The Development of Fleet Management System for Mobile Robots Delivering Medicine in Healthcare Environments

Thi Thoa Mac, Anh Quan Pham, Xuan-Thuan Nguyen*

School of Mechanical Engineering, Hanoi University of Science and Technology,

No. 1 Dai Co Viet, 100000 Vietnam; thoa.macthi@hust.edu.vn,
quan.pa212654m@sis.hust.edu.vn, thuan.nguyenxuan@hust.edu.vn
*Corresponding author

Abstract: This paper focuses on implementing a Fleet Management System (FMS) in hospitals, using autonomous mobile delivery robots. The primary aim is to enhance efficiency and alleviate the workload of healthcare staff. The FMS plays a crucial role in planning and controlling drug delivery tasks performed by a fleet of mobile robots, which retrieve medications from storage. Operating on a centralized control architecture, the FMS is a central hub for receiving, processing, and distributing user tasks to the multi-robot system. It continuously monitors the status of robot systems and stores real-time data. The FMS comprises a backend Application Programming Interface (API), a task scheduler with algorithms to solve task allocation problems, and a robust database management system.

Keywords: Mobile robots; path planning; Robot Operating System; vehicle routing problem; web server

1 Introduction

In recent years, healthcare centres have focused on improving patient and staff service quality. One key aspect is enhancing the mobility of medical personnel within the hospital to increase productivity. Tasks that involve moving medicines, equipment, samples, bedding, pharmaceuticals, packages, and medical waste can be time-consuming and non-value-added. Bardram and Bossen (2005) [1] have shown that medical staff can walk an average of 6.1 km during a 7.9-hour shift, while on-call nurses and doctors walk an average of 6.1 km for 18.9 hours. Deploying mobile robots in hospital environments can significantly improve indoor mobility activities, as highlighted by Huang, Cao, and Zhu (2019) [2]. This would bring numerous advantages to healthcare facilities. In the studies [3] and

[4], the authors presented a web-based software tool that enables the visualization of robot movements on a map and allows interaction with the robots through a user interface. In the research described in [4], the authors also developed a NodeJS server and a MySQL database to manage user and robot data. However, both studies directly connect the graphical user interface (GUI) to the ROS operating system on the robot. This approach is only suitable when all robots share a common ROS Master (a ROS server responsible for managing and operating the robots), which poses challenges in scalability and system management when multiple robots need to collaborate and integrate with the hospital's information system. Therefore, there is a need for a fleet management system for robots that can effectively manage the robot system while seamlessly exchanging information with the healthcare environment's infrastructure. This fleet management system would provide centralized control and coordination of multiple robots, allowing for efficient task allocation, resource optimization, and real-time communication with the hospital's information system.

The main contribution of our work is to develop the implementation of a Fleet Management System (FMS) for planning and controlling transportation tasks using a fleet of mobile robots in a hospital environment. We also investigate the design and implementation of a routing system, task scheduler, controller, and backend application programming interface (API) at the core of the FMS. The Rosbridge protocol for communication between the server and robot system is developed in this paper. The FMS is implemented and evaluated using a virtual hospital floor model to validate the proposed algorithm.

The paper structure on multi-task allocation focusing on Fleet Management Systems (FMS) is as follows: Section 2 delves into the theoretical background necessary to address the challenges and complexities associated with multi-task allocation. Section 3 describes the critical components of the FMS architecture and emphasizes how the described architecture contributes to the efficient management of fleet tasks. Section 4 depicts critical findings from the computational experiment and the implications of the results for the Fleet Management System and multi-task allocation in general. Finally, conclusions and future research lines are outlined in the last Section.

2 Task Scheduler

2.1 Problem Description

In the given problem, the robot team can perform many logistics tasks related to transporting goods such as drugs, medical equipment, biological samples, bedding, pharmaceuticals, parcels, or medical waste within hospital premises.

From an operational point of view, the Fleet Management System (FMS)'s operational requirements involve determining each trolley's load, optimizing routes between distinct locations, and allocating tasks to robots based on specific criteria [5]. The load characteristics can also vary depending on the task, encompassing properties such as size, weight, and type (e.g., food, sanitary equipment, medicines).

The approach provides a systematic and efficient way to address the Fleet Management System (FMS) operational challenges involving trolley loading, robot routing, and task allocation. First, optimal paths between locations are calculated, then tasks are allocated to robots based on these paths, and finally, the load of each trolley is determined. Calculating the shortest path is computationally efficient and can be done cheaply for all location pairs. Once the distances are known, the task allocation problem becomes a Capacitated Vehicle Routing Problem (CVRP), where the objective is to minimize the total distance while considering constraints such as payload and battery. The optimal load of trolleys can then be easily determined based on the solution to the CVRP problem.

