from visiting nodes too often or too seldom.

We have not considered these problems in our current work, for there is a trade-off between trying to minimize the total cost of route and trying to minimize the time interval between the visits for the frequent-visit node at the same time in this model, where the frequency of visit to each location is not uniform. Therefore, the above-mentioned issues would be a good future direction for other researchers to consider and improve their work, which could lead to an interesting path to explore as a future research area.

Bibliography

[1] "Advantages of machine security service." `https://www.security-law.com/security-services-act/kikai.html`.

[2] Acevedo, J. J., Arrue, B. C., Diaz-Bañez, J. M., Ventura, I., Maza, I., and Ollero, A., "One-to-one coordination algorithm for decentralized area partition in surveillance missions with a team of aerial robots," *Journal of Intelligent & Robotic Systems*, 74(1-2):269–285, 2014.

[3] Agmon, N. and Peleg, D., "Fault-tolerant gathering algorithms for autonomous mobile robots," *SIAM Journal on Computing*, 36(1):56–82, 2006.

[4] Ahmadi, M. and Stone, P., "Continuous area sweeping: A task definition and initial approach," in *Proceedings of the 12th International Conference on Advanced Robotics (ICAR'05)*, pp. 316–323. IEEE, 2005.

[5] Ahmadi, M. and Stone, P., "A multi-robot system for continuous area sweeping tasks," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA'06)*, pp. 1724–1729. IEEE, 2006.

[6] Alam, T., "Decentralized and nondeterministic multi-robot area patrolling in adversarial environments," *International Journal of Computer Applications*, 156(2), 2016.

[7] Almeida, A., Ramalho, G., Santana, H., Tedesco, P., Menezes, T., Corruble, V., and Chevaleyre, Y., "Recent advances on multi-agent

patrolling," in *Brazilian Symposium on Artificial Intelligence*, pp. 474–483. Springer, 2004.

[8] Arai, T., Pagello, E., and Parker, L. E., "Advances in multi-robot systems," *IEEE Transactions on robotics and automation*, 18(5):655–661, 2002.

[9] Bast, H. and Hert, S., "The area partitioning problem," pp. 163–171, 2000.

[10] Bektas, T., "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, 34(3):209–219, 2006.

[11] Beynier, A., "A multiagent planning approach for cooperative patrolling with non-stationary adversaries," *International Journal on Artificial Intelligence Tools*, 26(05):1760018, 2017.

[12] Breitenmoser, A., Schwager, M., Metzger, J.-C., Siegwart, R., and Rus, D., "Voronoi coverage of non-convex environments with a group of networked robots," in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA'10)*, pp. 4982–4989. IEEE, 2010.

[13] Calvo, R. W. and Cordone, R., "A heuristic approach to the overnight security service problem," *Computers & Operations Research*, 30(9):1269–1287, 2003.

[14] Cao, Y. U., Fukunaga, A. S., and Kahng, A., "Cooperative mobile robotics: Antecedents and directions," *Autonomous robots*, 4(1):7–27, 1997.

[15] Cepeda, J. S., Chaimowicz, L., Soto, R., Gordillo, J. L., Alanís-Reyes, E. A., and Carrillo-Arce, L. C., "A behavior-based strategy for single and multi-robot autonomous exploration," *Sensors*, 12(9):12772–12797, 2012.

[16] Chao, I.-M., Golden, B. L., and Wasil, E. A., "A fast and effective heuristic for the orienteering problem," *European journal of operational research*, 88(3):475–489, 1996.

[17] Chen, S., Wu, F., Shen, L., Chen, J., and Ramchurn, S. D., "Multi-agent patrolling under uncertainty and threats," *PLoS ONE*, 10(6):e0130154, 2015.

[18] Chevaleyre, Y., "Theoretical analysis of the multi-agent patrolling problem," in *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04)*, pp. 302–308. IEEE, 2004.

