electronics

MDPI

Article

A Cloudware Architecture for Collaboration of Multiple AGVs in Indoor Logistics: Case Study in Fabric Manufacturing Enterprises

Fatih Okumuş [1,*], Emrah Dönmez [2] and Adnan Fatih Kocamaz [3]

[1] Department of Computer Technology, Malatya Turgut Ozal University, Dogansehir 44500, Turkey
[2] Department of Management Information Systems, Malatya Turgut Ozal University, Battalgazi 44210, Turkey; emrah.donmez@ozal.edu.tr
[3] Department of Computer Engineering, Inonu University, Malatya 44000, Turkey; fatih.kocamaz@inonu.edu.tr
* Correspondence: fatih.okumus@ozal.edu.tr

Received: 2 November 2020; Accepted: 25 November 2020; Published: 30 November 2020

Abstract: In Industry 4.0 compatible workshops, the demand for Automated Guided Vehicles (AGVs) used in indoor logistics systems has increased remarkably. In these indoor logistics systems, it may be necessary to execute multiple transport tasks simultaneously using multiple AGVs. However, some challenges require special solutions for AGVs to be used in industrial autonomous transportation. These challenges can be addressed under four main headings: positioning, optimum path planning, collision avoidance and optimum task allocation. The solutions produced for these challenges may require special studies that vary depending on the type of tasks and the working environment in which AGVs are used. This study focuses on the problem of automated indoor logistics carried out in the simultaneous production of textile finishing enterprises. In the study, a centralized cloud system that enables multiple AGVs to work in collaboration has been developed. The finishing enterprise of a denim manufacturing factory was handled as a case study and modelling of mapping-planning processes was carried out using the developed cloud system. In the cloud system, RestFul APIs, for mapping the environment, and WebSocket methods, to track the locations of AGVs, have been developed. A collaboration module in harmony with the working model has been developed for AGVs to be used for fabric transportation. The collaboration module consists of task definition, battery management-optimization, selection of the most suitable batch trolleys (provides mobility of fabrics for the finishing mills), optimum task distribution and collision avoidance stages. In the collaboration module, all the finishing processes until the product arrives the delivery point are defined as tasks. A task allocation algorithm has been developed for the optimum performance of these tasks. The multi-fitness function that optimizes the total path of the AGVs, the elapsed time and the energy spent while performing the tasks have been determined. An assignment matrix based on K nearest neighbor (k-NN) and permutation possibilities was created for the optimal task allocation, and the most appropriate row was selected according to the optimal path totals of each row in the matrix. The D* Lite algorithm has been used to calculate the optimum path between AGVs and goals by avoiding static obstacles. By developing simulation software, the problem model was adapted and the operation of the cloud system was tested. Simulation results showed that the developed cloud system was successfully implemented. Although the developed cloud system has been applied as a case study in fabric finishing workshops with a complex structure, it can be used in different sectors as its logistic processes are similar.

Keywords: cloud robotics; multi-AGVs collaboration; multi-task allocation; logistics planning; intelligent manufacturing

1. Introduction

With Industry 4.0 becoming widespread, the use of autonomous systems in manufacturing enterprises has become inevitable [1–3]. AGVs (Automated Guided Vehicles) that are frequently used in indoor autonomous transportation provide important advantages in performing tasks that are time-consuming, costly and risky for human health. In addition to features such as environmental perception, path planning, and behavior modelling, their adaptation to production environments compared to rail or wire guide tools is highly flexible and cost-effective [4].