2.2 Theoretical Background

In the FMS task scheduler, we rely on the Capacitated Vehicle Routing Problem (CVRP) to address the robot path planning problem by picking up items from a warehouse and transporting them to specified locations.

The classical CVRP has defined on an undirected graph a graph $G = (V, H, c)$ where $V = \{0,1,2,...,n\}$ as the set of nodes (rooms), H is set of arcs, and $C = (c_{ij})$ is the distance matrix that associates each arc (i, j) belonging to *H*. Node 0 represents a depot at which are located at most *p* identical robots of capacity *Q*. The problem consists of determining routes for p robots (i) starting and ending at the depot, and such that (ii) each node is visited by exactly one vehicle, (iii) the total demand of any route does not exceed *Q*, and (iv) the total routing cost is minimized [6].

The minimum number of robots needed to serve all customers is $p_{\min} = \left\lceil \frac{\sum_{i=1}^{n} d_i}{Q} \right\rceil$.

The integer linear programming formulation (ILP) model is described next, considering the nomenclature (Table 1) and the mathematical notation (Janacek et.al. [7]).

Table 1

ILP for CVRP Multi Robot

Sets and Index	
V	set of nodes (vertices)
H	set of arcs (i,j)
i	origin node
j	destination node
Parameters	
Q	Maximum weight carrying limitation of each robot
c_{ij}	cost (distance) matrix is associated with each arc $(i,j) \in H$
p	robot located at the depot 0
d_i	The demand at the node $i \in V \setminus \{0\}$ and $d_i \le Q$
Decision Variables	
x_{ij}^r	1 if the robot r traverses an arc (i,j) in an optimal solution. 0 otherwise

Minimize

$$\min \sum_{r=1}^{p} \sum_{i=0}^{n} \sum_{j=0, i \ne j}^{n} c_{ij} x_{ij}^r \tag{1}$$

Subject to

$$\sum_{r=1}^{p} \sum_{j=0, i \ne j}^{n} x_{ij}^r = 1, \forall j \in \{1, \ldots, n\} \tag{2}$$

$$\sum_{j=1}^{n} x_{0j}^r = 1, \forall r \in \{1, \ldots, p\} \tag{3}$$

$$\sum_{i=0, i \ne j}^{n} x_{ij}^r = \sum_{i=0}^{n} x_{ji}^r, \forall j \in \{0, \ldots, n\}, \forall r \in \{1, \ldots, p\} \tag{4}$$

$$\sum_{i=0}^{n} \sum_{j=1, i \ne j}^{n} d_j x_{ij}^r \le Q, \forall r \in \{1, \ldots, p\} \tag{5}$$

$$\sum_{r=1}^{p} \sum_{i \in S} \sum_{j \in S, i \ne j} x_{ij}^r \le |S| - 1, \forall S \subseteq \{1, 2, \ldots, n\}, |S| \ge 2 \tag{6}$$

$$x_{ij}^r \in \{0,1\}, \forall r \in \{1, \ldots, p\}, \forall i, j \in V, i \ne j \tag{7}$$

The function (1) defines the objective function (Objective Function (1)) that aims to minimize the total travel cost incurred by the robots and (2) the degree constraints ensuring that each node is visited by exactly one robot. The flow constraints (3) and (4) ensure that each robot departs from the depot only once, and these constraints maintain balance in the number of robots arriving at and leaving each customer and the depot. Constraints (5) articulate the capacity constraints to ensure that the sum of demands from nodes visited in a route is at most the capacity of the robot providing the service, managing the load, and ensuring the feasibility of the assigned tasks. Sub-tour elimination constraints (6) guarantee that the solution contains no cycles disconnected from the depot and prevents inefficient or disconnected routes. The constraints (7) specify the definition domains of the variables. This model is recognized as a three-index vehicle flow formulation [7].

The VRP is NP-hard because it includes the Traveling Salesman Problem (TSP) as a particular case when and [6]. Most exact approaches are often developed from precise algorithms for the TSP, including Naddef and Rinaldi [8], Baldacci et al. [9]. Additionally, some heuristic algorithms to solve CVRP include Clarke and Wright [10], Fisher and Jaikumar [11]. In this study, Google OR-tools is employed for solving the CVRP, wherein the Euclidean distance determines the weights of edges between adjacent nodes. Figure 1 illustrates the solution to the CVRP with 16 delivery points and four vehicles with a maximum capacity based on the dataset of Google Or-tools [12].