[19] Chevaleyre, Y., "The patrolling problem: theoretical and experimental results," *Combinatorial Optimization and Theoretical Computer Science*, pp. 161–174, 2007.

[20] Chevaleyre, Y., Sempe, F., and Ramalho, G., "A theoretical analysis of multi-agent patrolling strategies," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, vol. 3, pp. 1524–1525. IEEE Computer Society, 2004.

[21] Chu, H. N., Glad, A., Simonin, O., Sempe, F., Drogoul, A., and Charpillet, F., "Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation," in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'07)*, vol. 1, pp. 442–449. IEEE, 2007.

[22] Cortes, J., Martinez, S., and Bullo, F., "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, 11(4):691–719, 2005.

[23] Du, K. L. and Swamy, M. N. S., *Search and optimization by metaheuristics*. Springer, 2016.

[24] Dudek, G., Jenkin, M. R., Milios, E., and Wilkes, D., "A taxonomy for multi-agent robotics," *Autonomous Robots*, 3(4):375–397, 1996.

[25] Dutta, A., Dasgupta, P., Baca, J., and Nelson, C., "A bottom-up search algorithm for dynamic reformation of agent coalitions under coalition size constraints," in *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*, pp. 329–336. IEEE Computer Society, 2013.

[26] Elmaliach, Y., Agmon, N., and Kaminka, G. A., "Multi-robot area patrol under frequency constraints," *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.

[27] Elor, Y. and Bruckstein, A. M., "Multi-a(ge)nt graph patrolling and partitioning," in *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'09)*, vol. 2, pp. 52–57. IEEE, 2009.

[28] Fazli, P., Davoodi, A., and Mackworth, A. K., "Multi-robot repeated area coverage," *Autonomous robots*, 34(4):251–276, 2013.

[29] Glad, A., Simonin, O., Buffet, O., and Charpillet, F., "Theoretical study of ant-based algorithms for multi-agent patrolling," in *Proceedings of the 18th European Conference on Artificial Intelligence including Prestigious Applications of Intelligent Systems (PAIS 2008)-ECAI 2008*, pp. 626–630. IOS press, 2008.

[30] Granta, "Advantages and disadvantages of robotic automation." `http://www.granta-automation.co.uk/news/advantages-and-disadvantages-of-robotic-automation/`, 2017.

[31] Hahnel, D., Burgard, W., Fox, D., and Thrun, S., "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, vol. 1, pp. 206–211. IEEE, 2003.

[32] Hahsler, M. and Hornik, K., "Tsp-infrastructure for the traveling salesperson problem," *Journal of Statistical Software*, 23(2):1–21, 2007.

[33] Harlan, B. and Lamar, S., "Advantages of robotics in engineering." `https://www.\\brighthubengineering.com/robotics/76606-advantages-of-robotics-in-engineering/`, 2010.

[34] Hennes, D., Claes, D., Meeussen, W., and Tuyls, K., "Multi-robot collision avoidance with localization uncertainty," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 147–154. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[35] Heppner, G., Roennau, A., and Dillman, R., "Enhancing sensor capabilities of walking robots through cooperative exploration with aerial robots," *Journal of Automation Mobile Robotics and Intelligent Systems*, 7(2), 2013.

[36] Hernández, E., del Cerro, J., and Barrientos, A., "Game theory models for multi-robot patrolling of infrastructures," *International Journal of Advanced Robotic Systems*, 10(3):181, 2013.

[37] Hert, S. and Lumelsky, V., "Polygon area decomposition for multiple-robot workspace division," *International Journal of Computational Geometry & Applications*, 8(04):437–466, 1998.

[38] IFR, "Executive summary world robotics 2016 industrial robots." `https://ifr.org/img/uploads/Executive_Summary_WR_Industrial_Robots_20161.pdf`, 2016.

[39] IFR, "Executive summary world robotics 2016 service robots." `https://ifr.org/downloads/press/02_2016/Executive_Summary_Service_Robots_2016.pdf`, 2016.