In human-powered indoor logistics processes, transport tasks are assigned to any employee; however, this assignment is not based on the principles of optimizing important resources such as path, energy or time. These transport tasks can be multiple and need to be run simultaneously. Thus, for autonomous logistics, there is a need for multiple AGVs. For the multiple AGVs to work autonomously and simultaneously in collaboration, basic problems such as optimized multitasking allocation, path planning, collision avoidance, mapping, and positioning need to be solved. In traditional applications, collaboration is achieved through the principle of communicating with each other using the hardware and sensors owned by AGVs. As an alternative to traditional solutions, collaboration implemented with cloud architecture allows most of AGVs to be controlled by a central unit (cloud server). Thus, transactions that require high processing can be performed by the cloud server instead of AGVs. With a system designed in this way, multiple object control can be applied to multiple areas, even if they are in different locations.

One of the most important challenges for multiple AGVs to cooperate simultaneously is the multi-task allocation problem. The multi-task allocation has advantages such as solving complex tasks, high performance, reliability and low cost with simply designed robots [5]. In multi-task allocation, it is aimed to improve components such as cost, time, energy and security depending on parameters such as the number of goals, the priority of tasks and the position of obstacles. Multipurpose optimization approaches can be used to improve the multiple components. It is important to design this multi-purpose optimization in a form whereby tasks and other components are not independent of each other [6]. Many parameters such as the number of AGVs, their location, charge status, the weight of the load to be transported, the number of stations to be visited and the distances of these stations should be evaluated at the task allocation. Studies on multi-task allocation can be handled under two headings: exact solutions [7–9] and heuristic approach. Studies with exact solutions produce optimal results, but they do not work as fast as heuristic approaches [10]. Heuristic algorithms are frequently used to multi-task allocation such as particle swarm optimization [11–16], genetic algorithm [17–21], *k*-WTA [22], Hungarian method [23].

The multi-task allocation problem is related to the path planning problem. If the path planning step is to be performed after the task allocation step, this will affect the benefit of optimal task allocation [24]. The main purpose when assigning tasks is to minimize the total length of the paths travelled by all AGVs. When this is considered as a constraint, it turns out that path planning should also be made while assigning tasks. Path planning is calculated based on establishing a suitable path between a starting position and one or more goals. Among the studies in the field of path planning, in addition to traditional studies based on cell decomposition, potential field, and road map, there are also heuristic approaches [25–29]. Although heuristic approaches work much faster than other solution methods, they are not stable in achieving a globally optimum solution [30]. Algorithms such as A*, D*, Dijkstra, K nearest neighbor (k-NN), Genetic Algorithm, Artificial Potential Field, PSO, and their derivatives were used in the literature for path planning. In this study, the D* Lite algorithm is used in path planning. D* Lite implements D*'s strategy while looking for the most suitable path, but it works with a simple method and higher efficiency [31].

In this study, a cloud system was developed for the collaboration of AGVs used in indoor logistics. In the developed cloud system, a new method that provides optimum task allocation has been proposed. The method firstly calculates the optimum paths between AGVs and goals by avoiding static obstacles. Then, an assignment matrix based on permutation possibilities is created and the optimal combination

of assignments is determined according to the minimum total path. Assignment combinations are divided into 2-dimensional pieces according to the nearest neighborhood principle to solve the problem of the increased complexity of permutation calculation in environments with many AGVs and goals. Since the computational complexity of these two-dimensional pieces of the matrix is low, the cost of the assignment is reduced and the processing speed is increased. In the selection of the most suitable assignment combinations, one-to-one assignment and battery levels of AGVs are determined as constraints. A safe area approach has been proposed to prevent the robots from colliding with each other. When another AGV enters this safe area, new local paths have been created dynamically.

The cloud system was developed by modelling the fabric finishing enterprise of Çalık Denim R&D Center within the scope of a university-industry cooperation project. Simulation software has been developed and assets within the enterprise are simulated in accordance with the real map. As a result of simulation experiments, it has been observed that the system works correctly and stably.

Related studies examined within the scope of the study are given in Section 2. In Section 3, the case study is introduced and the problem is described. In Section 4, materials and methods developed within the scope of the study are given. Simulation results and observations are explained in Section 5. In the Section 6, discussion and results are mentioned.