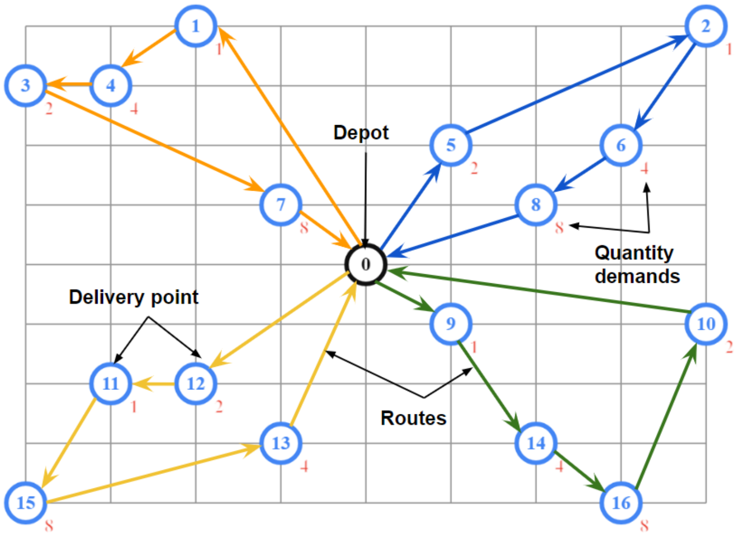


Figure 1

Google Or-tools solution for CVRP with $n = 16, p = 4$ *and* $Q = 15$

2.3 Task Allocation

To address a collection of pre-arranged or immediately triggered transportation tasks, this section introduces a task allocation strategy using a queuing mechanism, encompassing two categories: task queue and backlog queue. Let's assume there are p identical robots in the warehouse, each with a maximum capacity of Q. Each transportation task consists of a set of n destinations that the robot needs to reach. Each destination requires transporting d_i items from the warehouse.

Let $p' \leq p$ be the number of remaining robots that have not yet accepted the task. A new task will be pushed into the main queue. When it's time to execute the task, the server will dequeue tasks one by one for processing. If the total transportation demand of task k is less than the maximum capacity of a robot, the server will use the solution to the TSP problem to find the optimal route for a robot. If the remaining number p' of robots is insufficient to execute the task k, this task will be placed in the backlog queue (Figure 2).

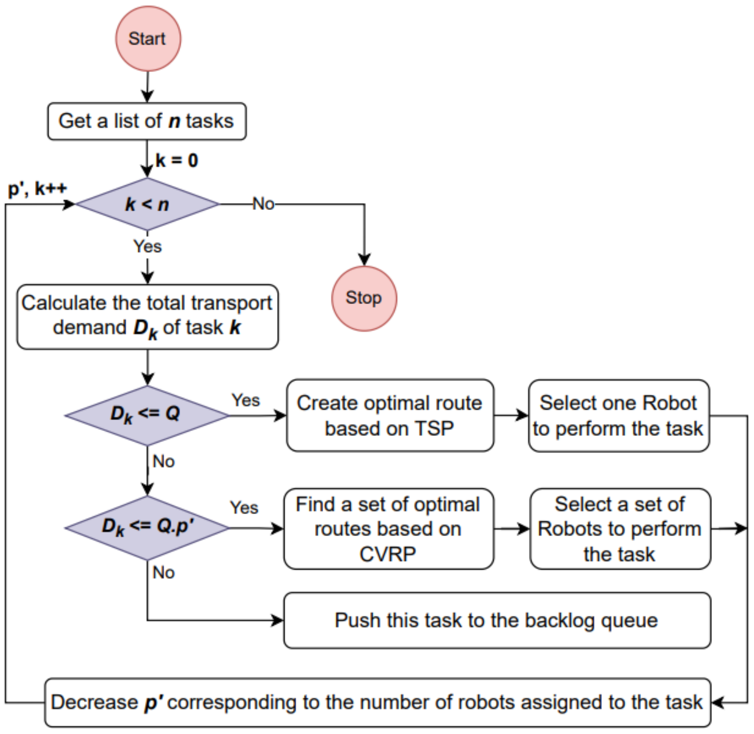

Start
Get a list of **n** tasks
k = 0
p', k++
k < n
No
Yes
Stop
Calculate the total transport demand D_k of task **k**
D_k <= Q
Yes
Create optimal route based on TSP
Select one Robot to perform the task
No
D_k <= Q.p'
Yes
Find a set of optimal routes based on CVRP
Select a set of Robots to perform the task
No
Push this task to the backlog queue
Decrease **p'** corresponding to the number of robots assigned to the task

