UNIVERSITÄT DUISBURG-ESSEN
FAKULTÄT FÜR WIRTSCHAFTSWISSENSCHAFTEN
INSTITUT FÜR INFORMATIK UND WIRTSCHAFTSINFORMATIK
LEHRSTUHL FÜR PERVASIVE COMPUTING

Masterarbeit

Exploiting Knowledge of Room Occupation for the Scheduling of Navigation Tasks of a Fleet of Robots in Office Environments

Xuanjiao Zhu
Matrikelnummer: 3038674

Networked Embedded Systems Group
Institut für Informatik und Wirtschaftsinformatik
Fakultät für Wirtschaftswissenschaften
Universität Duisburg-Essen

18. Oktober 2020

Erstprüfer: Prof. Dr. Pedro José Marrón
Zweitprüfer: Prof. Dr. Gregor Schiele
Zeitraum: 1.May 2020 - 1.October 2020

Contents

Abstract

The function of the abstract is to summarize, in one or two paragraphs, the major aspects of the entire bachelor or master thesis. It is usually written after writing most of the chapters.

It should include the following:

- Definition of the problem (the question(s) that you want to answer) and its purpose (Introduction).
- Methods used and experiments designed to solve it. Try to describe it basically, without covering too many details.
- Quantitative results or conclusions. Talk about the final results in a general way and how they can solve the problem (how they answer the question(s)).

Even if the Title can be a reference of the work's meaning, the Abstract should help the reader to understand in a quick view, the full meaning of the work. The abstract length should be around 300 words.

Abstracts are protected under copyright law just as any other form of written speech is protected. However, publishers of scientific articles invariably make abstracts publicly available, even when the article itself is protected by a toll barrier. For example, articles in the biomedical literature are available publicly from MEDLINE which is accessible through PubMed. It is a common misconception that the abstracts in MEDLINE provide sufficient information for medical practitioners, students, scholars and patients[citation needed]. The abstract can convey the main results and conclusions of a scientific article but the full text article must be consulted for details of the methodology, the full experimental results, and a critical discussion of the interpretations and conclusions. Consulting the abstract alone is inadequate for scholarship and may lead to inappropriate medical decisions[2].

An abstract[IGM97, **?**, MAdR02, Sal89] allows one to sift through copious amounts of papers for ones in which the researcher can have more confidence that they will be relevant to his research. Once papers are chosen based on the abstract, they must be read carefully to be evaluated for relevance. It is commonly surmised that one must not base reference citations on the abstract alone, but the entire merits of a paper.

Chapter 1

Introduction

[You should answer the question: What is the problem?]

This paragraph should establish the context of the reported work. To do that, authors discuss over related literature (with citations[1]) and summarize the knowledge of the author in the investigated problem.

ToDo: how to make citations

An introduction should answer (most of) the following questions:

- What is the problem that I want to solve?
- Why is it a relevant question?
- What is known before the study?
- How can the study improve the current solutions?

To write it, use if possible active voice:

- We are going to watch a film tonight (Active voice).
- A film is going to be watched by us tonight (Passive voice).

The use of the first person is accepted.

1.1 Motivation

A good introduction usually starts presenting a general view of the topic and continues focusing on the problem studied. Begin it clarifying the subject area of interest and establishing the context (remember to support it with related bibliography).

[1] To cite a work in latex

1.2 Problem definition

Additionally, focuses the text on the relevant points of your investigation and problems that you want to solve, relating them with the first part.

1.3 Thesis/Diplom/Bachelor/Master Structure

Present your work to the reader giving a brief overview of what is going to cover every chapter. Write only general concepts, no more than one or two sentences per chapter should be necessary.

Chapter 2

Materials and Methods

This Chapter introduced the background and state of the art on multi-robot task scheduling. Chapter 2.1 introduces important concepts of ROS. Chapter 2.2 introduces the 3D modeling of robot and environment in Gazebo. Chapter 2.3 introduces robot navigation. Chapter **??** introduces exist task scheduling methods. Chapter **??** introduces distributed methods. Chapter 2.5 introduces the cost funtion methods.

2.1 Important Concepts of ROS

The ROS Wiki [ros] defines ROS as an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. In another word, it is a robot software platform that provides various development environments specialized for developing robot application programs[HC17].

2.1.1 Node

A ROS node is the smallest unit of processor running in ROS. In another word, it is one executeble program. In this project, some existing special nodes are used. For example "move_base" node provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. There are some nodes created for this project, and each node are created for different purpose, for example, one "Robor controller" node controls one robot.

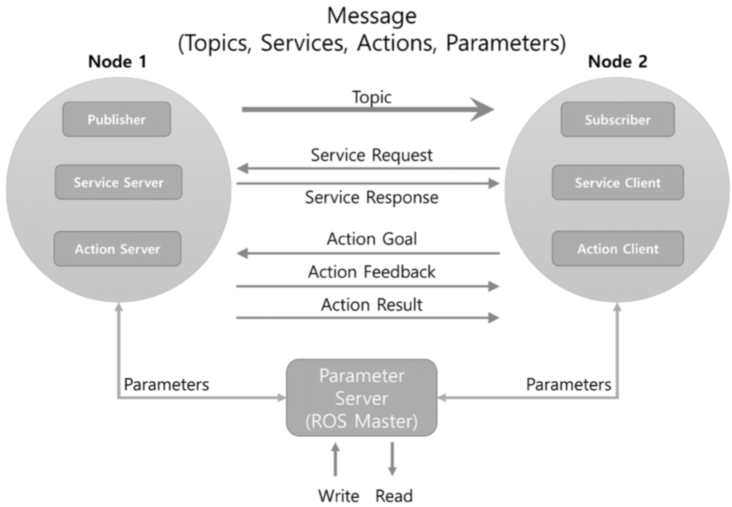


Figure 2.1: ROS Message Communication

2.1.2 Communication Infrastructure

ROS has a built-in and well-tested messaging system. There are different methods to exchange messages. ROS topic is a unidirectional anonymous communication. It is used when exanging data continuesly. The subscriber receives messages of publisher node only when both of them registered the same topic name. ROS service is bidirectional synchronous communication. The service client request a service and the service server responds to the request. The ROS action is a bidirectional asynchronous communication. It is used when it is difficult to use the service due to long response times after the request or when an intermediate feedback value is needed. The diagram of message communication is shown in Figure 2.1. There are some utilization of message communication in this project. Chapter 5.2 introduces the sensor simulation senario that utilizes ROS Topic method. Chapter 4.1 introduces the communication protocols in this project that utilize ROS service method and ROS Action method.

2.1.3 ROS Tools

ROS's core functionality is enhanced by a variety of tools and packages:

- **Rviz.** Rviz[RVI] is the 3D visualization tool of ROS. It can visualize information like the distance from a Laser Distance Sensor (LDS) to an obstable, image value obtained from a camera, Point Cloud Data (PCD) of the 3D distance sensor such as RealSense, Kinect, or Xtion. As is shown in Figure 2.2, multiple robot model and its path and laser data can be displayed.

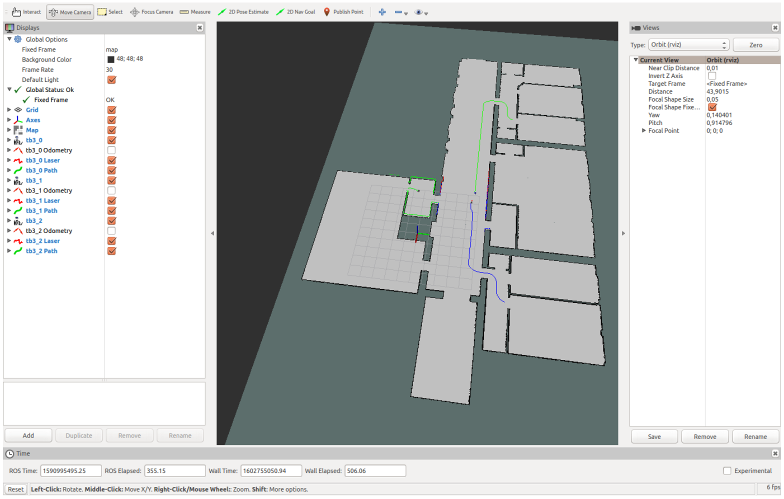


Figure 2.2: Rviz Example: Navigaion using TurtleBot 3 and Laser Distance Sensor (LDS)

- **rqt.** rqt is an intergration of more than 30 Qt-based ROS GUI development tool. It has plugins such as "rqt_tf_tree", "rqt_plot" and "rqt_graph"

- **rqt_tf_tree.** rqt_tf_tree is a type of rqt. It is a tool for visualizing the tree of frames being broadcast over ROS. Figure 2.3 presents the relative coordinate transformation (tf) of multi-robot system. If the poses of Laser Distance Sensor (LDS) are considered as the poses of the robots, the pose information of each robot is connected in the order of odom → base_footprint → base_link → base_scan.

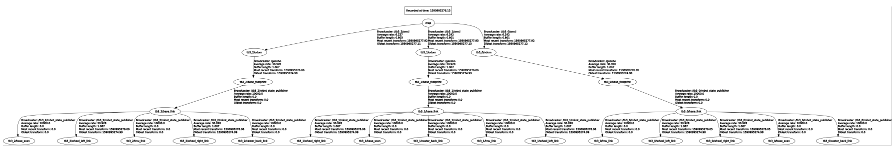


Figure 2.3: rqt_tf_tree of Multi-robot Scheduling System

- **rqt_graph.** rqt_graph is a type of rqt. It is a graphical tool that presents the status of nodes and topics. Figure 2.4 presents relation of nodes in Multi-robot Scheduling System.

- **Gazebo.** Gazebo [GZ] is the 3D simulation tool intergrated with ROS. The details of gezebo are introduced in Chapter 2.2.

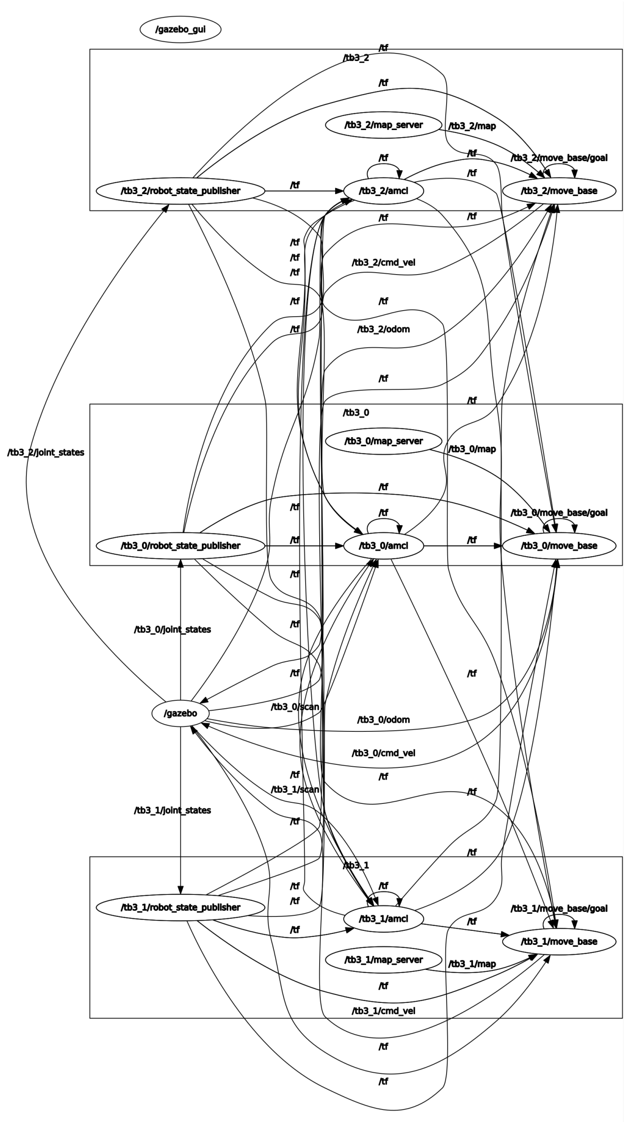


Figure 2.4: rqt_graph of Multi-robot Scheduling System

2.2 TurtleBot3 Simulation Using Gazebo

2.2.1 Gazebo Simulator

Robot simulation is essential for robotics research, because it can pre-estimate the performance of robot algorithms before applying it to a real robot [ASM15]. The simulatior used in this project is Gazebo. Gazebo is a 3D dynamic indoor and outdoor multi-robot simulator intergrated with ROS, and it offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs [GZ]. Using Gazebo simulatior affords to create new 3D model with geometrical primitive or import existing simulated robots and environments.

2.2.2 TurtleBot3 Robot

The robot model used in this project is TurtleBot3 Burger. Because it is a small, affordable, programmable ROS-based mobile robot for use in research and prototyping. As is shown in Figure 2.5, the basic components are actuators, an SBC (Single-Board Computer) for operating ROS, a LDS sensor for SLAM (Chapter 2.3) and navigation, restructurable mechanism, an OpenCR embedded board used as a sub-controller, sprocket wheels that can be used with tire and caterpillar, and a 3 cell lithium-poly battery. The simulated robot (Figure 2.6) has a similar outfit. In addition, Gazebo simulates the robot locomotion and sensor measurements used for localization and navigation, and exports the simulation results to ROS.

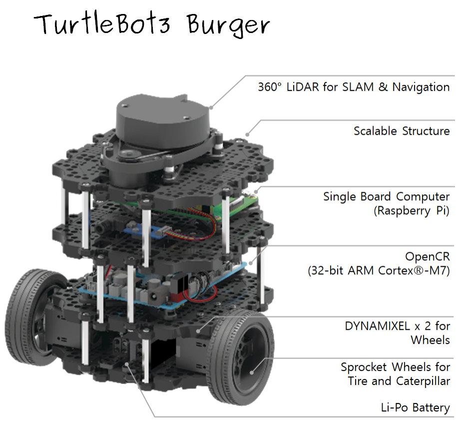


Figure 2.5: Robot Hardware Configuration

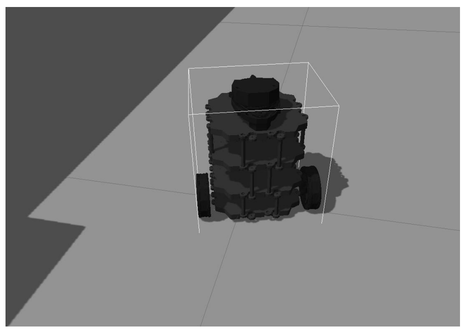


Figure 2.6: Robot 3D Modeling in Gazebo

2.2.3 3D Modeling of Indoor Environment

In this project we selected a model exactly the same as the floor of the department as a trial 3D model (Figure 2.7). This model is a typical office enviroment that contains a corridor along the central x-axis and 16 rooms located around the corridor.