2. Related Works

Schillinger et al. [32] presented a framework that optimizes autonomous action behavior for a group of robots. In their study, it was stated that they allocated separable tasks to the existing robots in the most appropriate behavior, without requiring a clear representation of tasks or the calculation of all task execution costs. Turner et al. [33] focused on the problem of maximizing task allocation in time-limited scenarios of multi-robot systems. They claimed that with the PI-MaxAss (their proposed method for the solution of this problem), they effectively reduced costs by changing task assignments. They tested their method with simulation and stated that there was a noticeable increase in performance measured as the total number of assigned tasks. Du et al. [34] used Warshall-Floyd to search for the most convenient route between two random points in their application for textile workshops. By using the genetic particle swarm optimization algorithm (GA-PSO), they accelerated the convergence rate of the algorithm with the time-based particle iteration mechanism by creating the mathematical model of the path planning problem. Afrin et al. [35] proposed a cloud architecture for managing multiple robots, which are frequently used in the industry. With an Edge Cloud-based system, they focus on the solution of simultaneous production, energy consumption, and cost optimization problems. For the multitasking problem, they used the redesigned NSGA-II algorithm working on the cloud. Liu et al. [36] pointed out the impact of cloud computing capability on the management of multiple robots in their work. They enabled the management of multiple robots integrated into the cloud system developed by a task assignment algorithm based on reinforcement learning (RL). They showed the benefit of the method they proposed by comparing it with the Greedy Search algorithm. Chowdhury and Maier [37] have proposed a task allocation strategy based on parameters such as the robot's position, remaining energy, and task execution time. They have established this strategy based on data from a cloud system and neighboring robot. Yan et al. [38] stated that the basis of intelligent production environments is the integration of cloud computing and industrial robots. However, they considered the lack of flexible load sharing and excessive resource consumption of existing algorithms in these environments. In their study, they developed a cloud architecture for group learning and shared their results. Wang and Liu [39], stated that collaboration is achieved by interaction between robots instead of a central operator in many multi-robotic applications, and cloud robotic applications offer an alternative approach focused on central control. They stated that a resource distribution strategy is required for such a central application to use resources effectively, and they proposed the link quality matrix (LQM) method for this problem. They stated that this method performed better than the recent algorithms of resource allocation. Clark [40] presented a Probability Roadmap (PRM) motion planning algorithm based on Dynamic Robotic Networks, a multi-robot coordination platform that works with limited sensing and

communication between robots. According to their results, it is stated that the DRN platform shows that it works well even if the network becomes complex or broken. Solovey and Halperin [41] present a simple and natural extension of the multi-robot motion planning problem in which robots are divided into groups (colors), thus expressing that robots are interchangeable in each group. In the developed algorithm, there is a new approach wherein the k-color problem is reduced to a large number of discrete multiple robot motion planning problems. Ma et al. [42] present a decentralized multi-agent motion planning method for air robots moving between 3D polygonal obstacles. The algorithm combines a method based on barrier functions (for low level (local) coordination and control) and the priority A* algorithm (for high level (global) planning). With the algorithm, they calculated the paths by redefining the waypoints when appropriate, while ensuring the collision-free movements of the agents. Draganjac et al. [43] provide an algorithm for the decentralized control of multiple automated guided vehicles used for transportation in industrial operations and warehouse environments. It is stated in their studies that the proposed control strategy is not only limited to multiple AGV systems but is also applicable for decentralized coordination of mobile robots in various other multi-robot applications. Mousavi et al. [17] developed a mathematical model and integrated it with evolutionary algorithms (Genetic Algorithm—GA, Particle Swarm Optimization—PSO and hybrid GA—PSO) to optimize AGVs' task planning to minimize the completing time. According to their results, they claimed that the three algorithms have proven their applicability in completing time and reducing AGV numbers. It is emphasized that the best result is obtained with the hybrid system. Dewangan et al. [44] investigate some of the methods that determine the choice of multi-robot priority path planning techniques. They have analyzed the changes in the path planning approach according to various configurations by addressing the difficulties existing in multiple robot systems. Digani et al. [27] provide an optimization strategy to coordinate an AGV fleet travelling on predefined temporary road maps. It is stated that the specific aim of the study is to maximize traffic flow in an automatic warehouse by minimizing the time spent. They also focused on preventing AGVs from colliding with each other and overcoming complex traffic models. Oh et al. [45] proposed a system to work with multiple agents for collaboration in scheduling tasks using an optimal task allocation algorithm based on particle swarm optimization (PSO). In their work, they changed the conventional PSO-based algorithm using graph theory infrastructure. They showed the performance of the proposed algorithm in simulation. Guerrero et al. [46] state that choosing the best task to perform (task assignment problem) is one of the main problems in multi-robot systems. In their work, they show that, concerning a task assignment problem, fuzzy Markov chains can reach a stationary stage in a limited number of steps. They say that fuzzy Markov chains are better at modelling motion prediction. Trigui et al. [47] address the problem of assigning a team of autonomous robots to goal areas in the context of a disaster management scenario. It has been stated that this problem may arise as more than one travelling salesman problem (GSP), which several robots must visit at designated locations. The researchers stated that by providing an analytical hierarchy process (AHP)-based approach to this problem, they minimize three goals: total travel distance, maximum tour, and deviation rate.