Figure 2
Algorithm diagram of task queue processing

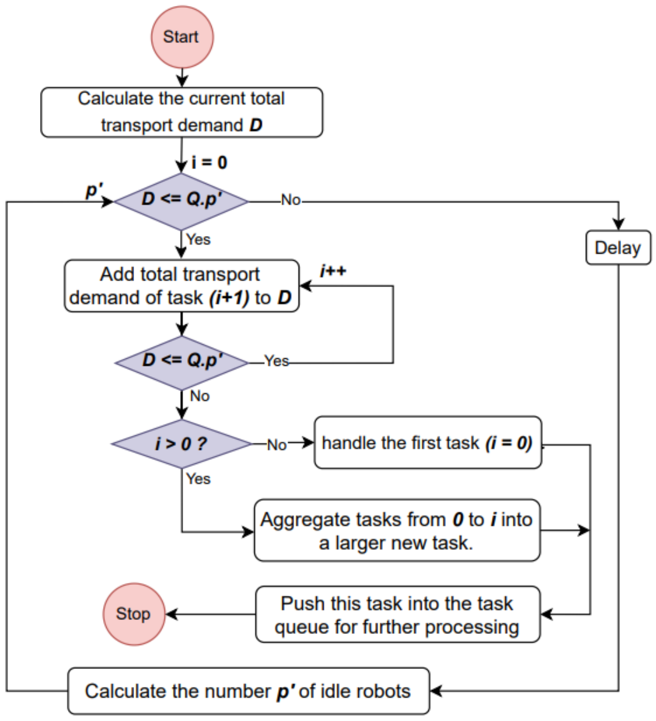

Start
Calculate the current total transport demand D
i = 0
p'
D <= Q.p'
No
Yes
Delay
Add total transport demand of task (i+1) to D
i++
D <= Q.p'
Yes
No
i > 0 ?
No
handle the first task (i = 0)
Yes
Aggregate tasks from 0 to i into a larger new task.
Push this task into the task queue for further processing
Stop
Calculate the number p' of idle robots

Figure 3
Algorithm processes and merges backlog tasks

The FMS system will monitor the backlog queue on a regular cycle. When a sufficient number of robots is available, the server will consolidate a batch of stalled tasks by aggregating tasks with the exact delivery location into a more significant task suitable for execution by a group of robots. After that, it will push this consolidated task back into the task queue (Figure 3).

3 Fleet Management System

3.1 Overview Fleet Management System

In this study, the Fleet Management System (FMS) comprises a Nodejs server communicating with the database, implementing authentication mechanisms, and algorithms for task allocation and optimal route computation for the transportation robot system (Figure 4). A web interface allows end-users to interact directly with the server and command the robots. FMS uses both MySQL and Redis databases, where MySQL is employed to store user information to serve the authorization authentication mechanism. Simultaneously, it stores and manages data, configurations of Robots, map information, and details of each transport order. These data can then be analyzed and used for statistical purposes. Concurrently, Redis is used to store and create a cache memory to support the Robot task allocation mechanism, processing queued tasks.

In this study, the Fleet Management System (FMS) comprises a Nodejs server communicating with the database, implementing authentication mechanisms and algorithms for task allocation and optimal route computation for the transportation robot system (Figure 4). A web interface allows end-users to interact directly with the server and command the robots. FMS uses MySQL and Redis databases, where MySQL stores user information to serve as the authorization authentication mechanism. Simultaneously, it stores and manages data, configurations of Robots, map information, and details of each transport order. These data can then be analyzed and used for statistical purposes. Concurrently, Redis stores and creates a cache memory to support the Robot task allocation mechanism, processing queued tasks.

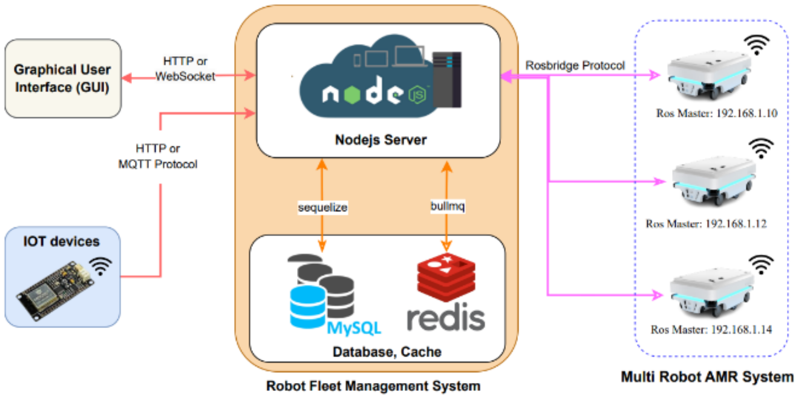


Figure 4
Communication Diagram in FMS System

Protocols used in the system: HTTP protocol is a hypertext transfer protocol that operates on a request-response mechanism, facilitating communication between the server and clients. WebSocket protocol: a full-duplex communication protocol that enables real-time data transmission and interactive connections between the server and clients over the internet. MQTT protocol: a lightweight, reliable messaging protocol based on the publish/subscribe mechanism, used to connect remote devices in bandwidth-constrained environments. This protocol is used for the Nodejs server to communicate with Internet of Things (IoT) devices, such as robot charging stations, robot call stations, or goods collection stations.