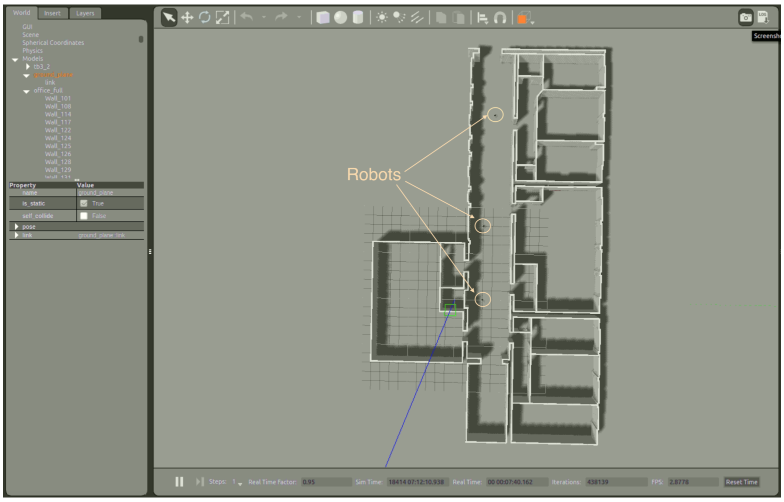


Figure 2.7: 3D Modeling of Indoor Environment in Gazebo

2.3 Robot Navigation and Virtual SLAM

There are essential technologies to realize autonomous robot navigation:

1. **Having map of the given enviroment.** In this project, SLAM(Simultaneous Localization Ana Mapping) is used to create a map of the given enviroment. Using SLAM, the robot explores the unknown spaces and detects its surrounding areas and estimates its current location as well as creating a map. The steps of executing virtual SLAM with TurtleBot3 is shown in Website [T3S]. Once the robot finish exploring the indoor enviroment, an occupancy grid map (OCM) is generated (Figure 2.8).

2. **Measuring or estimating the pose of robot.** Pose is consists of position and orientation. The dead reckoning [DEA] is the most popular indoor pose estimation method for the robot. The amount of movement of robot is measured with the rotation of the wheel. However, the error between th calculated distance with wheel ratation and the actual travel distance increases over time. Therefore, the inertial measurement unit (IMU) sensor[Seo17] is used to measure triaxis angular velocities and triaxis acceleration to estimate the position of the mobile robot.

This inertial data can be used to compensate the error of position and orientation between calculated value and the actual value.

3. **Avoiding obstacles such as walls and furniture.** The laser-based distance sensor on robot are widely used in figuring out whether there are obstables including walls, furniture and other robots. The common laser-based distance sensor inludes LDS (Laser Distance Sensor), LDF (Laser Doppler flowmetry) and LiDAR (Light Detection And Ranging) and ultrasonic sensors and infrared distance sensors. The TurtleBot3 equips 360 Laser Distance Sensor LDS-01. The visualization of Laser data in Rviz is shown in Figure 2.2.

4. **Finding the optimal route calculation and driving.** It is important to find the optimized route to the distination. There are many algorithms that perform path searching and planning such as A* algorithm[ASE], potential field[POT], particle filte[PAR], and RRT (Rapidly-exploring Random Tree)[RRT].

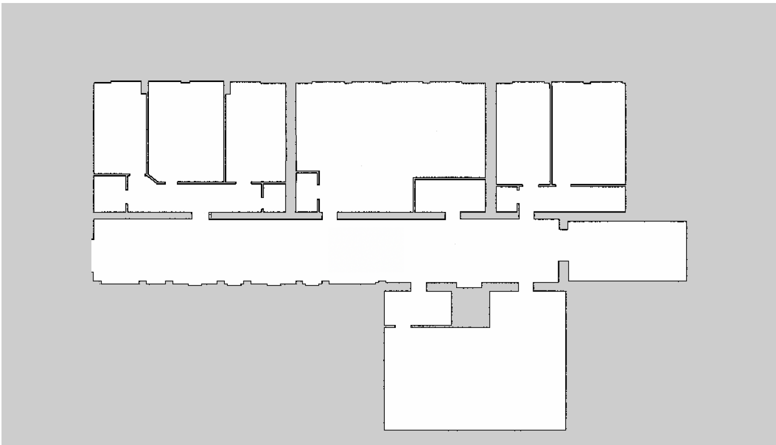


Figure 2.8: Occupancy Grid Map (OGM). White presents the free area in which robots can move, black presents the occupied area in which robots can not move, and gray is the unknown area.

2.4 Exist Task Scheduling Methods

So far, the background information such as tools and important concepts of ROS are introduced. This Chapter discusses some popular methods of task scheduling. Those task scheduling methods can be divided into centralized methods and distributed methods.

2.4.1 Centralized Method vs. Distributed Method

In the case of centralized method, a centralized schedule collects all task requests and uses resource utilization information to schedule tasks. The centralized method is easy to manager and faster to repair in case of failure. However, the autonomy of the robots in pure centralized method is limited becaued all robots only execute commands from centralized scheduler and not determine what tasks to do [NMMG17]. In addition, since the centralized scheduler must compute all resources and tasks, it has less scalability which becomes the bottleneck in large-size network [CSMV09]. Chapter 2.4.2 introduces a centralized constraint programming method.

In the case of distributed method, there are multiple distributed schedulers keep tracking the resource availability and use this information to perform task scheduling[CSMV09]. This method distributes the associated computation overhead, as a result it eliminates the bottleneck caused by the centralized scheduler and improves the scalability and realiability and scalability if the network. The challenge in distribute method is the coordination of distribute schedulers as well as the required control plane overhead [CSMV09]. Chapter 2.4.4 introduces a distributed auction method and Chapter 2.4.3 introduces a distributed capability-based method.

2.4.2 Centralized Constraint Programming Method

Booth proposed a multi-robot system that support elderly residents in a retirement home setting in [BMR+17]. The robots search for elderly residents in the enviroment in the morning, eliciting their availability and preferences for activities. The centralized scheduler then use constraint programming method to allocate these assistive activities over the day. Those problem-specific constrains includes robot energy consumption, activity priority, robot-user activity synchronization, user location, and user availability carlendar that identifies their busy intervals. Once this information is attained, the system allocate and schedule activities to robots for the day before executing the plan.

In addition, there are other centralized methods such as centralized mixed-integer programming [KSD13].

2.4.3 Distributed Capability-based Method

Kim proposed a distributed capability-based method [KM16]. This method is designed to allow robots to perform resonable decision-making when they can handle the new task as well as tasks in its queue. In this method, the scheduler sends a query to all robots to get a list of capability for the tasks. Then it selct one robot with the best capablility to delegate, and then send instruction and deadline to the selected robot. Besides, the robot capability to a task varies from [0,1]. This value determins how well the robot can handle the new task by deadline while taking care of current tasks.

2.4.4 Distributed Auction Method

When system perform a long-term task alloacation process, the communication link between costumer agent and robots may be disconnected. This may couse a conflict or failed assignment. Distributed methods are more suitable in this case as distribute the computation to individual agents [NMMG17]. Dong-Hyun Lee proposed a resource-oriented, distributed auction algorithm [LZK15]. The customer agents and robots with limited communication ranges construct an ad-hoc network tree. The customer agent becomes auctioneer and broadcasts an auction call to the task. The robots become bidders and submit their bid values to the customer agent. The bid values consider local information such as the tasks in robot task queue, robot's resource levels and estimated travel distance and time for multiple path. Since each path consists of different charging stations, the robot's resource levels after completing a task and estimated travel time depends on the path. After receiving all bid values, the agent assigns the task to the robot with the lowest bid value. This senario has many advantages. It is more efficient communication overhead and energy efficiency. To be more precise, it not only avoids unexpected battery drain while robot processing task, but also let robots maintain high capacities. In addition, in distributed method, robots don't need to broadcast its information such as current position and battery levels frequently. Therefore, compared to centralized methods, distributed method not only avoids unnecessary communication but also improves robot autonomy [SM07].

2.5 Exist Quantities of Cost Function

One of the most important steps when designing a multi-robot task scheduling algorithm is determin the costs of tasks. Jia summaries several physical quantities used in algorithm's cost in [JM13]. In their study it can be concluded that the most common used decision variables are estimated travel distance and time, as proposed in [LZK15]. Other kinds of decision variables involved are the number of traversals and energy consumed. In addition, Korsah proposed a comprehensive taxonnomy of multi-robot task scheduling problems that explicitly takes into consideration the issues of interrelated utilities and constraints. In this taxonnomy, tasks are distinguished by decomposability and multi-agent-allocatability [KSD13].

Chapter 3

Approach

A communication efficient task scheduling system is designed to help multiple robots handle various task. This system schedule task according to system resources, including system environment information, robot status and task specifications. Once this information is attained, the task scheduling system sends robot a set of task.

- **Robot.** Each robot is responsible for moving in 2-dimensional physical space as well as gathering measurement result from sensors. It has a rechargeable battery, and its level drops as robot moves and rotates.
- **Tasks.** Each task requires one or more robots to traverse a path in the workspace and carry out certain actions[GMS17].
- **Environment.** In this project, all robots are considered moving in an office environment that contains a corridor along the central x-axis and 16 rooms located around the corridor. The environment factors, such as room locations and occupancy possibilities help task scheduling.

3.1 Architecture Design

The architecture of the system consist of several parts: centralized pool, robot controller, navigation stack, charging station and system environment(Figure 3.1).

- **Centralized Pool.** A centralized pool consist of several modules: multi-robot task scheduling module, map information, database, execution and monitoring. The database stores dynamic indoor enviroment information such as measurement reuslt. The map information modules contain the static map information(Figure 4.2). The execution and monitoring module interacts with robots. The multi-robot task scheduling module schedule tasks to robots.

- **Robot Controller.** A robot controller contains several modules: local task queue, execution and robot action. The local task queue stores tasks that the robot needs to complete sequentially. The execution module receives commands from centralized pool and decides when and which task the robot should run. The robot action module run tasks in local task queue when receives decision from execute module and interacts with enviroment and its navigation stack.

- **Navigation stack.** The move_base node provides a ROS interface for configuring, running, and interacting with the navigation stack on a robot. It makes robot move to desired positions using the navigation stack. Its advantages include optionally performing recovery behaviors when the robot perceives itself as stuck[MOV].

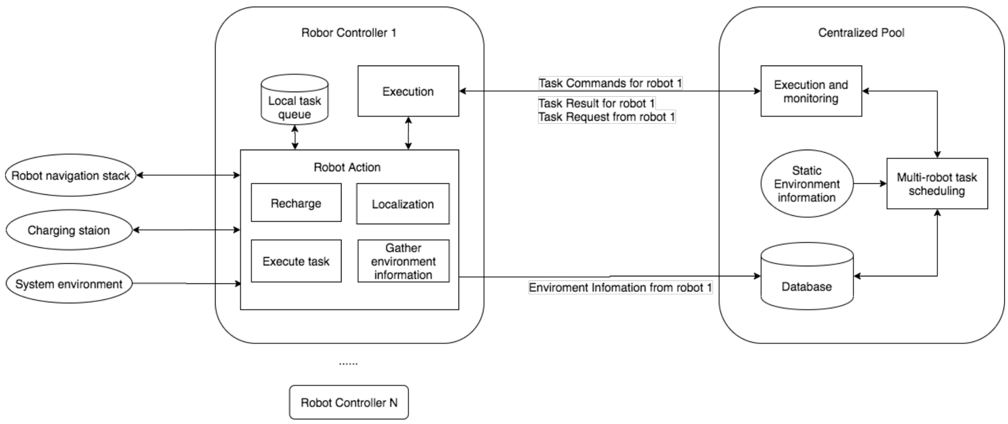


Figure 3.1: Multi-robot Task Scheduling Architecture

3.2 Environment and Gather Environment Infomation Approach

3.2.1 System enviroment

As is discussed in the previous section, the system environment is an office environment. The SLAM (Simultaneous Localization and Mapping) is a technique to draw a map by estimating current location in an arbitrary space [T3S]. Map (Figure 3.2) is created by SLAM.

Important areas and coordinates:

- **Rooms.** The enviroment is divided into regions that represent rooms in the facility (Figure 3.2). If the coordinates of a point are in a region, it can be judged that the point is located in the corresponding room.
- **Doors.** The positions of doors (Figure 3.3) are stored in database. There are used by a ROS door simulator node, which broadcasts positions and door status periodically. The broadcast messages are received and filtered by robots.
- **Charging Stations.** The positions of charging stations (Figure 3.3) are used by ROS charging station nodes. For details please refer to Chapter 4.3.3.