Existing multiple robot planners are typically based on disaggregated planning or the unified search approach in the common status area of all robots. Combined approaches can find the optimal solution [48,49], but the worst time complexity increases exponentially with the number of robots with potential conflict in the environment. In this case, if there are more than a few robots, coordination between robots becomes difficult. In this study, a cloud-based system was developed for multiple AGV collaboration, taking into account the approaches in existing related works. The modular features in the system are quite high. These modular features are compared with prominent related works and the results of the comparison are given in Table 1.

Table 1. Comparison of the developed system in prominent related works.

	Chowdhury and Maier [37]	Afrin et al. [35]	Turner et al. [33]	Schillinger et al. [32]	Du et al. [34]	Our Work
Environment Assets Definition	✓	✓	✓	✓	✓	✓
AGV Charging Stations	-	-	-	✓	-	✓
Mapping	✓	-	-	✓	-	✓
Multi-AGV	-	✓	✓	✓	-	✓
Multi-Task Allocation	✓	✓	✓	✓	✓	✓
Simultaneous Task Execution	✓	✓	✓	✓	-	✓
Coordination Type	Centralized	Centralized	Decentralized	Centralized	Decentralized	Centralized
Monitoring	-	-	-	-	-	✓
Simulation	-	✓	✓	✓	✓	✓
Path Planning	-	-	-	STAP	Warshall-Floyd + GA-PSO	D* Lite
Multi-AGV Collaboration	-	✓	-	✓	-	✓
Battery Optimization	-	-	✓	✓	-	✓
Collision Avoidance	-	-	-	✓	-	✓
Pseudocode	✓	✓	✓	✓	-	✓
Cost Optimization	✓	✓	✓	✓	✓	✓
Cloud-based Solution	✓	✓	-	-	-	✓
Position Tracking	-	-	-	-	-	✓

3. Case Definition

Indoor logistics refers to the process of transporting products or other materials to maintain the supply chain within the enterprise building. In some industrial enterprises, the products to be delivered are subjected to some sequential processes before obtaining their final form. The transportation of semi-manufactured fabrics can be complex due to the increase in the number of products in the case of simultaneous production in these enterprises. In fabric finishing enterprises, fabrics are subjected to some processes such as pre-finishing, coloring, dyeing, printing, and sanforizing processes on finishing machines. Batch trolleys, which allow mobility and can be easily connected to the finishing machines, are used to transmit the fabrics to these machines. The fabrics are wrapped in rolls in this transfer vehicle and transported using towing tractors and placed in the finishing machine to be processed. In Figure 1, a sample fabric loaded batch trolley and towing tractors are shown.