Additionally, the FMS is developed on top of the ROS platforms, and model Mir100 is used to validate the FMS in real-world scenarios. ROS is an open-source meta-operating system designed to enable scheduling and planning activities in industrial environments [13]. From the side of the ROS, the rosbridge server [14] was used, a middleware abstraction layer that provides the communication protocol used by the ROSbridge server to enable seamless interaction between ROS and the Node.js server. The ROS Master will control and operate an AMR unit (Mir100). The NodeJS server distinguishes between AMR units based on the static IP address of the robot within the network. Each robot can listen and receive assignments from the server via Rosbridge protocols, and then they will automatically move to the specified destination locations to perform the required task.

3.2 Software Architecture

Initially, the Robot is able to scan and build a map describing the boundaries and fixed objects in the environment (Figure 5) based on the SLAM algorithm. Once completed, this map will be stored in a pair of files: PNG and YAML file corresponding to the map_server package in ROS, where the YAML file describes the map meta-data and names the image file, and the image file encodes the occupancy data. It is then shared and synchronized with AMR Robots via FMS. Based on this map, engineers determine the coordinates of the destination points (locations of rooms, pharmacies, elevators, warehouses) that the Robot needs to go to and store them in the MySQL database. The FMS server will receive transportation tasks from users, perform task allocation mechanisms for the Robot system, and send corresponding destination coordinates to the navigation system of each Robot.

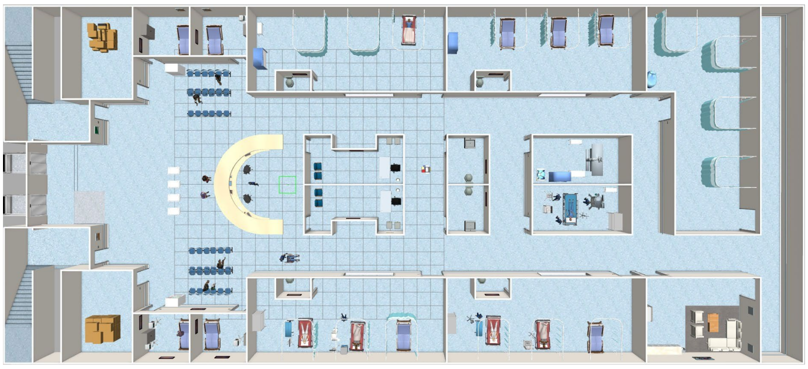


Figure 5

Simulating the hospital environment based on the AWS Robot Maker package

In Figure 6 the system architecture of the proposed FMS is presented. ReactJS Client (1) implements the interface for managing transportation tasks and tracking the location of each AMR Robot on the map. ReactJS, an open-source JavaScript library, is used to construct a user interface based on individual UI components. The functionality of the Node.js Server is to add clarity to the role it plays in managing the master data of the Fleet Management System (FMS) and facilitating various tasks.

NodeJS Server (2) provides a REST interface to manage the master data of the FMS, calculate an optimal assignment of a set of tasks to a group of robots to be executed on a specific date and time, and add a new task to be executed in the current shift.

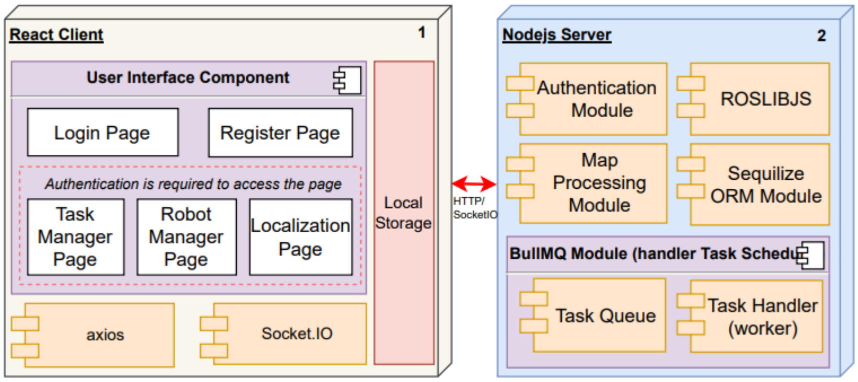
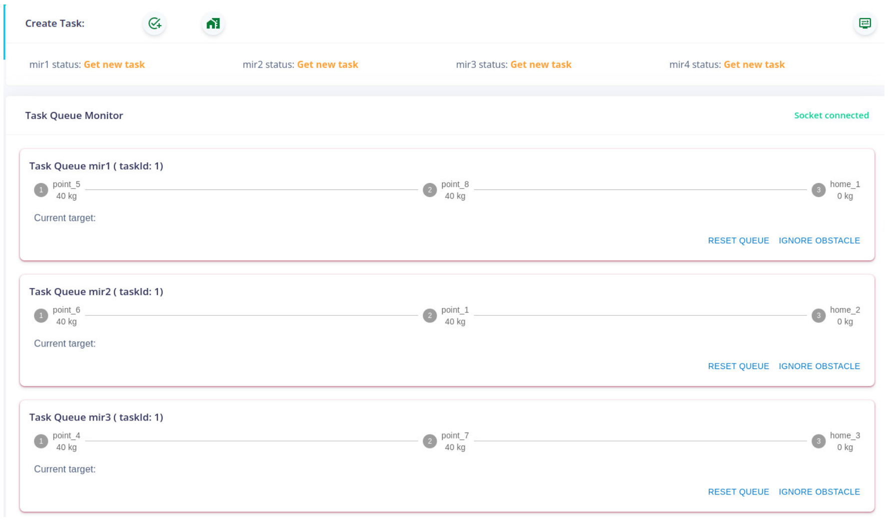