[40] IFR, "Executive summary world robotics 2017 industrial robots." `https://ifr.org/downloads/press/Executive_Summary_WR_2017_Industrial_Robots.pdf`, 2017.

[41] Jager, M. and Nebel, B., "Dynamic decentralized area partitioning for cooperating cleaning robots," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA'02)*, vol. 4, pp. 3577–3582. IEEE, 2002.

[42] Kalra, N., Stentz, T., and Ferguson, D., "Hoplites: A market framework for complex tight coordination in multi-agent teams," tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 2004.

[43] Karimov, J. and Ozbayoglu, M., "Clustering quality improvement of k-means using a hybrid evolutionary model," *Procedia Computer Science*, 61:38–45, 2015.

[44] Kato, C. and Sugawara, T., "Decentralized area partitioning for a cooperative cleaning task," in *Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'13)*, pp. 470–477. Springer, Berlin, Heidelberg, 2013.

[45] Kim, K. H. and Park, Y.-M., "A crane scheduling method for port container terminals," *European Journal of operational research*, 156(3):752–768, 2004.

[46] Kurabayashi, D., Ota, J., Arai, T., and Yoshida, E., "Cooperative sweeping by multiple mobile robots," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1744–1749. IEEE, 1996.

[47] Lauri, F. and Charpillet, F., "Ant colony optimization applied to the multi-agent patrolling problem," in *IEEE Swarm Intelligence Symposium*, 2006.

[48] Lauri, F. and Koukam, A., "A two-step evolutionary and aco approach for solving the multi-agent patrolling problem," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 861–868. IEEE, 2008.

[49] Lauri, F. and Koukam, A., "Hybrid aco/ea algorithms applied to the multi-agent patrolling problem," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 250–257. IEEE, 2014.

[50] Luo, C. and Yang, S. X., "A real-time cooperative sweeping strategy for multiple cleaning robots," in *Proceedings of the 2002 IEEE International Symposium on Intelligent Control*, pp. 660–665. IEEE, 2002.

[51] Machado, A., Almeida, A., Ramalho, G., Zucker, J.-D., and Drogoul, A., "Multi-agent movement coordination in patrolling," in *Proceedings of the 3rd International Conference on Computer and Game*, pp. 155–170, 2002.

[52] Machado, A., Ramalho, G., Zucker, J.-D., and Drogoul, A., "Multi-agent patrolling: An empirical analysis of alternative architectures," in *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pp. 155–170. Springer, 2002.

[53] Mao, T. and Ray, L., "Frequency-based patrolling with heterogeneous agents and limited communication," *arXiv preprint arXiv:1402.1757*, 2014.

[54] Marier, J.-S., Besse, C., and Chaib-Draa, B., "A markov model for multiagent patrolling in continuous time," in *International Conference on Neural Information Processing*, pp. 648–656. Springer, 2009.

[55] Matai, R., Singh, S., and Mittal, M. L., "Traveling salesman problem: an overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem, Theory and Applications*. InTech, 2010.

[56] McCaffrey, J. D., "Graph partitioning using a simulated bee colony algorithm," in *Proceedings of the 2011 IEEE International Conference on Information Reuse and Integration (IRI'11)*, pp. 400–405. IEEE, 2011.

[57] Mead, R., Weinberg, J. B., and Croxell, J. R., "An implementation of robot formations using local interactions.," in *Proceedings of the national conference on artificial intelligence*, vol. 22, p. 1989. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[58] Menezes, T., Tedesco, P., and Ramalho, G., "Negotiator agents for the patrolling task," in *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, pp. 48–57. Springer, 2006.

[59] Mikhalchuk, A., "Unimate, robostuff." `http://robostuff.com/http://robostuff.com/robots-catalog/robots-by-country/usa/unimate/`, 2009.

[60] Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R., and Parthiban, P., "Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics," *International Journal of Nonlinear Science*, 9(2):171–177, 2010.