(**a**) (**b**) (**c**)

Figure 1. (**a**) Fabric loaded batch trolley, (**b**) Electric Towing Tractor, (**c**) Human-Powered Towing Tractor.

Batch trolleys are moved on the flat floor corridors within the enterprise and are navigated to the in-port/out-port of finishing machines. An example navigation corridor is shown in Figure 2a and finishing machine is shown in Figure 2b. A typical fabric finishing enterprise can have multiple finishing machines within multiple corridors.

(**a**) (**b**)

Figure 2. (**a**) Navigation corridor, (**b**) Finishing machine.

In many finishing enterprises, batch trolleys are generally used by human operators. Multiple productions are carried out in these enterprises and multiple batch trolleys are navigated simultaneously. The fact that the logistics operators leave these fabrics to the faulty machine has the potential to cause serious damage to the manufacturing enterprises. Besides, accidents during transfer also threaten the health of employees. In this study, it is aimed to control the disruptions in orders and increase work-time efficiency by carrying out these processes using AGVs.

4. Materials and Methods

4.1. Enterprise Assets and Working Model

To carry out indoor logistics activities autonomously, assets directly affecting logistics within the facility have been identified and a working model that uses these assets in harmony has been created. Thus, the decision mechanism of the autonomous system has been determined according to this model.

Çalık Denim factory finishing facility operating in Malatya 2nd Organized Industrial Zone was used for the modelling of the working environment. The working environment is modelled by using the 2-dimensional real map and assets of the finishing enterprise. According to the working model, the assets in the environment are:

- K_i: *Semi-manufactured fabrics to be processed*
- M_i: *Machines which fabrics are subjected to specific finishing processes*
- T_i: *Batch trolleys providing mobility to fabrics*
- W_i: *Waiting stations for batch trolleys*
- A_i: *AGVs that provide navigation of batch trolleys (AGV)*
- C_i: *Charging stations for AGVs*
- S_i: *Starting ports where the semi-manufactured fabrics enter the finishing enterprise*
- D_i: *Delivery ports of finished fabrics*

Depending on the fabric type, the processing time (t_m) in any finishing machine can vary. The length of the fabric (l_k) is also effective in calculating the time spent in the finishing machine. Batch trolleys provide mobility in transporting the fabrics to the target machine and provide flexibility for the finishing process with its rotating structure. If the machine in which a fabric is to be processed is busy, the batch trolley is taken to the waiting station. Batch trolleys waiting in the queue according to the order priority are moved back to their goals when the machines become available. During the processing, the parts of the fabric that are finished should be wrapped in another empty batch trolley at the out-port of the machine. Therefore, if a batch trolley is directed to the in-port of a machine, an empty batch trolley should be directed to the out-port of the machine at the same time.

In the study, modelling was made on the use of AGVs instead of batch trolley towing tractors. Charging units have been designed in the working environment for the energy needs of AGVs.

Starting ports are created for the new fabrics and delivery ports for finished fabrics. Figure 3 shows a minimized part of the modelled fabric finishing enterprise. As can be seen in the Figure 3, K_1 and K_2 are new fabrics. T_1, T_3, and T_5 are batch trolleys that unload fabric at the in-port of finishing machines. T_2, T_4, and T_5 are batch trolleys whose fabric is loaded up from the out-port of the machines. T_7 and T_8 are batch trolleys taken to the destination by AGVs. T_9 and T_{10} are fabric-loaded or empty batch trolleys at the waiting station. While AGV_1 and AGV_2 take batch trolleys to the destination, other AGVs are charging.