Figure 6
System Architecture

These APIs enable end-users to register tasks, schedule deliveries, and monitor the progress of Robot tasks through the user interface. The NodeJS server use the BullMq library, which implements a fast and robust queue system built on top of Redis. Task Handler (worker) listens to the task queue, and when a new task is registered, they dequeue it and execute the algorithm to allocate tasks and optimize routes for the robot fleet, as described in section 2.3 (Task Handler). Subsequently, they accurately send the coordinates of the destination points for each robot using the ROSLIBJS library (a library that transforms the ROS messages into a JSON format and transmits them to the ROS environment through the rosbridge module [4]). Figure 7 illustrates the web interface for managing task queues and robot locations.


Create Task:

mir1 status: Get new task mir2 status: Get new task mir3 status: Get new task mir4 status: Get new task

Task Queue Monitor Socket connected

Task Queue mir1 (taskId: 1)
1 point_5 40 kg — 2 point_8 40 kg — 3 home_1 0 kg
Current target:
RESET QUEUE IGNORE OBSTACLE

Task Queue mir2 (taskId: 1)
1 point_6 40 kg — 2 point_1 40 kg — 3 home_2 0 kg
Current target:
RESET QUEUE IGNORE OBSTACLE

Task Queue mir3 (taskId: 1)
1 point_4 40 kg — 2 point_7 40 kg — 3 home_3 0 kg
Current target:
RESET QUEUE IGNORE OBSTACLE

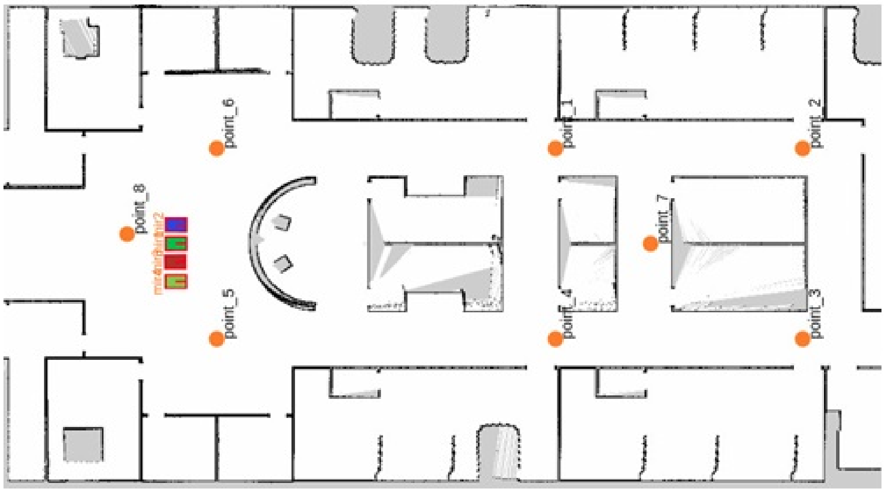


Figure 7
Monitors the robot's task queue and localization via the frontend interface

Simultaneously, the NodeJS server provides APIs that can apply CRUD (Create, Read, Update, or Delete) operations to manage MySQL data, including user information (user roles), core data of the robot fleet (robot types, robot groups, and robot members), navigation information (navigation maps, location marker, etc.), and task information (types, task execution plans, etc.) [5]. The MySQL database design for the Fleet Management Robot system is divided into three main groups of tables:

Group 1 is designed for authentication and authorization purposes, including the user's table, which contains columns for storing user information and permissions. The roles table is used to store the roles available in the system.