[61] Ogston, E., Overeinder, B., Van Steen, M., and Brazier, F., "A method for decentralized clustering in large multi-agent systems," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS'03)*, pp. 789–796. ACM, 2003.

[62] Okonjo-Adigwe, C., "An effective method of balancing the workload amongst salesmen," *Omega*, 16(2):159–163, 1988.

[63] Parker, L. E., "Multiple mobile robot systems," in *Springer Handbook of Robotics*, pp. 921–941. Springer, 2008.

[64] Pemmaraju, S. and Skiena, S., *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica ®*. Cambridge university press, 2003.

[65] Popescu, M.-I., Rivano, H., and Simonin, O., "Multi-robot patrolling in wireless sensor networks using bounded cycle coverage," in *Proceedings*

of the 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI'16), pp. 169–176. IEEE, 2016.

[66] Portugal, D., Pippin, C., Rocha, R. P., and Christensen, H., "Finding optimal routes for multi-robot patrolling in generic graphs," in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'14)*, pp. 363–369. IEEE, 2014.

[67] Portugal, D. and Rocha, R., "Msp algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning," in *Proceedings of the 2010 ACM symposium on applied computing (SAC'10)*, pp. 1271–1276. ACM, 2010.

[68] Portugal, D. and Rocha, R., "A survey on multi-robot patrolling algorithms," in *Doctoral Conference on Computing, Electrical and Industrial Systems*, pp. 139–146. Springer, 2011.

[69] Ranjbar-Sahraei, B., Weiss, G., and Nakisaee, A., "A multi-robot coverage approach based on stigmergic communication," in *German Conference on Multiagent System Technologies*, pp. 126–138. Springer, 2012.

[70] RobotWorx, "Benefits of using robotics." `http://www.robots.com/articles/viewing/benefits-of-using-robotics`, 2017.

[71] Roerty, D. F., *M-salesman balanced tours traveling salesman problem with multiple visits to cities allowed*. PhD thesis, Texas Tech University, 1974.

[72] RSI, "Overview of repetitive strain injury (rsi)." `https://www.nhs.uk/conditions/repetitive-strain-injury-rsi/`, 2016.

[73] Sak, T., Wainer, J., and Goldenstein, S. K., "Probabilistic multiagent patrolling," in *Brazilian Symposium on Artificial Intelligence*, pp. 124–133. Springer, 2008.

[74] Sales, D. O., Feitosa, D., Osório, F. S., and Wolf, D. F., "Multi-agent autonomous patrolling system using ann and fsm control," in *2012*

Second Brazilian Conference on Critical Embedded Systems (CBSEC), pp. 48–53. IEEE, 2012.

[75] Sampaio, P. A., Ramalho, G., and Tedesco, P., "The gravitational strategy for the timed patrolling," in *Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI'10)*, vol. 1, pp. 113–120. IEEE, 2010.

[76] Santana, H., Ramalho, G., Corruble, V., and Ratitch, B., "Multi-agent patrolling with reinforcement learning," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1122–1129. IEEE Computer Society, 2004.

[77] Schwager, M., Rus, D., and Slotine, J.-J., "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *The International Journal of Robotics Research*, 30(3):371–383, 2011.

[78] Sea, V. and Sugawara, T., "Area partitioning method with learning of dirty areas and obstacles in environments for cooperative sweeping robots," in *Proceedings of the 2015 IIAI 4th International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 523–529. IEEE, 2015.

[79] Sempé, F. and Drogoul, A., "Adaptive patrol for a group of robots," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, vol. 3, pp. 2865–2869. IEEE, 2003.

[80] Stranders, R., De Cote, E. M., Rogers, A., and Jennings, N. R., "Near-optimal continuous patrolling with teams of mobile information gathering agents," *Artificial intelligence*, 195:63–105, 2013.

[81] Sugiyama, A., Sea, V., and Sugawara, T., "Effective task allocation by enhancing divisional cooperation in multi-agent continuous patrolling tasks," in *Proceedings of the 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI'16)*, pp. 33–40. IEEE, 2016.