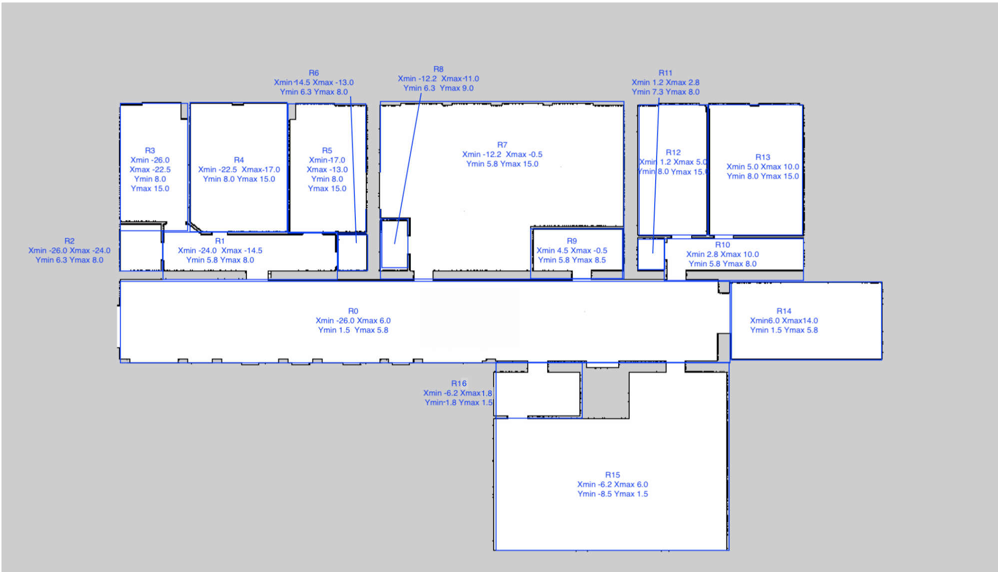


Figure 3.2: Room division

3.2.2 Gather Environment Infomation Approach

The goal of this project is to schedule tasks to robots based on the information gathered about room occupancy. However, current robot sensing technology, including robots equipped with external sensors, is not good enough to gather information of whole office enviroment satisfactorily. For example, compared with office enviroment, the field within sensing range is rather narrow. Therefore, use of external enviroment information sources is essential to bridge the local knowledge gap. In this project, since distributed sensor network cost higher than robot and its network connections consume much energy, the fixed sensors capable of short-distance communication are installed in the enviroment. They can share their environment information with robots, thus robots don't need to equip with numerous sensors [PNK+15]. Robots interact with sensors while moving in

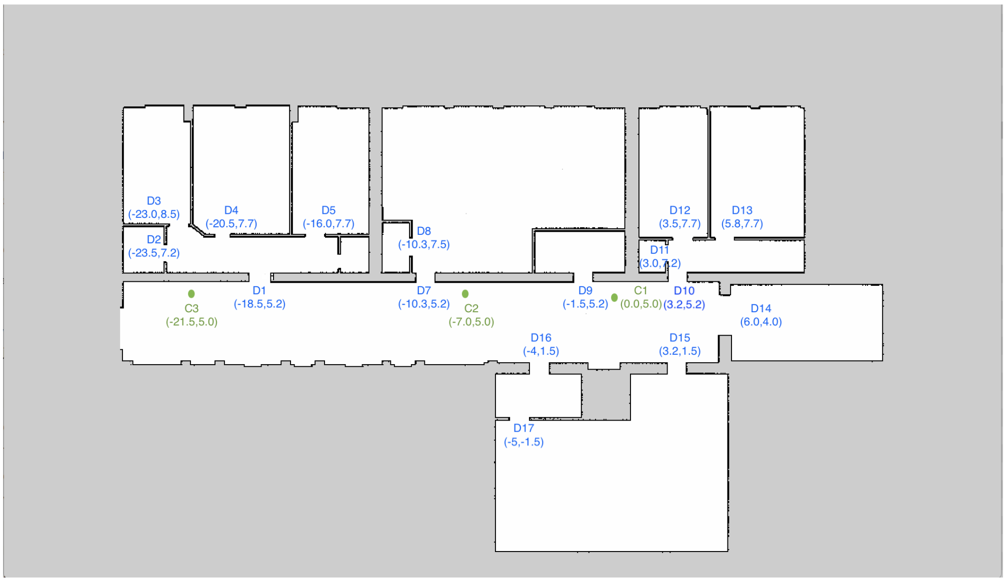


Figure 3.3: Positions of Charging Stations (C1-C3) and Doors (D1-D17)

the enviroment and report those enviroment information to centralized pool. The details about centralized pool storing and studying from enviroment information are discussed in section 4.3.

3.3 Tasks and Task Composition and Decomposition Approach

3.3.1 Task Specification

Robots should execute tasks in order to achieve the overall system goals: gather information and environmental information continuously for a long time and schedule tasks to robots based on the environmental information. Therefore, three task names are defined: "gather enviroment information task" asks a robot gathers enviroment information from sensors, "execute task" asks a robot moves to a point and "charging task" asks robot to refill its battery at charging station. The task specifications are stored in "task table" in database (Table 4.7).

3.3.2 Task Composition and Decomposition

Tasks can be distinguished to "simple tasks" and "Complex tasks". "Simple tasks" comprises a single action that can be performed by a single robot. A "Complex tasks" can be broken up or decomposed into multiple "small tasks". Those sub-tasks of a complex task need to be performed by the same robot. In this project, the centralized poll not only can create “simple tasks” according to task specifications (Table 4.7), but also can analyze the dependencies of “simple tasks” and form a dependency chain to compose “complex tasks”. The robot (robot controller) can decompose a “Complex task” to “simple tasks” and execute “small tasks” according to their dependencies.

3.4 Multi-robot Task Scheduling Approach

The multi-robot task scheduling module in the architecture should perform multi-robot task scheduling. The implementation of task scheduling is shown in Chapter 4.3. There are some general rules for multi-robot task scheduling.

1. When the battery of robot belows 10%, a charging task will be created and sent to robot.
2. When the battery of robot aboves 10%, firstly, “simple exeucte tasks” according to the task table in database are created. Secondly, “complex execute tasks” will be composed and one of them will be selcted and sent to robot. To ensure consistency, a “complex task” composed by only one “simple task” is allowed.
3. If there are no “execute tasks” in database or after “execute tasks scheduling”, the cost of tasks exceeds the threshold, a “gather environment task” will be created and sent to robot.

3.4.1 Execute Task

As discussed in Chapter 3.4, one of the “complex execute tasks” should be selected for requesting robot. In order to select an “execute task”, the decision variables and Equation 3.1 are used to calculate the cost. The “complex execute tasks” with the lowest cost will be selected.

W: Weight

n: Number of doors

$$\text{Cost}_{\text{Large execute task}} = \frac{W_{\text{battery}} \times \text{Battery consumtion}}{n} + W_{\text{waiting}} \times \text{waiting time} + W_{\text{possibility}} \times \prod_{i=1}^{n} \text{Door open possibility} + W_{\text{priority}} \times \text{Priority} \tag{3.1}$$

Decision variables

- **Task Priority.** The priority is discussed Chapter 4.2.
- **Product of Door Open Possibility.** The product of open possibilities of doors on trajectory: All doors that the robot will pass through when moving from its location to the target point. An example of "measurement result" table is shown in Table 4.8, an example of "open possibility" table is shown in Table 4.9.
- **Waiting Time.** The waiting time is the difference between the current simulation time and start time of the first task to be executed. $T_{waiting} = T_{first\_task} - T_{now}$
- **Battery Consumption.** The Battery Consumption is related to robot trajectory. For a Large "execute task" that contains n simple task, Equation 3.2 can be used to calculate battery consumption. The centralized pool will send the task with the lowest cost to this robot.

B:Battery consumption

W: Weight

m: Number of waypoint

n: Number of simple task

$$
\begin{aligned}
&B_{\text{complex task}} = \sum_{\text{task}_1}^{\text{task}_n} B_{\text{trajectory}} \\
&= \sum_{t=\text{task}_1}^{\text{task}_n} \sum_{\text{waypoint}_1}^{\text{waypoint}_m} [W_{\text{position}} \times \text{position variation} + W_{\text{angle}} \times \text{angle variation}] \\
&= \sum_{t=\text{task}_1}^{\text{task}_n} \sum_{p=\text{waypoint}_1}^{\text{waypoint}_m} [W_{\text{position}} \times \sqrt{(x_p - x_{p-1})^2 + (y_p - y_{p-1})^2} \\
&+ W_{\text{angle}} \times 2 \times \arccos(w_p)]
\end{aligned} \tag{3.2}
$$

3.4.2 Environment Task

As is discussed in section 3.4, once there are no suitable tasks in centralized pool, the task scheduling module should create a "gather environment information task" to gather more measurement results and further more improve the accuracy of "open possibilities" table. To create a "gather environment information task", Equation 3.3 and following decision variables are used to calculate the costs of doors. A "gather environment information task" to the door with the lowest cost will be created.

W: Weight

n: Number of doors on trajectory

$$\text{Cost}_{\text{door}} = \frac{W_{\text{battery}} \times \text{Battery consumtion}}{n} + W_{\text{time}} \times (T_{\text{last update}} - T_{\text{now}}) + W_{\text{possibility}} \times \prod_{i=1}^{n} \text{Door open possibility} \tag{3.3}$$

Decision variables

- **Door Last Update Time.** The latest timestamp when the door is measured.
- **Product of Door Open Possibility.** The product of open possibilities of doors on trajectory: All doors that the robot will pass through when moving from its location to the front of the target door.
- **Battery Consumption.** The battery consumption is related to the trajectory from robot to the front of the door. Equation 3.2 can be used to calculate battery consumption.

3.4.3 Charging Task

As is discussed in section 3.4, once a robot sends task request to the centralized pool, the centralized pool should figure out whether this robot need charging, if yes it should create a "charging task" for requesting robot. To create a "charging task", Equation 3.3 and following decision variables are used to calculate the costs of charging station. A "charging task" to the charging station with the lowest cost will be created.

W: Weight

$$\text{Cost}_{\text{charging station}} = \frac{W_{\text{battery}} \times \text{battery consumtion}}{n} + W_{\text{time}} \times T_{\text{remain}} \tag{3.4}$$

Decision variables

- **Remain Time.** It describes how long will a charging station be free.
- **Battery Consumption.** Similar to "execute task" scheduling, the battery consumption is related to the trajectory from robot to the charging station. Equation 3.2 can be used to calculate battery consumption.

Chapter 4

Implementation

4.1 Communication Protocols

Centralized pool, robots, charging stations and sensors need to share information with each other. To improve the communication efficiency, communication protocols are designed.

4.1.1 Message about Measurement

When a robot passes by a door, it should receive messages from a sensor. In this project, we use a ROS node "sensor simulator" to simulate door sensors (Chapter 5.2) to publish instant measurement result (Table 4.1).

These Communication protocols save unnecessary communication cost by avoiding keep tracking the current position, availability and states of all robots (Figure 4.1).

Door ID	Position	Timestamp	Measurement Result
1	(-18.5,5.2)	2020-06-01 9:00:02	Door opened

Table 4.1: Measurement Message Format and Example

4.1.2 Message about Task

There are some basic requirements for communication between robot and centralized pool: firstly, robot should initiate the communication once task queue becomes empty. Secondly, robot should forward sensor data to centralized pool immediately after interacting with snesors. Four types of message are defined: (1)Task request message(Table4.2); (2) Task goal messages(Table 4.3); (3) Task feedback message (Table 4.4); (4) Task result message (Table 4.5).

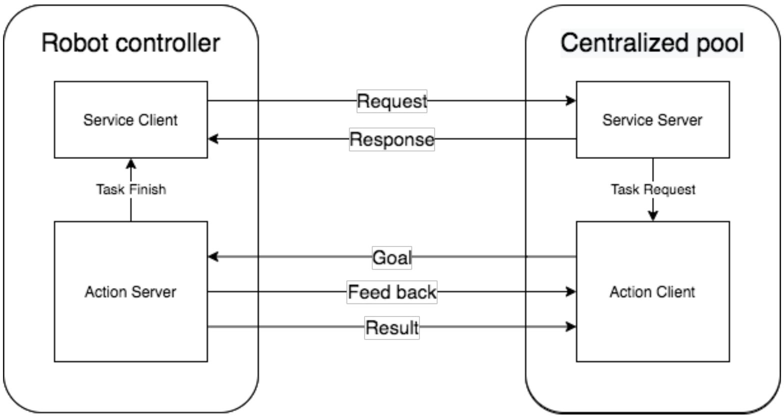


Figure 4.1: Communication between Robot and Centralized Pool

Battery Level	Position	Robot ID
93	(2,4)	1

Table 4.2: Request Message Format and Example

4.1.3 Message about Charging

When a robot arrives charging station's position, it sends a message to Charging station(Figure 4.6). The details of charging station is discussed in Chapter 4.3.3.

4.2 Database

The centralized pool keep environment information in database to make decisions. The structure of database is shown in Figure 4.2.

Table tasks

- **Column Task ID.** A unique task identification.

Task ID-[]	Task type	Target ID	Goal[]
1	Gather Environment Info	9	(-1.5,5.2) 2020-06-01 9:00:00
[3,4]	Execute task	21, 22	(-24.0,12.0), 2020-06-01 9:02:00 (-21.0,12.0) 2020-06-01 9:02:00
5	Charging	17	(0.0,5.0), 2020-06-01 9:04:00

Table 4.3: Action Goal Message Format and Example

Robot ID	Door ID	Measurement time	Measurement result
1	3	2020-06-01 9:00:03	Door open

Table 4.4: Action Feedback Message Format and Example

Task ID	Task type	Result
1	Gather Environment Info	Success

Table 4.5: Action Result Message Format and Example

- **Column Task Name.** Tasks names are "gather enviroment information task ", "execute task" or "charging task".
- **Column Start Time.** The start time refers to when the robot should move towards the target. A starting time is given when the task is created. This time can be a time in the future or empty (no time limit).
- **Column Finish Time.** The default value is empty. When the centralized pool receives task result, this column will be updated to the time when the centralized pool received the result.
- **Column Target ID.** Targets include doors, points and charging stations. When a robot run a "gather environment information task", it moves to the front of a door and interact with a sensor in the door position without entering the door. When robot run an "execute task", the robot moves to a given point ether in corridor or in the room. When robot run a "charging task", the robot moves to a charging station and interact with this charging station.
- **Column Robot ID.** A unique robot identification.
- **Column Priority.** Task priorities allow user to easily prioritize tasks to clearly plan what to do next. The "charging tasks" are given the highest priority of 5. The "gather environment information tasks" are given the lowest priority of 1. The "execute tasks" has priority between 2-4 in the "created" status. Once this task failed once, its priority will be increased by 1 until it exceeds the maximum and is marked as "Failed" (Figure 4.4).