Group 2 consists of tables that store data related to maps and locations: the maps table holds information about maps that have been scanned using the SLAM algorithm. The position_goals table is used to store information and coordinates of destination points.

Group 3 stores data related to tasks and robots: the tasks table contains information about tasks users request. The robots table is used to store information and configurations of AMR (Autonomous Mobile Robot) models. The subTasks table stores information about deliveries at corresponding destination points that a robot needs to perform during a mission.

The structure and relationships between the desired locations are illustrated in Figure 8.

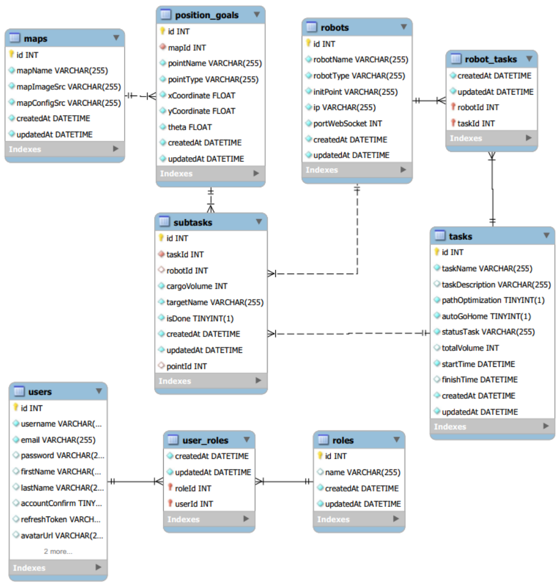


Figure 8
Database design on MySQL

4 Computational Experiments

The simulations were carried out on an HP zBook G3 laptop equipped with an Intel® Core™ i7-6820HQ CPU 2.7GHz (8 CPUs). Using Gazebo software and the Mir100 robot model developed by DFK in the ROS Noetic environment on

Ubuntu 20.04, we simulated the collaborative work of four Mir100 Robots with a maximum load capacity of 100 kg in a hospital environment created by AWS Robot Maker (5). The coordinates of the depot and delivery points are depicted in Figure 9. The closed-loop coloured paths with arrows represent the routes of each robot. Below each destination node is the cargo the robot needs to transport to that node. Their task is to transport medical supplies from the warehouse to the registered locations and then return to the warehouse.

In Table 2, four Mir100 Robots are assigned identifiers (robotId) from 1 to 4, and transportation tasks are numbered as taskId. Each task represents the total transportation demand, initialization time, and allocation time for the AMR Robot system. Here, tasks 1 and 2 are allocated immediately after initialization since they both have transportation demands of 200 kg and 120 kg, respectively, requiring two Mir100 Robots each. Transportation tasks initiated later must be put into the backlog queue and postponed until the list of available Robots is sufficient to perform the next task. This is why their allocation time is greater than the initialization time.

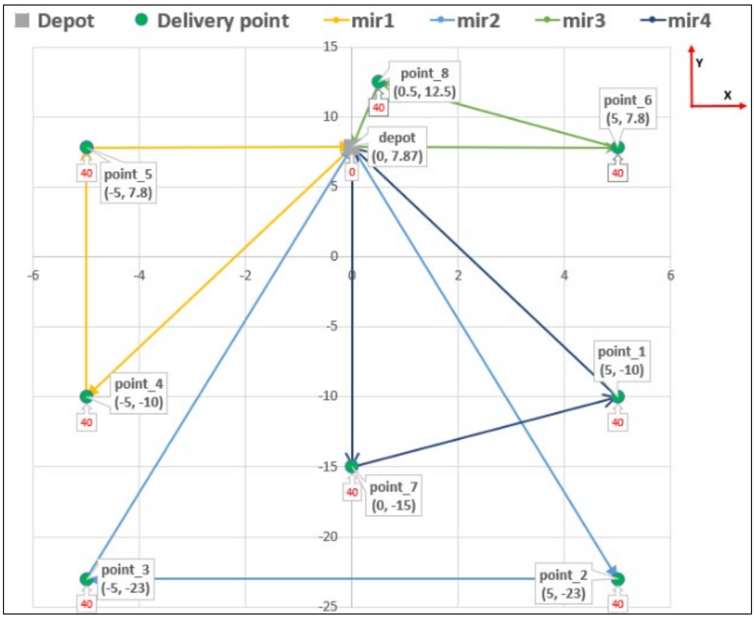


Figure 9

Result of CVRP based on Google Or-tools in the simulation environment

The data indicates that when Robot 1 completes its transportation route, it will be allocated the next task at the front of the queue (Task 3 with a total transportation demand of 100 kg). Similarly, Task 4, requiring two Robots for execution, will be allocated when there are two available Robots (Robots 3 and 4). The remaining tasks are also allocated according to the strategy outlined in Section 2.3.