[82] Sugiyama, A. and Sugawara, T., "Autonomous strategy determination with learning of environments in multi-agent continuous cleaning," in *Proceedings of the International Conference on Principles and Practice of Multi-Agent Systems (PRIMA'14)*, pp. 455–462. Springer, 2014.

[83] Sugiyama, A. and Sugawara, T., "Meta-strategy for cooperative tasks with learning of environments in multi-agent continuous tasks," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC'15)*, pp. 494–500. ACM, 2015.

[84] Wagner, I. A., Lindenbaum, M., and Bruckstein, A. M., "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.

[85] Wolf, D. F. and Sukhatme, G. S., "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, 19(1):53–65, 2005.

[86] Yasuyuki, S., Hirofumi, O., Tadashi, M., and Maya, H., "Cooperative capture by multi-agent using reinforcement learning application for security patrol systems," in *Proceedings of the 2015 10th Asian Control Conference (ASCC'15)*, pp. 1–6. IEEE, 2015.

[87] Yoneda, K., Kato, C., and Sugawara, T., "Autonomous learning of target decision strategies without communications for continuous coordinated cleaning tasks," in *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 02*, pp. 216–223. IEEE Computer Society, 2013.

[88] Yoneda, K., Sugiyama, A., Kato, C., and Sugawara, T., "Learning and relearning of target decision strategies in continuous coordinated cleaning tasks with shallow coordination1," in *Web Intelligence*, vol. 13, pp. 279–294. IOS Press, 2015.

[89] Zach, "Industrial robotics arm." `http://www.flickr.com/photos/7204008@N06/410462775`, 2007.

List of Publications

1. ◯ **Vourchteang Sea**, Chihiro Kato, and Toshiharu Sugawara, "Coordinated Area Partitioning Method by Autonomous Agents for Continuous Cooperative Tasks," *Journal of Information Processing* (*JIP*), 25(1):75–87, January 2017.

2. ◯ **Vourchteang Sea**, Ayumi Sugiyama, and Toshiharu Sugawara, "Frequency-Based Multi-agent Patrolling Model and Its Area Partitioning Solution Method for Balanced Workload," in *the* 15^{th} *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (*CPAIOR2018*), pp. 530–545. Springer, Cham, Delft, The Netherlands, June 26-29, 2018.

3. ◯ **Vourchteang Sea** and Toshiharu Sugawara, "Area Partitioning Method with Learning of Dirty Areas and Obstacles in Environments for Cooperative Sweeping Robots," in *the* 4^{th} *IIAI International Congress on Advanced Applied Informatics* (*IIAI-AAI2015*), pp. 523–529. IEEE, July 12-16, 2015. (Best Student Paper Award)

4. Ayumi Sugiyama, **Vourchteang Sea**, and Toshiharu Sugawara, "Effective Task Allocation by Enhancing Divisional Cooperation in Multi-Agent Continuous Patrolling Tasks," in *the* 28^{th} *IEEE International Conference on Tools with Artificial Intelligence* (*ICTAI2016*), pp. 33–40. IEEE, San Jose, USA, Nov. 6-8, 2016.

5. Ayumi Sugiyama, **Vourchteang Sea**, and Toshiharu Sugawara, "An Effective Autonomous Task Allocation for division of labor in Multi-Agent Continuous Patrolling Tasks (in Japanese)," in *the* 15^{th} *Joint Agent Workshop* (*JAWS2016*), Sponsored by JSSST, IEICE, JSAI and IPSJ, Yamanaka Onsen, Japan, Sep. 15-16, 2016.

6. Ayumi Sugiyama, **Vourchteang Sea**, and Toshiharu Sugawara, "Method of Promoting Division of Labor by Using Communication for Multi-agent Continuous Cleaning (in Japanese)," in *the* 30^{th} *Annual Conference of the Japanese Society for Artificial Intelligence*, 2016.