Robot ID	Battery Level
1	93

Table 4.6: Message to Charging Station

- **Column Task Status.** Task status are “Created”, “Succeeded”, “Failed”, “To rerun”, “Error”. The difference between task status is discussed in Figure 4.4
- **Column Dependency.** If task B has a dependency of task A, task A needs to be preceded by tasks B. Those dependent tasks should be composed in the centralized pool.
- **Column Description.** The describtion of a succedded task is “succedded”. The describtion of a failed task is its failure reason.

Task ID	Task Type	Start Time	Target ID	Robot ID	Priority	Status	dependency	Finish Time	Description
1	Charging task	2020-06-01 9:00:00	18	1	5	RanToCompletion	0	2020-06-01 9:00:20	Succeeded
2	Execute task	2020-06-01 9:00:50	22	2	2	RanToCompletion	0	2020-06-01 9:01:20	Succeeded
3	Gather environment information task	2020-06-01 9:02:00	2	2	1	Running	0	2020-06-01 9:02:40	Succeeded

Table 4.7: Task Table in Database

Table measurements

- **Column Door ID.** Unique identification of the door.
- **Column Door Status.** Value 0 represent door closed. Value 1 represent door opened.
- **Column Date Time.** Measuring time.

Door ID	Door Status	Date Time
9	1	2020-06-01 09:05:39
1	0	2020-06-01 09:05:49
7	1	2020-06-01 09:05:49
9	1	2020-06-01 09:05:49
1	1	2020-06-01 09:05:59
7	1	2020-06-01 09:05:59
9	1	2020-06-01 09:05:59
7	1	2020-06-01 09:06:09
16	0	2020-06-01 09:06:29
7	1	2020-06-01 09:06:39
16	1	2020-06-01 09:06:39
9	1	2020-06-01 09:06:49
8	0	2020-06-01 09:06:59
...	...	...

Table 4.8: Table measurements

Table door open possibilities

- **Column Door ID.** Unique identification of the door.
- **Column Day of Week** a weekday.
- **Column Start Time and End Time** a time slot between start time and end time.
- **Column Initialized Open Possibility** Predefined value to used to simulate door sensors (Chapter 5.2).
- **Column Open Possibility Statistic** Statistics of measurement result in the weekday and time slot.

Door ID	Day Of Week	Start Time	End Time	Initialized Open Possibility	Open Possibility Statistic
1	2	9:00:00	9:59:59	0.90	0.85
1	2	10:00:00	10:59:59	0.90	0.92
1	2	11:00:00	11:59:59	0.10	0.05
...	...	...	...	...	...

Table 4.9: Door Open Possibility

Table doors

- **Column Door ID** Unique identification of door.
- **Column Last Update** The timestamp of last measurement result on the door.
- **Column Is Used** Value 1 represent at least one other robot is moving to this door. Value 0 represents no robot is moving to this door.

Door ID	Last Update	Is Used
1	2020-06-01 15:15:26	0
2	2020-06-01 15:15:06	0
3	2020-06-01 15:12:36	0
4	2020-06-01 15:15:16	0
5	2020-06-01 15:11:46	0
6	2020-06-01 15:11:36	1
7	2020-06-01 15:14:26	0
...	...	...

Table 4.10: Doors Table

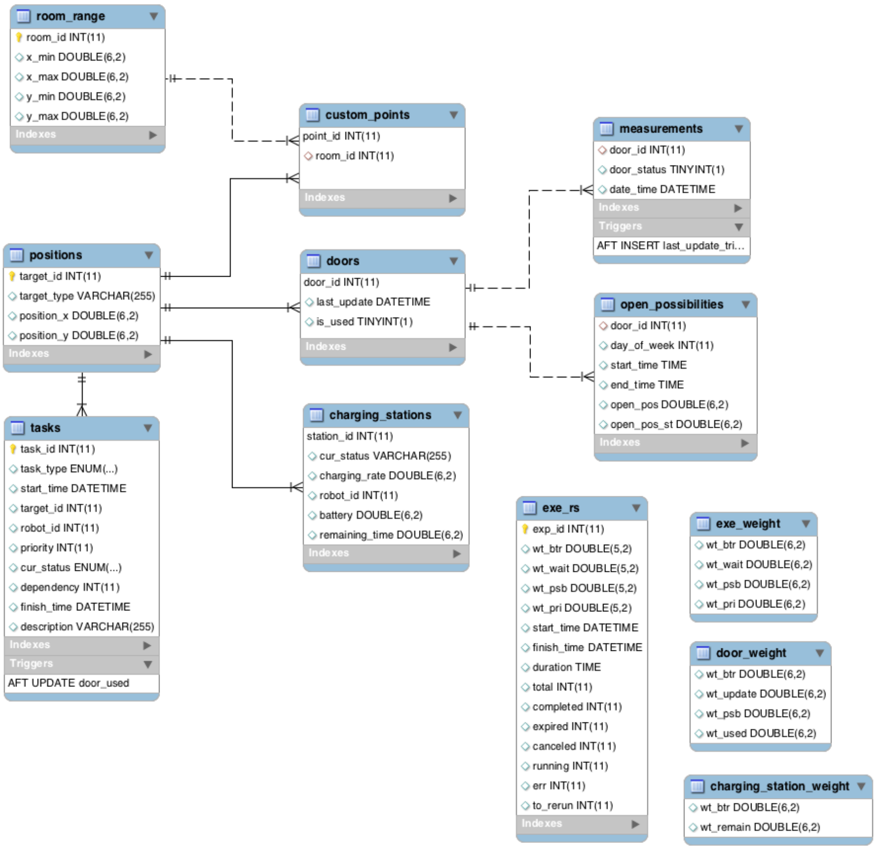


Figure 4.2: Database Entity Relationship Diagram

4.3 Procedure

As stated in the Chapter 3, the goal of task scheduling is finishing all tasks as soon as possible while keep the cost as low as possible. The task assignment and execution has at two level. [GMS17] the task and the path planner solves a planning problem. It takes and occupancy grid, a specific robot and a set of task specifications, and generates trajectories for each task under the assumption that there are no dynamic obstacles (include other robots). According to those trajectories and task specifications, the task with the lowest cost will be assigned to robot. At the dynamic level, after each robot receive a task, it runs a navigation stack to execute this task stepwise. Each robot computes a local trajectory but takes into account dynamic obstacles. The process of the robot task scheduling system is as follows.

4.3.1 Centralized Pool

Handle task Request With robot status such as positions and available battery provided by robot, the multi-robot task scheduling module in the architecture should perform multi-robot task scheduling. When the centralized pool receives a task request (Table 4.2) from robot, the multi-robot task scheduling module in the architecture. The implementation of task scheduling is shown in Figure 4.3.

Handle Task Feedback. When the centralized pool receives a task feedback (Table 4.4) that contains a new measurement result from robot, it will add a record in "measurement table" and update "open possibilities table" in database (Table 4.9).

Handle Task Result. When the centralized pool receives a task result (Table 4.5), it updates status column in "tasks" table in database. In order to make the robot complete the task as much as possible, everytime when an "execute task" failed, its start time will be delayed and its priority will be increased by 1 until it exceeds the maximum and is marked as "Failed" (Figure 4.4).

4.3.2 Robot

Robot Process Tasks When the task queue(Figure 3.1) in a robot is empty, the robot requests a new task. If the robot gets a "charging task", it will move to the position of charging staion(Figure 3.3) and interact with charging station node (Chapter 4.3.3). When a robot gets an "execute task" which is a complex task, it will move to all goals in order. When a robot gets a "gather environment information" task, it will move to the door's position. During task processing, the timer checks periodically the status of

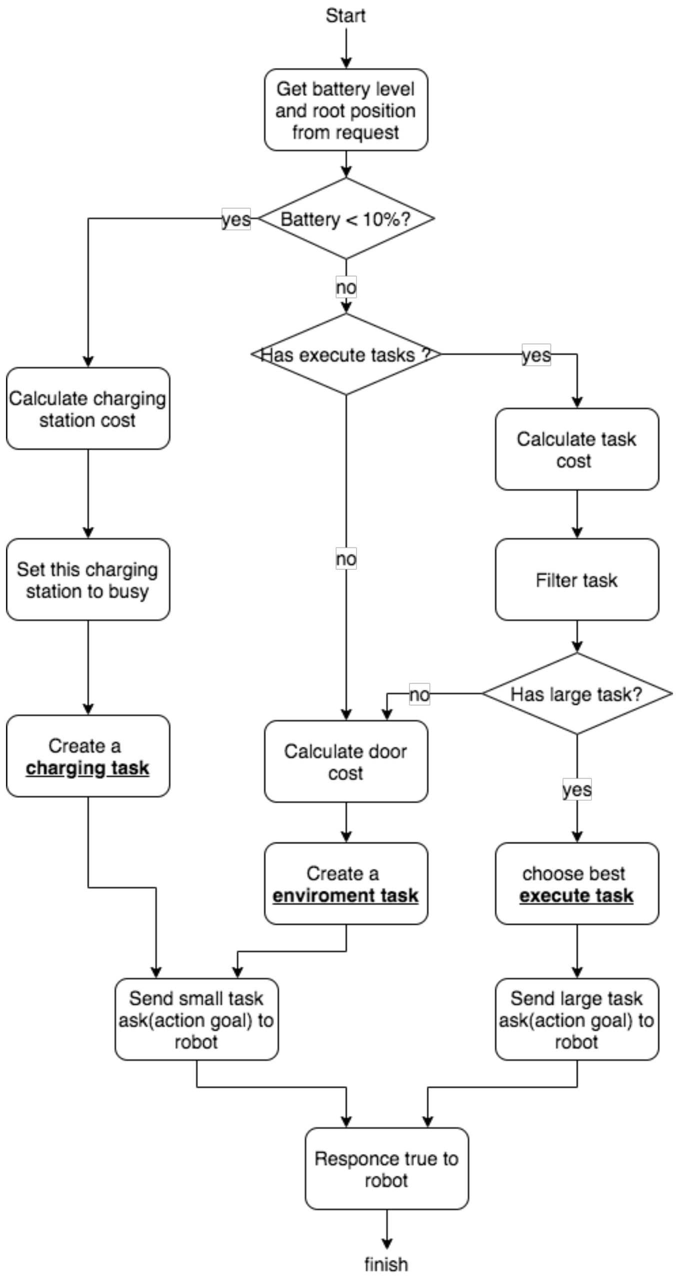


Figure 4.3: Centralized Pool Task Allocation

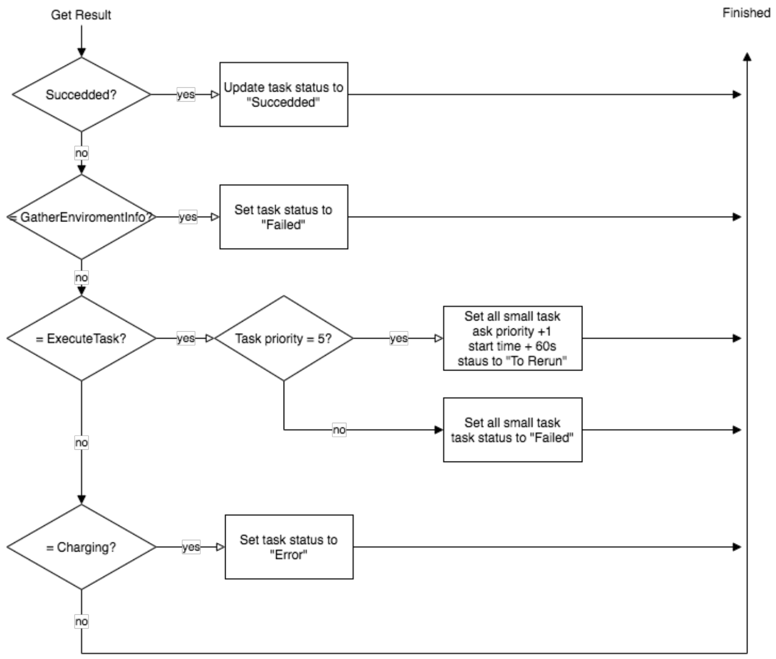


Figure 4.4: Centralized Pool Handle Task Result

Task Status Explanation

- **Succedded.** Robot successfully moved to the goal position and complted the task.
- **Error.** An error that cannot be corrected by itself occurred and the system requires manual restart. For example, a robot failed charging.
- **Failed.** A Task failed. Reasons of task failure includes: The robot was not able to move to goal positions or process robot action(3.1).
- **To rerun.** If an "execute task" failed, its priority was increased. The task is marked as "to rerun task" for a future scheduling by task scheduling module.

navigation stack. If any errors occurs, the robot send a “failed” result with description to the centralized pool. When all tasks are complted without error, the robot will send “Succedded” result to the centralized pool.

Robot Handle Messages While a robot is processing a task, it listens to door sensors and forwards measurement result to the centralized pool. Besides messages from sensor, it also receives messages from “move_base” node. The details of robot message handling is shown in Figure 4.7.

4.3.3 Charging Station

The charging station consists of a charging station node and “charging station” table in database (Table 4.2).

A charging station has four states : “Free”, “Charging” and “Charging finished”. Its initial state is “Free”. When a robot arrives the charging station, it will start interacting with charging station node (Figure 4.9). Once the charging station receives robot information, its state will be changed to “Charging” and its “battery level” will be increased and its “remaining time” will be decreased (Figure 4.8). Once finishing charging, its status will be set to “Charging finished”. When robot leaves charging station, its status will be set to “Free”.