Table 2
Task allocation results from the MySQL database

taskId	Total transport weight (kg)	Created At (hh:mm:ss)	Executed At (hh:mm:ss)	robotId
1	200	22:49:46	22:49:53	3, 4
2	120	22:50:35	22:50:41	1, 2
3	100	22:50:55	22:51:27	1
4	200	22:53:26	22:53:33	3, 4
5	100	22:53:46	22:53:52	1
6	50	22:54:15	22:54:21	2
7	130	22:55:25	22:58:18	2
8	90	22:55:33	22:58:37	2
9	40	22:55:58	22:59:17	4
10	60	22:56:15	23:02:13	1
11	50	23:01:57	23:02:59	4
12	60	23:03:16	23:03:57	2
13	100	23:03:34	23:05:33	4

Conclusions and Future Work

The FMS (Fleet Management System) integrates a task allocation strategy based on a queue processing mechanism using solutions from two problems, TSP (Traveling Salesman Problem) and CVRP (Capacitated Vehicle Routing Problem). This integration holds significant potential for optimizing the distribution of medical supplies in healthcare environments, leading to increased efficiency and cost savings. Based on these results, the research demonstrates practical applications for optimizing shared resources within healthcare systems.

The current limitation in the Fleet Management System's experimental setup is insightful, considering the actual path length instead of the Euclidean distance, which can enhance the cost matrix's accuracy.

In future work, we will expand the Fleet Management System (FMS) by partitioning the map into zones and incorporating restrictions based on human activities as a strategic move to enhance functionality. In addition, human-robot interaction within zones, considering safety, communication, and collaboration, is another development direction. Furthermore, the mechanisms for robots to respond to human activities and adapt their behaviour are also considered.

Acknowledgment

This research is funded by Hanoi University of Science and Technology (HUST) under project number T2022-PC-023.

References

[1] Bardram, J., Bossen, C., "Mobility Work: The Spatial Dimension of Collaboration at a Hospital" Computer Supported Cooperative Work (CSCW), An International Journal, Vol. 14, No. 2, pp. 131-160, 2005

[2] Huang, X., Cao, Q., & Zhu, X., "Mixed path planning for multi-robots in structured hospital environment" The Journal of Engineering, pp. 512-516, 2019

[3] Artem I.; Aufar Z.; Tatyana T.; Kuo-Hsien H., "Online Monitoring and Visualization with ROS and ReactJS" 2021 International Siberian Conference on Control and Communications (SIBCON), 2021

[4] Sidiropoulos A., Sidiropoulos V., Bechtsis D., Vlachos D., "An industry 4.0 tool to enhance human-robot collaboration" the 32nd International DAAAM Virtual Symposium "Intelligent Manufacturing & Automation", 2021

[5] Eduardo Guzmán O., Beatriz A., Francisco F., Raul P.r, Ángel Ortiz B., "Fleet management system for mobile robots in healthcare environments" Journal of Industrial Engineering and Management, Vol. 14, No. 1, 2021

[6] Laporte G., "What You Should Know about the Vehicle Routing Problem" NavalResearch Logistics, No. 54, pp. 811-819, 2007

[7] Janacek J., Janosikova L., Kohani M., "Modelovanie a optimalizacia" EDISvydavatelstvo ZU, in Slovak, 2013

[8] Augerat P., Belenguer J. M., Benavent E., Corberan A., Naddef D., Computational results with a branch-and-cut code for the capacitated vehicle routing problem" 1998

[9] Baldacci R., Christofides N., and Mingozzi A., "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts," Mathematical Programming, Vol. 115, No. 2, pp. 351-368, 2008

[10] Clarke G. and Wright J. V., "Scheduling of vehicles from a central depot to a number of delivery points," Oper Res, No. 12, pp. 568-581, 1964

[11] Fisher M. L. and Jaikumar R., "A generalized assignment heuristic for the vehicle routing problem," Networks, No. 11, pp. 109-124, 1981

[12] "Google Optimization Tools," [Online] Available: https://developers.google.com/optimization/routing/cvrp [Accessed 05 08 2023]

[13] Karamanos, X.; Mallioris, P.; Poulimenos, D.; Bechtsis, D. & Vlachos, D., "A ROS Tool for Optimal Routing in Intralogistics," Proceedings of the 30th DAAAM International Symposium, 2019

[14] Toris, R.; Kammerl, J.; Lu, D. V.; Lee, J.; Jenkins, O. C.; Osentoski, S.; Wills, M. & Chernova, S., "Robot Web Tools: Efficient messaging for cloud robotics," IEEE International Conference on Intelligent Robots and Systems, pp. 4530-4537, 2015