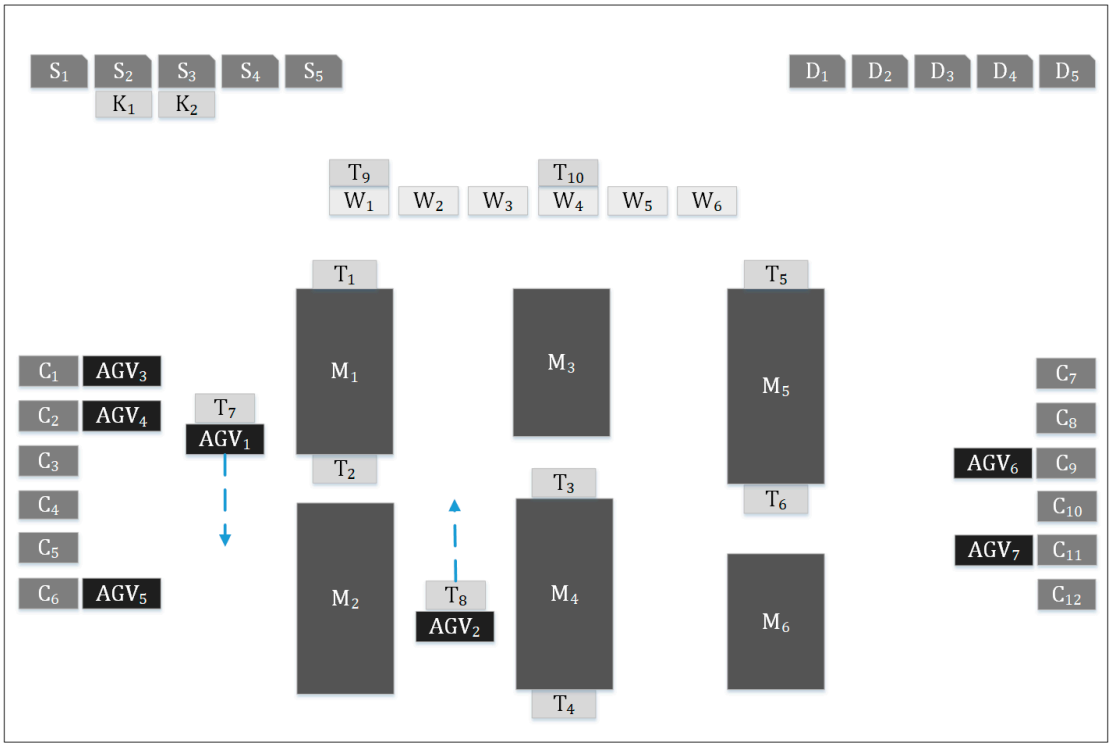


Figure 3. A minimized part of the modelled fabric finishing enterprise.

When the finishing process is started for fabric, the machines to be processed and the orders of these machines are determined. An example study scenario for this situation is shown in Figure 4. This example shows the machines and sequencing where K_1 and K_2 fabrics will be processed simultaneously until they reach the delivery point. According to the scenario stated in Figure 4: The movement plan for K_1 fabric is $\{S_1, M_1, M_5, M_6, M_7, D_1\}$ and the movement plan for K_2 fabric is $\{S_2, M_3, M_2, M_4, M_5, M_8, D_2\}$.

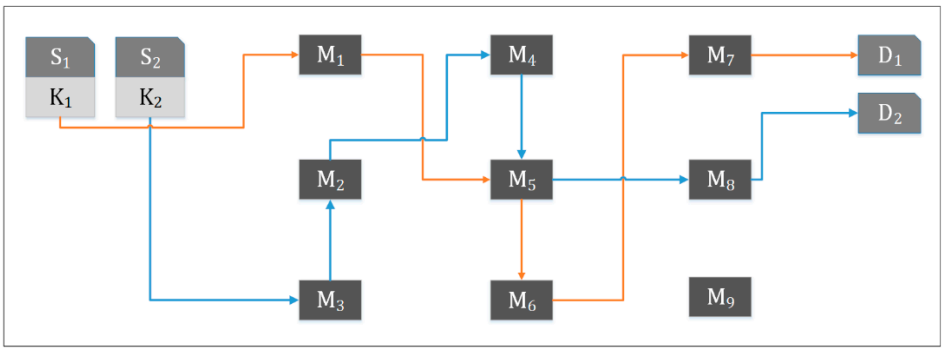


Figure 4. A scenario of two different fabrics being sanforized simultaneously in the environment of 9 finishing machines and taken to the delivery point.