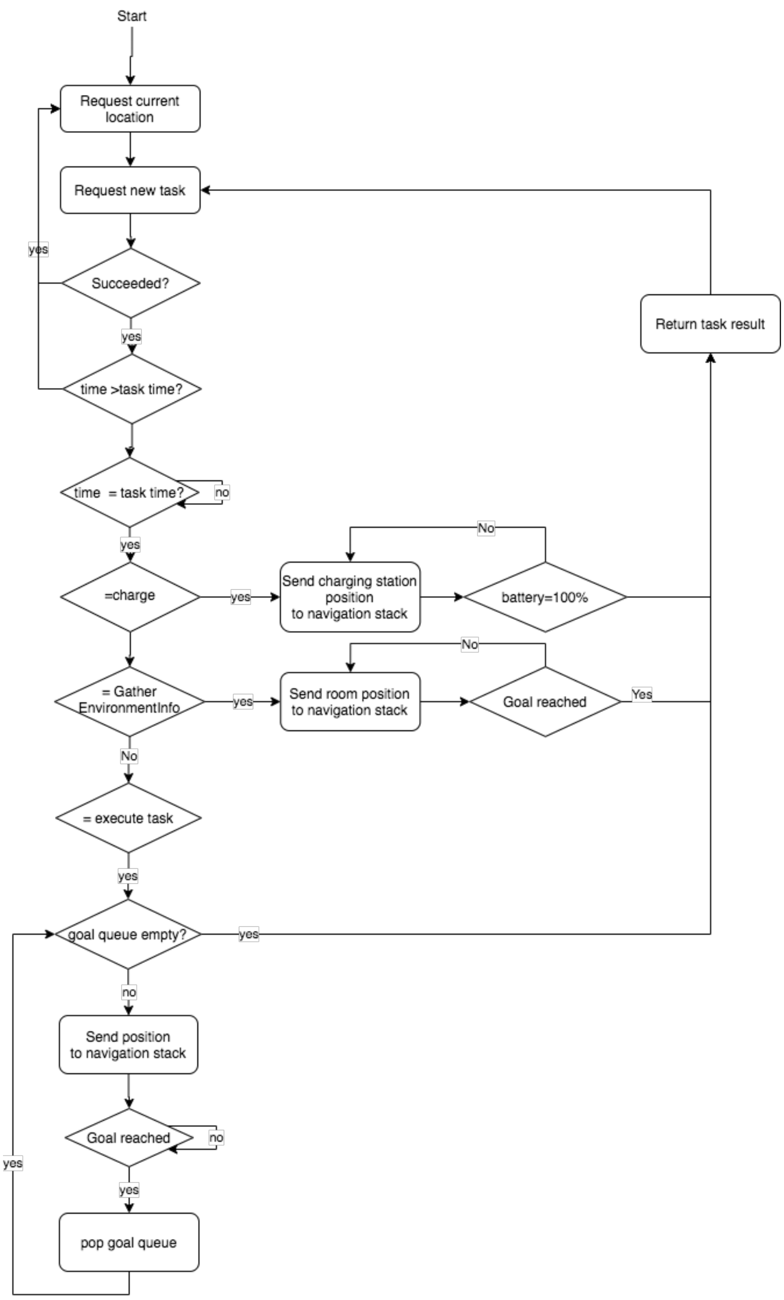


Figure 4.5: Robot Process Task

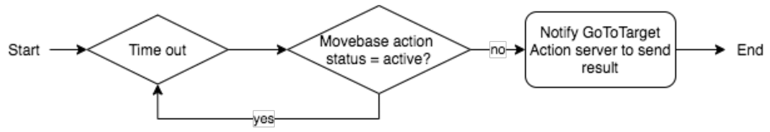


Figure 4.6: Robot Timer

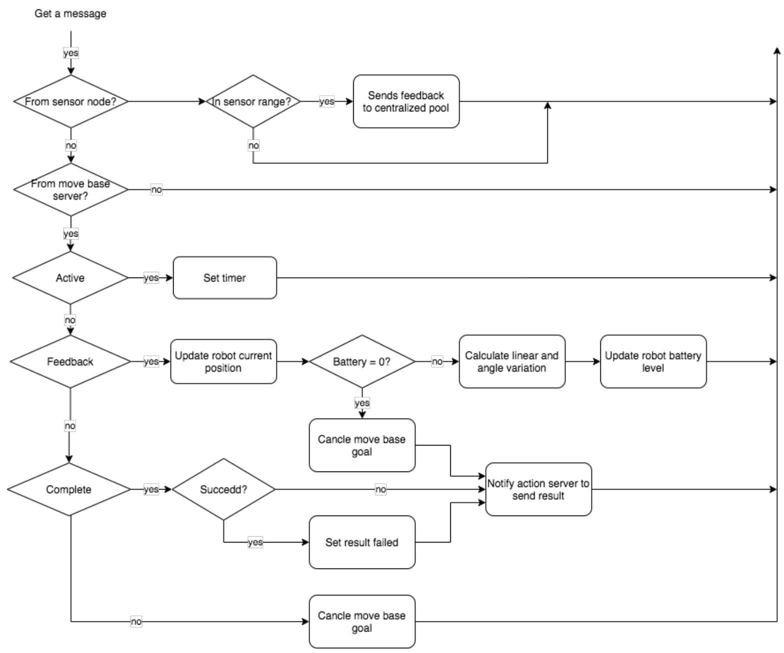


Figure 4.7: Robot Handle Message

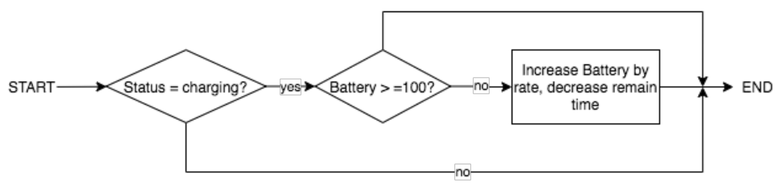


Figure 4.8: Charging Station Scheduled Charging Event in Database

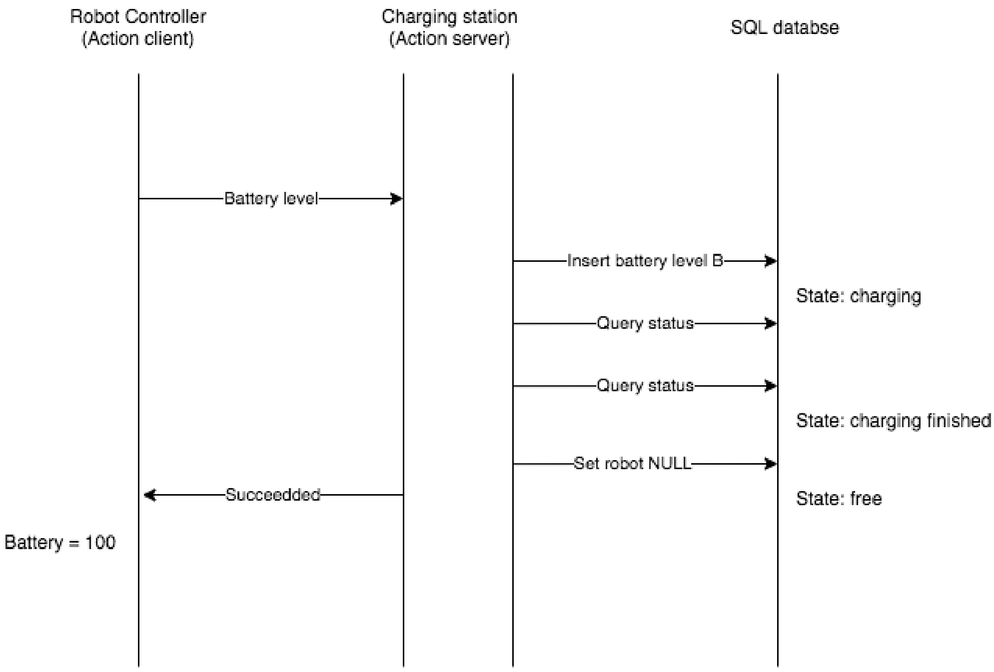


Figure 4.9: Charging Station Message

Chapter 5

Evaluation

In the system, cost functions are importent because an accurate cost function can improve the efficiency of task scheduling. Followings are goals of experiments:

- Evaluate the need of decision variables.
- Find the best weight combinations in cost functions.
- Evaluate the hypothesis that the more robots, the faster to complete a task set, the higher the completion rate.

5.1 Experiment Setup

1. Run Gazebo and load office world model (Figure 5.1).
2. Configure word property. For example, the simulation time were configured as "2020-06-01 9:00:00".
3. Use SLAM [T3S] to create a map for the office model(Figure 5.2).
4. Start SQL server and Charging event in database 4.8.
5. Start ROS nodes including "centralized pool", "robot controller", "sensor simulatior" and "charging station".

5.2 Sensor Simulator

Sensor Message Publish In this project, a sensor simulator node is used to publish measurement results. The door simulator publish one messages for each door periodically to a ROS topic "sensor data". The message contains door id, sensor position and door status(Table 4.1.1). The door status are created according to open possibilities table (Table 4.9).

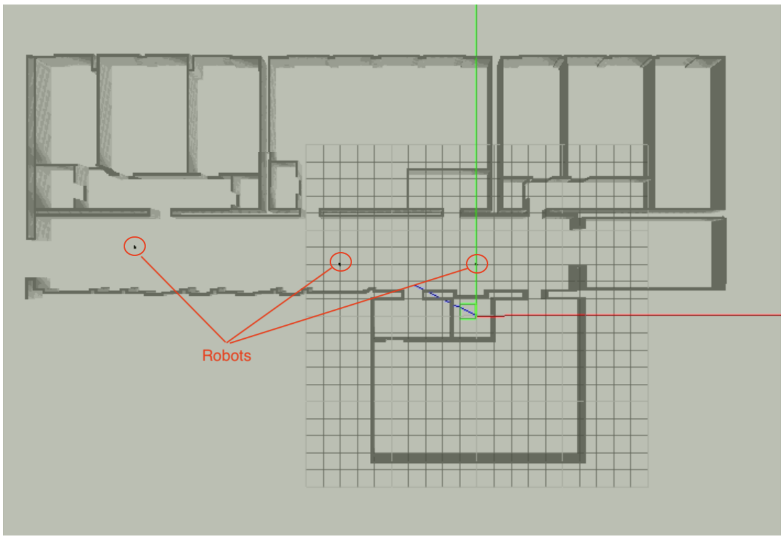


Figure 5.1: Gazebo Simulation

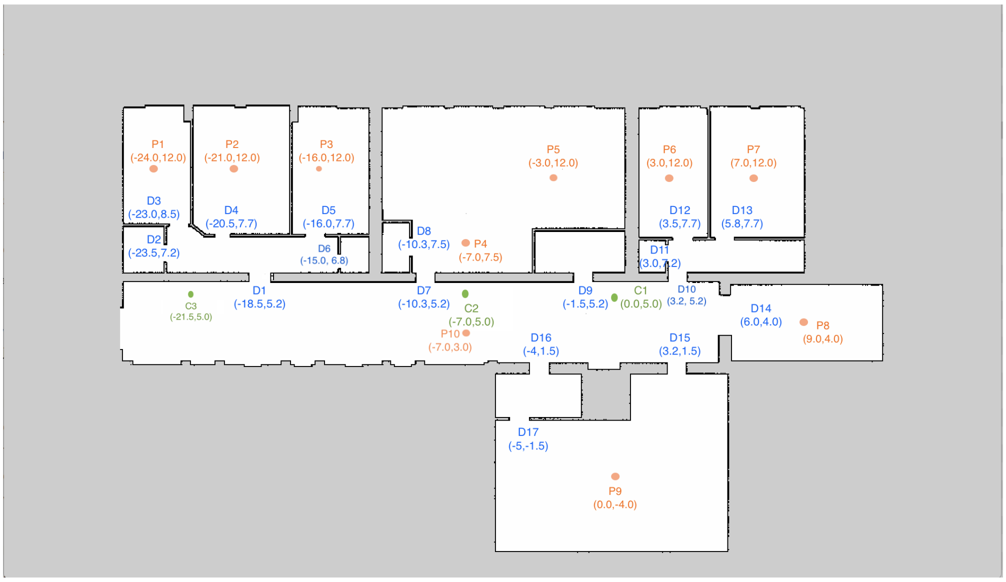


Figure 5.2: Experiment Map

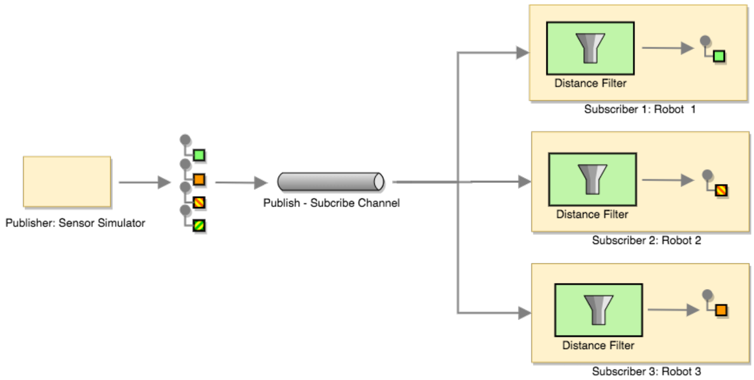


Figure 5.3: Sensor Simulator

Door Status Generation For example, on monday (day of week is 2) at simulation time "10:30:00" (on timeslot 10:00:00-10:59:59), in 80% possibility a "door open" is generated and in 20% a "door closed" is generated (Initialized Open Possibility is 0.80).

Sensor Message Subscribe The process of sensor simulation is shown in Figure 5.2. The robots (robot controller nodes) subscribe the same ROS topic "sensor data". Every time the sensor simulator sends a message, all robots will receive this message at the same time. Their distance filters filter sensor messages with position outside the comminication range and keep sensor messages within the communication range. With this process, the sensor simulatior sends instant measurement result to robots within communication range. However, if the system is applied to the real world, instead of sending instant measurement result, the real world sensor could send a record with history measurement.

5.3 Enviroment Task Evaluation Experiment

5.3.1 Experiment: Use Enumeration Method to Find the Best Weight Combinations

Experiment Introduction To find the best weight combinations, 30 experiemnt are created with $W_{\text{battery consumption}} \in \{0, 1, 5, 10\}, W_{\text{update}} \in \{-10, -5, -1\}, W_{\text{possibility}} \in \{-10, -5, -1, 0\}$. In addition, the conditions $W_{\text{update}} = 0$ was not testable, because during the experiment centralized pool generate the same "gather enviroment information tasks“, which let robots not moving and measuring their nearest door. Experiment Duration T = 10 min.

Sampling Design When simulation started, robots ran charging task (Task 1-3) at their charging station and started charging (Figure 4.8). For example, robot 1 charged at charging staion 1, robot 2 charged at charging staion 2, robot 3 charged at charging staion 3. When robot fully charged, the first experiment started (Figure 5.4). After that, the task scheduling module in centralized pool created a “gather enviroment information task” to each robot requested a task. After a constant experiment duration T, experiments were finished and robots went to their charging stations. These rules ensured that 1) Each Robot always started at an initial position. 2) Robots would not shut down because of power exhaustion. 3) Robots spent the same time gathering enviroment information.

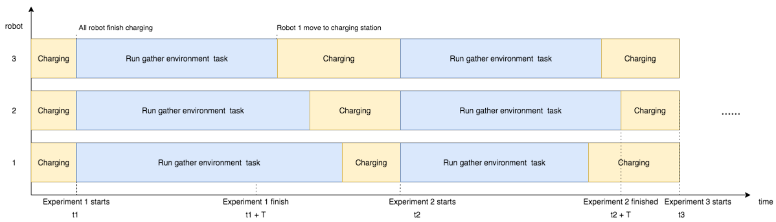


Figure 5.4: Enviroment Task Experiment Timeline

Analysis design The experiment start time was when all robot finshed charging and started request task (Figure 5.4). The experiment duraion is a constant value T. The experiment finish time $T_{\text{finish time}} = T_{\text{start time}} + T$. Some important factors are evaluated.

- **Last update.** The “last update” factor is the time difference between experiment start time and minial value in “last update” column in door table (Table 4.10) when an experiment finished. For example, “last update” factor in experiment 1 (Figure 5.5) is “00:02:57”. It means that the door in the worst case not be measured since 2 minites 57 seconds after experiment start.

- **Average Update Interal** The “Average update Interal” means the average interval of door update. For example, “Average Update Interal” factor in experiment 1 (Figure 5.5) is “00:01:00”. It means that on average, every door is updated every minute.

- **Succedded task.** The “Number of task” factor means the number of succeeded “gather enviroment information” task.

Experiemnt Result Figure 5.5 represent the experiemnt result.

Experiment Analysis As shown in experiment result, all "Minial Last Update" value is from 1 min to 5 min, which is much less than experiment duraion (10 min). It means some doors were not timely updated information. One possible reason is that the velocity of robot is small (about 0.2 per second), the experiemnt duraion (10min) was not long enough to let three robot pass to all doors. Another possible reason is that the robot's route is partially duplicated. For example, when system started, robot 1 was in charging station 1 and got a task to door 3, while robot 2 was in charging station 2 and got a task to door 4. As shown in Figure 3.3, these two route are partially duplicated, both of them pass through door 1 and entered room 1 (Figure 3.2). The idealy solution was to give both task to one robot.

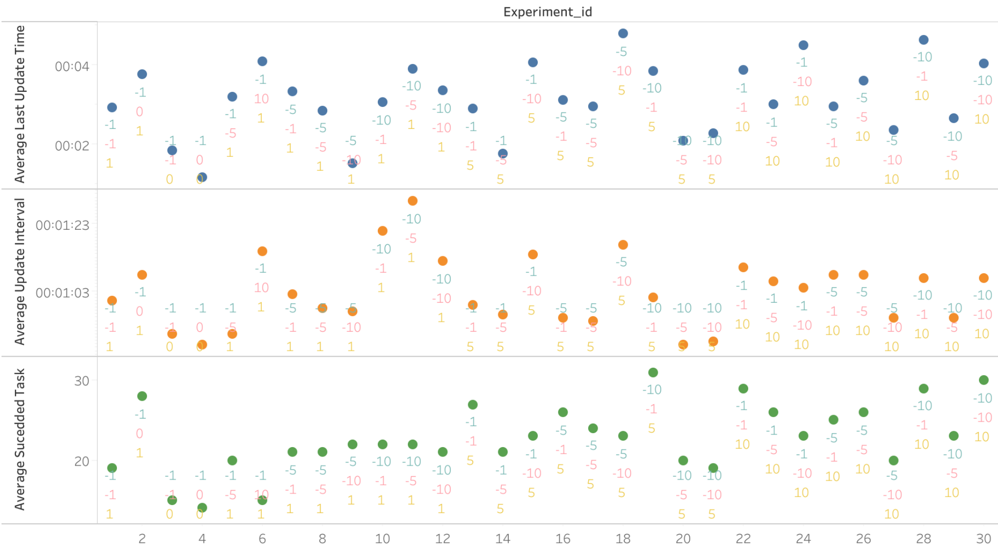


Figure 5.5: Experiment: Use enumeration method to find best weight combination Green: W_{battery} Pink: $W_{\text{possibility}}$ Yellow: W_{update}

5.3.2 Experiment: Use Analysis to Find the Best Weight Combinations

Experiment Introduction As is discussed in last experiemnt (Chapter 5.3.1), the experiment time was too short to allow the robot to explore each door. Therefore, the experiemnt duraion in this experiemnt was increased to 20 min. Firstly, we shuold find the best weight combination for "weight battery" and "weight last update", then find the third weight "weight possibility". Finally, the best combination of weight will be concluded. According to cost table (Figure 5.6), the update time value has a minial value of 6.986 and a maximal value of 336.986 (column 3), which are much larger than battery comsumption (column 2) and product of door open possibilities (column 4). Therefore, 18 experiments are created with $W_{\text{battery}} = 0.1$ and $W_{\text{battery}} \in \{1, 5, 10, 15, 20, 25\}$.

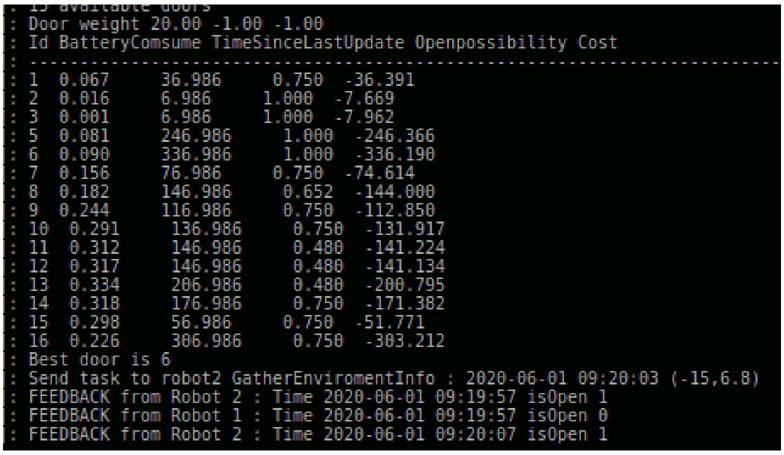
Sampling Design When simulation started, robots ran charging task (Task 1-3) at their charging station and started charging (Figure 4.8). For example, robot 1 charged at charging staion 1, robot 2 charged at charging staion 2, robot 3 charged at charging staion 3. When robot fully charged, the first experiment started (Figure 5.4). After that, the task scheduling module in centralized pool created a "gather enviroment information task" to each robot requested a task. After a constant experiment duration T, experiments were finished and robots went to their charging stations. These rules ensured that 1) Each Robot always started at an initial position. 2) Robots would not shut down because of power exhaustion. 3) Robots spent the same time gathering enviroment information.

Analysis design The experiment start time was when all robot finshed charging and started request task (Figure 5.4). The experiment duraion is a constant value T. The experiment finish time $T_{\text{finish time}} = T_{\text{start time}} + T$. Some important factors are evaluated.

- **Last update.** The "last update" factor is the time difference between experiment start time and minial value in "last update" column in door table (Table 4.10) when an experiment finished. For example, "last update" factor in experiment 1 (Figure 5.5) is "00:02:57". It means that the door in the worst case not be measured since 2 minites 57 seconds after experiment start.
- **Average Update Interal** The "Average update Interal" means the average interval of door update. For example, "Average Update interal" factor in experiment 1 is "00:01:00". It means that on average, every door is updated every 1 minute.
- **Succedded task.** The "Number of task" factor means the number of succeeded "gather enviroment information" task.

Exapriment Result The experiemnt results are shown in Figure 5.7 and Figure 5.8

Exapriment Analysis As shown in experiemnt result Figure 5.7, "weight battery" increased from 0 to 25, the "average succedded experiment task number" slighly increased from 45 to 50, but "average update interval" were floated in a small range around "00:01:35". Especially, as "weight battery" increased from 0 to 15, "average last update" showed an upward trend, and as "weight battery" increased from 15 to 25, "average last update" showed a downward trend, therefore "weight battery" 15 and "weight update" 0.1 was the best combination. This combination was used in next experiemnt set (Figure 5.8). From this experiemnt set, it was concluded that "weight battery" 15, "weight update" 0.1, "weight possibility" 0.1 and "weight battery" 15, "weight update" 0.1, "weight possibility" -0.3 were two best weight combinations for enviroment cost function.



```
: 15 available doors
: Door weight 20.00 -1.00 -1.00
: Id BatteryComsume TimeSinceLastUpdate Openpossibility Cost
: ------------------------------------------------------------------
: 1  0.067     36.986     0.750  -36.391
: 2  0.016     6.986     1.000  -7.669
: 3  0.001     6.986     1.000  -7.962
: 5  0.081     246.986     1.000  -246.366
: 6  0.090     336.986     1.000  -336.190
: 7  0.156     76.986     0.750  -74.614
: 8  0.182     146.986     0.652  -144.000
: 9  0.244     116.986     0.750  -112.850
: 10  0.291     136.986     0.750  -131.917
: 11  0.312     146.986     0.480  -141.224
: 12  0.317     146.986     0.480  -141.134
: 13  0.334     206.986     0.480  -200.795
: 14  0.318     176.986     0.750  -171.382
: 15  0.298     56.986     0.750  -51.771
: 16  0.226     306.986     0.750  -303.212
: Best door is 6
: Send task to robot2 GatherEnviromentInfo : 2020-06-01 09:20:03 (-15,6.8)
: FEEDBACK from Robot 2 : Time 2020-06-01 09:19:57 isOpen 1
: FEEDBACK from Robot 1 : Time 2020-06-01 09:19:57 isOpen 0
: FEEDBACK from Robot 2 : Time 2020-06-01 09:20:07 isOpen 1
```

Figure 5.6: Centralized Pool Cost Table

5.4 Execute Task Evaluation Experiment

5.4.1 Experiment: Impact of Decision Variables

Experiment Introduction This set of experiments evaluated the need of four decision variables: battery consumption, waiting time, product of door open possibility and priority.

Sampling design Table 4.7 shows "task table" in database. At this point, robots finished task 4-21 in experiment 1 and finished 21-39 in experiment 2. When simulation started, 3 robots ran charging task (Task 1-3) and moved to corresponding charging station and started charging (Figure 4.8). For example, robot 1 charged at charging staion 1, robot 2 charged at charging staion 2, robot 3 charged at charging staion 3. When robot fully charged, the first experiment started (Figure 5.9), and 15 "execute tasks" and 3 "charging tasks" were created (Task 4-21). Especially, in the same experiment, the task interal were the same (Table 4.7). In different experiment, the corresponding task had the same goal positions (Task 4 and 21, Task 4 and 22). When all robot finished tasks, the experiment would be finished and robots charged at previous charging station(Task 19-21). These rules ensured that robot not only started at their initial positions but also processed same task set and not shuted down because of power exhaustion.

Analysis design As is discussed in last paragraph, in each experiemnts, robots need to finish 15 "execute tasks" and 3 "charging tasks". The experiment start time was when all robot finshed charging and started request task (Figure 5.9), and the experiment

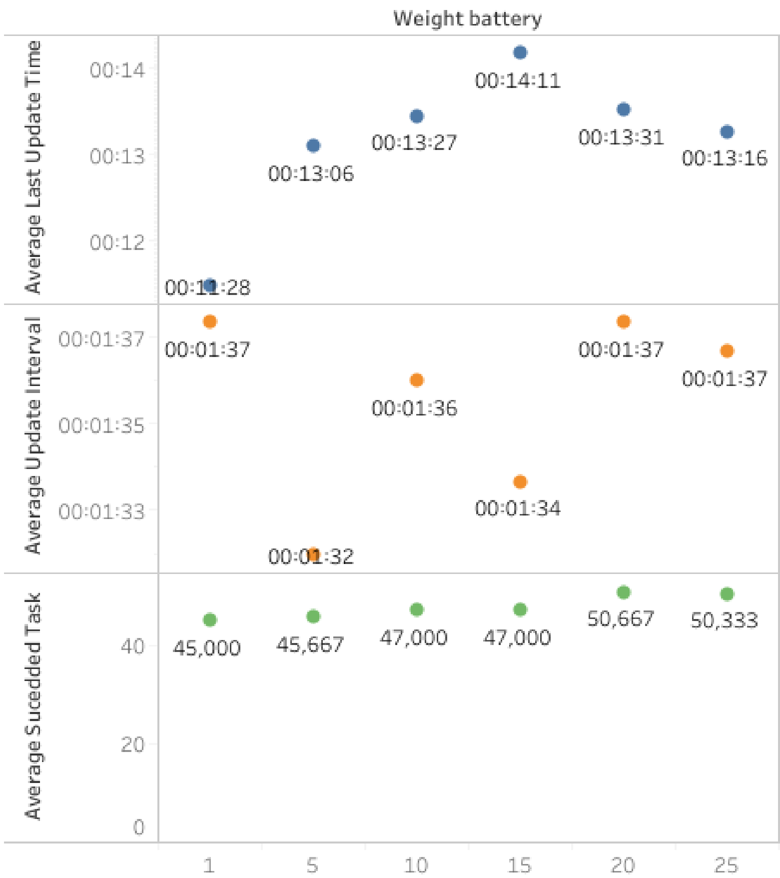


Figure 5.7: Experiment: Change W_{battery} under condition $W_{\text{update}} = 0.1$ and $W_{\text{possibility}} = 0$

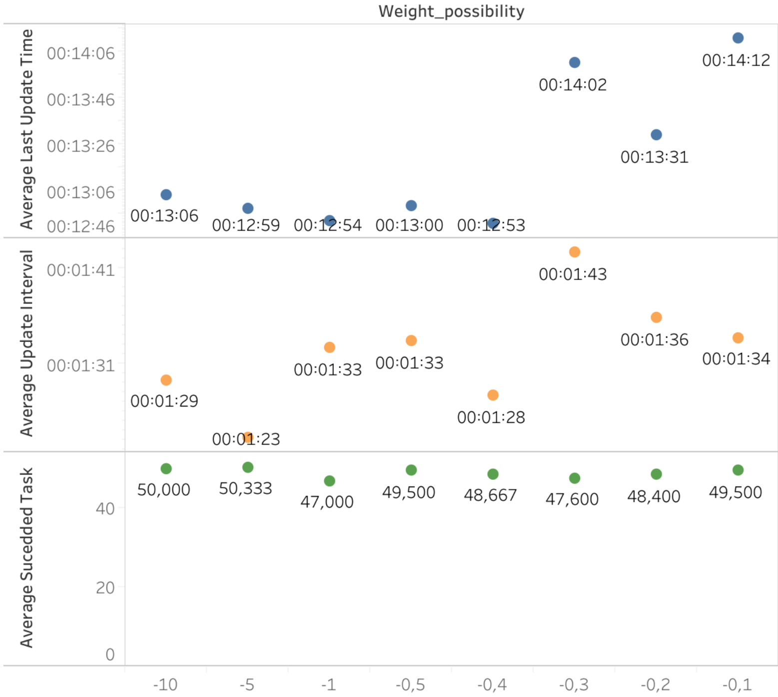


Figure 5.8: Experiment: Change $W_{\text{possibility}}$ under condition $W_{\text{battery}} = 15$ and $W_{\text{update}} = 0.1$

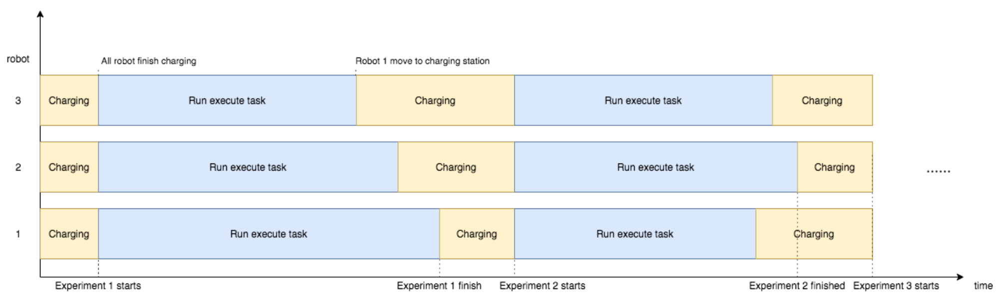


Figure 5.9: Execute Task Experiment Timeline

Task ID	Task Name	Start Time	Target ID	Robot ID	Priority	Task Status	Dependency	Finish Time	Description
1	Charging	NULL	18	1	5	Succedded	0	2020-06-02 11:03:12	Succeeded
2	Charging	NULL	19	2	5	Succedded	0	2020-06-02 11:03:12	Succeeded
3	Charging	NULL	20	3	5	Succedded	0	2020-06-02 11:03:11	Succeeded
4	ExecuteTask	2020-06-02 11:04:17	21	3	2	Succedded	0	2020-06-02 11:05:16	Succeeded
5	ExecuteTask	2020-06-02 11:05:22	22	1	2	Succedded	0	2020-06-02 11:07:29	Succeeded
6	ExecuteTask	2020-06-02 11:06:27	23	2	2	Succedded	0	2020-06-02 11:08:06	Succeeded
7	ExecuteTask	2020-06-02 11:07:32	24	3	2	Succedded	0	2020-06-02 11:09:25	Succeeded
8	ExecuteTask	2020-06-02 11:08:37	25	1	2	Succedded	0	2020-06-02 11:10:46	Succeeded
9	ExecuteTask	2020-06-02 11:09:42	26	2	2	Succedded	0	2020-06-02 11:12:42	Succeeded
10	ExecuteTask	2020-06-02 11:10:47	27	3	2	Succedded	0	2020-06-02 11:12:56	Succeeded
11	ExecuteTask	2020-06-02 11:11:52	28	1	2	Succedded	0	2020-06-02 11:14:13	Succeeded
12	ExecuteTask	2020-06-02 11:12:57	29	2	2	Succedded	0	2020-06-02 11:14:20	Succeeded
13	ExecuteTask	2020-06-02 11:14:02	30	3	2	Succedded	0	2020-06-02 11:15:36	Succeeded
14	ExecuteTask	2020-06-02 11:15:07	21	1	2	Succedded	0	2020-06-02 11:18:07	Succeeded
15	ExecuteTask	2020-06-02 11:16:12	22	2	2	Succedded	0	2020-06-02 11:18:49	Succeeded
16	ExecuteTask	2020-06-02 11:17:17	23	3	2	Succedded	0	2020-06-02 11:18:58	Succeeded
17	ExecuteTask	2020-06-02 11:18:22	24	1	2	Succedded	0	2020-06-02 11:20:14	Succeeded
18	ExecuteTask	2020-06-02 11:19:27	25	2	2	Succedded	0	2020-06-02 11:21:36	Succeeded
19	Charging	NULL	18	1	5	Succedded	0	2020-06-02 11:23:08	Succeeded
20	Charging	NULL	19	2	5	Succedded	0	2020-06-02 11:23:08	Succeeded
21	Charging	NULL	20	3	5	Succedded	0	2020-06-02 11:23:08	Succeeded
22	ExecuteTask	2020-06-02 11:24:13	21	3	2	Succedded	0	2020-06-02 11:25:12	Succeeded
23	ExecuteTask	2020-06-02 11:25:18	22	2	2	Succedded	0	2020-06-02 11:26:54	Succeeded
24	ExecuteTask	2020-06-02 11:26:23	23	1	2	Succedded	0	2020-06-02 11:28:35	Succeeded
25	ExecuteTask	2020-06-02 11:27:28	24	3	2	Succedded	0	2020-06-02 11:29:23	Succeeded
26	ExecuteTask	2020-06-02 11:28:33	25	2	2	Succedded	0	2020-06-02 11:30:41	Succeeded
27	ExecuteTask	2020-06-02 11:29:38	26	1	2	Succedded	0	2020-06-02 11:32:37	Succeeded
28	ExecuteTask	2020-06-02 11:30:43	27	3	2	Succedded	0	2020-06-02 11:32:54	Succeeded
29	ExecuteTask	2020-06-02 11:31:48	28	2	2	Succedded	0	2020-06-02 11:34:09	Succeeded
30	ExecuteTask	2020-06-02 11:32:53	29	1	2	Succedded	0	2020-06-02 11:34:15	Succeeded
31	ExecuteTask	2020-06-02 11:33:58	30	3	2	Succedded	0	2020-06-02 11:35:32	Succeeded
32	ExecuteTask	2020-06-02 11:35:03	21	2	2	Succedded	0	2020-06-02 11:38:03	Succeeded
33	ExecuteTask	2020-06-02 11:36:08	22	1	2	Succedded	0	2020-06-02 11:38:45	Succeeded
34	ExecuteTask	2020-06-02 11:37:13	23	3	2	Succedded	0	2020-06-02 11:38:54	Succeeded
35	ExecuteTask	2020-06-02 11:38:18	24	2	2	Succedded	0	2020-06-02 11:40:11	Succeeded
36	ExecuteTask	2020-06-02 11:39:23	25	1	2	Succedded	0	2020-06-02 11:41:40	Succeeded
37	Charging	NULL	18	1	5	Succedded	0	2020-06-02 11:43:45	Succeeded
38	Charging	NULL	19	2	5	Succedded	0	2020-06-02 11:43:45	Succeeded
39	Charging	NULL	20	3	5	Succedded	0	2020-06-02 11:43:45	Succeeded

Table 5.1: Task Table

finish time was when the latest task is finished. The experiment duration was an important evaluation factor, which was the time difference between experiment start time and experiment finished time. There end state of "execute tasks" were evaluated. In an experiment result table, the "Succeeded" column counted the tasks ended with "Succeeded" state, the "Expired" column counted the tasks ended with "Succeeded" state, the "Failed" column counted the tasks ended with "Runing", "Error", "Canceled", "To rerun" states. Table 5.2 is an example of experiment result.

Experiment Result The experiment result (Table 5.2.) shows that in experiment 1, all 15 task were succesfully finsihed with minimalexperiment duraion. In experiment 2, 4, 5 mores than half of the task were expired. In experiment 3, all tasks were successfully finished but it took more time than experiment 1. The experiment 6-24 (Table 5.3 and Figure 5.3) evaluated each decision variable sepratelly.

Experiment Analysis. Comparing experiment 1-5, it was concluded that the cost function with multiple decision variables had better performance than a cost function with single decition variable. However, according to experiment 6-24, the "experiment duration" changed very little when only one desition variable changed others unchanged, therefore how each decision variable affect the task scheduling is still unknown.

Experiment	W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door open possibility}}$	W_{priority}	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
1	1.00	1.00	-1.00	-1.00	00:18:23	15	15	0	0
2	1.00	0.00	0.00	0.00	00:17:23	15	7	8	0
3	0.00	1.00	0.00	0.00	00:18:56	15	15	0	0
4	0.00	0.00	-1.00	0.00	00:18:41	15	5	10	0
5	0.00	0.00	0.00	-1.00	00:17:21	15	3	12	0

Table 5.2: Runing execute task with single decision variable and multiple decision variables

5.4.2 Experiment: Find the Best Weight Combinations

Experiment Introduction In addition to cost function, the task interval may affect experiment duration and experiment succeeded rate. Therefore, three set experiment are generated to find the best weight combinations.

Sampling design Table 5.1 shows "task table" in database. At this point, robots finished task 4-21 in experiment 1 and finished 21-39 in experiment 2. When simulation started, robots ran charging task (Task 1-3 in Table 5.1) and moved to corresponding charging station and started charging (Figure 4.8). For example, robot 1 charged at charging staion 1, robot 2 charged at charging staion 2, robot 3 charged at charging staion 3. When robot fully charged, the first experiment started (Figure 5.9), and 15

Experiment	W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door open possibility}}$	W_{priority}	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
6	20	1	-1	-1	00:09:26	15	10	5	0
7	40	1	-1	-1	00:09:48	15	10	5	0
8	60	1	-1	-1	00:09:50	15	10	5	0
9	80	1	-1	-1	00:09:47	15	10	5	0
10	100	1	-1	-1	00:09:06	15	10	5	0
11	1	20	-1	-1	00:09:48	15	10	5	0
12	1	40	-1	-1	00:09:38	15	9	5	0
13	1	60	-1	-1	00:10:09	15	10	5	0
14	1	80	-1	-1	00:09:46	15	10	5	0
15	1	100	-1	-1	00:10:01	15	10	5	0
16	1	1	-20	-1	00:09:48	15	10	5	0
17	1	1	-40	-1	00:09:50	15	10	5	0
18	1	1	-60	-1	00:10:24	15	10	5	0
19	1	1	-80	-1	00:09:48	15	10	5	0
20	1	1	-100	-1	00:09:35	15	10	5	0
21	1	1	-1	-20	00:10:22	15	10	5	0
22	1	1	-1	-40	00:10:02	15	10	5	0
23	1	1	-1	-60	00:09:48	15	10	5	0
24	1	1	-1	-80	00:09:51	15	10	5	0

Table 5.3: only one desition variable changed others unchanged

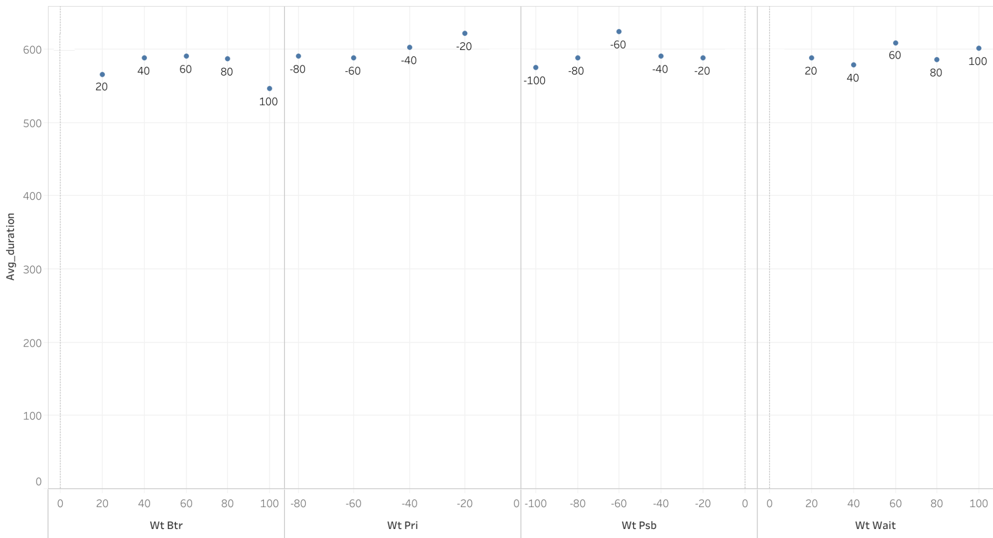


Figure 5.10: Only one desition variable changed others unchanged

"execute tasks" and 3 "charging tasks" were created (Task 4-21). Especially, in the same experiment, the task interal were the same. In Table 5.1 the task interval was 65 seconds. In different experiment, the corresponding task had the same goal positions (Task 4 and 21, Task 4 and 22). When all robot finished tasks, the experiment would be finished and robots charged at previous charging station(Task 19-21). These rules ensured that robot not only started at their initial positions but also processed same task set and not shuted down because of power exhaustion.

Analysis design In each experiemnt, 3 robots need to finish 15 "execute tasks" and 3 "charging tasks". The experiment start time was when all robot finshed charging and started request task (Figure 5.9), and the experiment finish time was when the latest task is finished. The experiment duration was an important evaluation factor, which was the time difference between experiment start time and experiment finished time. There end state of "execute tasks" were evaluated. In an experiment result table, the "Succeeded" column counted the tasks ended with "Succeeded" state, the "Expired" column counted the tasks ended with "Succeeded" state, the "Failed" column counted the tasks ended with "Runing", "Error", "Canceled", "To rerun" states. Table 5.2 is an example of experiment result.

Experiment Result The first set of experiment uses "execute tasks" with 30 seconds interval. Its best weight combinations is shown in (Table 5.4). The second set of experiments used "execute tasks" with 45 seconds interval. Its best weight combinations is shown in (Table 5.4). The third set of experiments used "execute tasks" with 65 seconds interval. Its best weight combinations is shown in (Table 5.6).

ToDo: Change experiment result table to graph and write more analysis according to graph

Analysis. It was concluded that the longer the task interval, the higher the task completion rate.

W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door open possibility}}$	W_{priority}	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
1	1	-1	-3	00:09:05	15	10	5	0
1	1	-30	-1	00:09:05	15	10	5	0
1	20	-20	-1	00:09:05	15	10	5	0
1	25	-1	-1	00:09:05	15	10	5	0
5	1	-5	-5	00:09:05	15	10	5	0

Table 5.4: Best weight combinations with task interval 30s

5.4.3 Experiment: Impact of the Number of Robots

Experiment Introduction In this set of experiments, how the number of robots affected task scheduling was evaluated. One of the best weight combination was selected: $W_{\text{battery}} = 10, W_{\text{waiting time}} = 1, W_{\text{possibility}} = -1, W_{\text{priority}} = -10.$

W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door open possibility}}$	W_{priority}	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
1	1	-1	-10	00:11:35	15	12	3	0
1	50	-50	-50	00:11:35	15	12	3	0
25	20	0	0	00:11:35	15	12	3	0
35	30	0	0	00:11:35	15	12	3	0
40	10	0	0	00:11:35	15	12	3	0
45	1	-1	-1	00:11:35	15	12	3	0
50	50	-50	-1	00:11:35	15	12	3	0

Table 5.5: Best weight combinations with task interval 45s

W_{battery}	$W_{\text{waiting time}}$	$W_{\text{door open possibility}}$	W_{priority}	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
10.00	1.00	-1.00	0.00	00:18:23	15	15	0	0
1.00	1.00	-1.00	0.00	00:18:23	15	15	0	0
1.00	1.00	-1.00	-1.00	00:18:23	15	15	0	0
1.00	5.00	-1.00	-1.00	00:18:23	15	15	0	0
1.00	5.00	-1.00	-5.00	00:18:23	15	15	0	0
1.00	10.00	-10.00	-1.00	00:18:23	15	15	0	0
1.00	5.00	-5.00	-5.00	00:18:23	15	15	0	0
1.00	5.00	-10.00	-5.00	00:18:23	15	15	0	0
1.00	5.00	-1.00	-1.00	00:18:23	15	15	0	0
1.00	20.00	-1.00	-1.00	00:18:23	15	15	0	0
1.00	1.00	-1.00	-30.00	00:18:23	15	15	0	0
1.00	1.00	-1.00	-35.00	00:18:23	15	15	0	0
10.00	1.00	-1.00	-5.00	00:18:23	15	15	0	0
10.00	10.00	-1.00	-1.00	00:18:23	15	15	0	0
10.00	1.00	-1.00	-1.00	00:18:23	15	15	0	0
10.00	1.00	-10.00	-1.00	00:18:23	15	15	0	0
10.00	5.00	-5.00	-5.00	00:18:23	15	15	0	0
10.00	1.00	-10.00	-5.00	00:18:23	15	15	0	0
10.00	10.00	-10.00	-10.00	00:18:23	15	15	0	0
10.00	1.00	-10.00	-1.00	00:18:23	15	15	0	0
25.00	1.00	-1.00	-25.00	00:18:23	15	15	0	0
30.00	30.00	-1.00	-1.00	00:18:23	15	15	0	0
45.00	1.00	-1.00	-1.00	00:18:23	15	15	0	0
45.00	1.00	-45.00	-1.00	00:18:23	15	15	0	0
5.00	10.00	-10.00	-1.00	00:18:23	15	15	0	0
5.00	5.00	-1.00	-1.00	00:18:23	15	15	0	0
5.00	10.00	-5.00	-1.00	00:18:23	15	15	0	0
5.00	5.00	-10.00	-1.00	00:18:23	15	15	0	0
5.00	10.00	-10.00	-1.00	00:18:23	15	15	0	0
5.00	5.00	-1.00	-5.00	00:18:23	15	15	0	0
5.00	10.00	-5.00	-10.00	00:18:23	15	15	0	0

Table 5.6: Best weight combinations with task interval 65s

Sampling design When simulation started, robots ran charging task and moved to corresponding charging station and started charging (Figure 4.8). For example, robot 1 charged at charging staion 1, robot 2 charged at charging staion 2, robot 3 charged at charging staion 3. When robot fully charged, the first experiment started (Figure 5.9), and 15 “execute tasks” and 3 “charging tasks” were created (Task 4-21). Especially, in the same experiment, the task interal were the same. In Table 4.7 the task interval was 65 seconds. In different experiment, the corresponding task had the same goal positions (Task 4 and 21, Task 4 and 22). When all robot finished tasks, the experiment would be finished and robots charged at previous charging station(Task 19-21). These rules ensured that robot not only started at their initial positions but also processed same task set and not shuted down because of power exhaustion.

Analysis design As is discussed in last paragraph, in each experiemnts, robots need to finish 15 “execute tasks” and 3 “charging tasks”. The experiment start time was when all robot finshed charging and started request task (Figure 5.9), and the experiment finish time was when the latest task is finished. The experiment duration was an important evaluation factor, which was the time difference between experiment start time and experiment finished time. There end state of “execute tasks” were evaluated. In an experiment result table, the “Succeeded” column counted the tasks ended with “Succeeded” state, the “Expired” column counted the tasks ended with “Succeeded” state, the “Failed” column counted the tasks ended with “Runing”, “Error”, “Canceled”, “To rerun” states. Table 5.2 is an example of experiment result.

Result The experiment result (Table 5.7) showed that when one robot processed tasks, only 7 tasks were finished successfully; when two robots processed the same task set, 11 tasks were finished succesfully. Compared with experiment 1, experiment 2 tooks 102 seconds more but 2 more tasks were completed; when three robots processed the same task set, all 15 tasks were finished succesfully. Compared with experiment 2, the experiment duration is slightly increased but 4 more tasks were finished succesfully.

Analysis It was concluded that as the number of robots increased, the task completion rate increased. However, the relationship between the number of robots and the speed of completing the task set cannot be obtained.

Experiment	Robot number	Experiment Duration	Total Task	Succedded Task	Expired Task	Failed Task
1	1	00:16:56	15	7	8	0
2	1	00:16:52	15	7	8	0
3	1	00:17:02	15	7	8	0
Average	1	00:16:56	15	7	8	0
4	2	00:19:06	15	11	4	0
5	2	00:18:24	15	11	4	0
6	2	00:18:24	15	11	4	0
Average	2	00:18:38	15	11	4	0
7	3	00:18:23	15	15	0	0
8	3	00:18:55	15	15	0	0
9	3	00:18:57	15	15	0	0
Average	3	00:18:45	15	15	0	0

Table 5.7: Result 4: Different number of robot processing the same task set

Chapter 6

Discussion

The meaning of this paragraph is to interpret the results of the performed work. It will always connect the introduction, the postulated hypothesis and the results of the thesis/bachelor/master.

It should answer the following questions:

- Could your results answer your initial questions?
- Did your results agree with your initial hypothesis?
- Did you close your problem, or there are still things to be solved? If yes, what will you do to solve them?

write about limitations on sensor simulator: sensor send a table of measuement results.

Chapter 7

Acknowledgements

(This part is optional, and it could be completely excluded by deleting
\include {content/chapters/chapter7}
from the Firstname_Lastname_Diplom_Master_arbeit.tex file)

This paragraph could mention people or institutions that supported you to some extent with your work or friends and relatives that supported you during your study period.

Bibliography

[ASE] *A* search algorithm.* https://en.wikipedia.org/wiki/A*_search_algorithm. accessed 2020-10-17.

[ASM15] Afanasyev, Ilya, Artur Sagitov and Evgeni Magid: *ROS-Based SLAM for a Gazebo-Simulated Mobile Robot in Image-Based 3D Model of Indoor Environment.* In Battiato, Sebastiano, Jacques Blanc-Talon, Giovanni Gallo, Wilfried Philips, Dan Popescu and Paul Scheunders (editors): *Advanced Concepts for Intelligent Vision Systems*, pages 273–283, Cham, 2015. Springer International Publishing.

[BMR+17] Booth, K. E. C., S. C. Mohamed, S. Rajaratnam, G. Nejat and J. C. Beck: *Robots in Retirement Homes: Person Search and Task Planning for a Group of Residents by a Team of Assistive Robots.* IEEE Intelligent Systems, 32(6):14–21, 2017.

[CSMV09] Christodoulopoulos, K., V. Sourlas, I. Mpakolas and E. Varvarigos: *A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in Grid networks.* Computer Communications, 32(7):1172 – 1184, 2009.

[DEA] *Dead reckoning.* https://en.wikipedia.org/wiki/Dead_reckoning. accessed 2020-10-14.

[GMS17] Gavran, Ivan, Rupak Majumdar and Indranil Saha: *Antlab: A Multi-Robot Task Server.* ACM Trans. Embed. Comput. Syst., 16(5s), September 2017.

[GZ] *Gazebo Main Page.* http://gazebosim.org/. accessed 2020-10-14.

[HC17] Hancheol Cho, Leon Jung, Darby Lim: *ROS Robot Programming (English).* ROBOTIS, 12 2017.

[IGM97] Ikeda, Mitsuru, Shogo Go and Riichiro Mizoguchi: *Opportunistic Group Formation.* In *Proceedings of the Conference on Artificial Intelligence in Education (AI-ED)*, pages 167–174, Amsterdam, 1997. IOS Press.

[JM13] Jia, Xiao and M. Meng: *A survey and analysis of task allocation algorithms in multi-robot systems.* 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 2280–2285, 2013.

[KM16] Kim, Yongho and Eric T. Matson: *A Realistic Decision Making for Task Allocation in Heterogeneous Multi-agent Systems.* Procedia Computer Science, 94:386 – 391, 2016. The 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops.

[KSD13] Korsah, G. Ayorkor, Anthony Stentz and M. Bernardine Dias: *A comprehensive taxonomy for multi-robot task allocation.* The International Journal of Robotics Research, 32(12):1495–1512, 2013.

[LZK15] Lee, D., S. A. Zaheer and J. Kim: *A Resource-Oriented, Decentralized Auction Algorithm for Multirobot Task Allocation.* IEEE Transactions on Automation Science and Engineering, 12(4):1469–1481, 2015.

[MAdR02] Middleton, S. E., H. Alani and D. C. de Roure: *Exploiting Synergy between Ontologies and Recommender Systems.* In *Proceedings of the Semantic Web Workshop at the WWW Conference*, Honolulu, HI, 2002.

[MOV] *Move Base Node.* `http://wiki.ros.org/move_base`. accessed 2020-09-25.

[NMMG17] Nunes, Ernesto, Marie Manner, Hakim Mitiche and Maria Gini: *A taxonomy for task allocation problems with temporal and ordering constraints.* Robotics and Autonomous Systems, 90:55 – 70, 2017. Special Issue on New Research Frontiers for Intelligent Autonomous Systems.

[PAR] *Potential field.* `https://en.wikipedia.org/wiki/Particle_filter`. accessed 2020-10-17.

[PNK+15] Pyo, Yoonseok, Kouhei Nakashima, Shunya Kuwahata, Ryo Kurazume, Tokuo Tsuji, Kenâ€™ichi Morooka and Tsutomu Hasegawa: *Service robot system with an informationally structured environment.* Robotics and Autonomous Systems, 74:148 – 165, 2015.

[POT] *potential field.* `http://www.cs.cmu.edu/~./motionplanning/lecture/Chap4-Potential-Field_howie.pdf`. accessed 2020-10-17.

[ros] *ROS.* `https://www.ros.org`. accessed 2020-10-14.

[RRT] *RRT (Rapidly-exploring Random Tree).* `https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree`. accessed 2020-10-17.

[RVI] *Rviz.* `http://wiki.ros.org/rviz`. accessed 2020-10-17.

[Sal89] Salton, Gerard: *Automatic text processing — the transformation, analysis, and retrieval of information by computer.* Addison-Wesley series in computer science. Addison-Wesley, 1989.

[Seo17] Seo, Woojin: *Indoor Dead Reckoning Localization Using Ultrasonic Anemometer with IMU.* Journal of Sensors, 2017:12, 2017.

[SM07] Shah, K. and Y. Meng: *Communication-Efficient Dynamic Task Scheduling for Heterogeneous Multi-Robot Systems.* In *2007 International Symposium on Computational Intelligence in Robotics and Automation*, pages 230–235, 2007.

[T3S] *turtlebot3 SLAM.* `https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/`. accessed 2020-10-17.

Versicherung an Eides Statt

German

Hiermit versichere ich, dass ich die vorliegende Bachelor(Master)arbeit selbständig - mit Ausnahme der Anleitung durch die Betreuer - verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Ich versichere dass ich alle entsprechenden Angaben nach bestem Wissen und Gewissen vorgenommen habe, dass sie der Wahrheit entspechen und dass ich nichts verschwiegen habe. Mir ist bekannt, dass eine falsche Versicherung an Eides Statt nach §156 und nach §163 Abs. 1 des Strafgesetzbuches mit Freiheitsstrafe oder Geldstrafe bestraft wird.

English

I hereby declare that I have written this Bachelor (Master) thesis - except for the guidance of the supervisor - independently, using no other than the specified sources and resources, and that all quotations have been indicated. I declare that I have reported to the best of my knowledge all the relevant information, that it is true and that I concealed nothing. I am aware that a false declaration will be punished according to §156 and §163 par. 1 of the criminal code with a prison sentence or a monetary penalty.

Essen, October 18, 2020
(Ort, Datum)

Xuanjiao Zhu