4.2. Developed Cloud Robotic Architecture

There are two different approaches, central and decentralized control, to ensure collaboration in multiple robot systems. In the decentralized collaboration approach, a robot in the environment is identified as the leader and this leader robot ensures coordination based on sensor data from other robots. In the central approach, servers in different locations from robots provide coordination. An application running on the server accesses and assigns tasks to AGVs via the wireless network or the internet. In this study, a cloud system application based on the central collaboration approach has been realized for the control of the logistics assets in the environment. The advantages of the central collaboration over the decentralized collaboration are as follows:

- The leader robot that provides coordination in a decentralized collaboration approach must have high computational ability. High hardware capacity is also required for high computation. However, since the computation unit is not a robot in the central collaboration approach, high-capacity hardware is not required for the robots.
- In decentralized collaboration, it is necessary to choose a new leader robot when the leader robot is out of charge or fails to operate. For the collaboration algorithms to be executed stably, the hardware properties of the new leader must be the same as the previous one. In this case, it reveals the necessity for all robots in the environment to have the same high capacity hardware.
- In the decentralized collaboration approach, communication is only within the limits of the robot's hardware. The robot can send signals to other robots in the environment, depending on the power of the communication technology used, within a certain region. However, in the cloud-based central collaboration approach, even the AGVs in many different locations can interact by dint of the internet infrastructure.

The cloud system was developed using the Django web framework by Python language. Data was stored in relational tables using PostgreSQL. Communication between the cloud server and client robots has been provided by WebSocket and Restful APIs. AGVs must be Robot Operating System (ROS)-enabled to be able to execute commands sent by the cloud server. ROS is a Linux-based and open source middleware software with a large library that can run all the functions of the robots it supports. Minicomputers with ROS libraries (Raspberry Pi, etc.) can be used for AGVs that do not have a Linux operating system.

The cloud system consists of mapping, position tracking, multiple AGV collaboration, web-based management and monitoring stages. The developed cloud robotic architecture is shown in Figure 5.

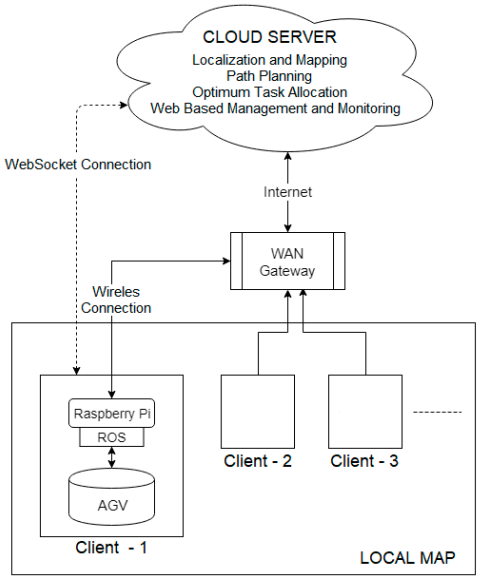


Figure 5. Developed cloud robotic architecture.

4.2.1. Cloud-Based Mapping

In the developed cloud robotic architecture, the map of the environment in which the robot moves must be known in advance. This is important for the path planning and multi-task allocation stages to work. During the map creation process, the locations of static assets in the environment can be transmitted to the cloud system according to the type of asset. Restful APIs have been created in the cloud robotic architecture for the transmission of these data. REST or RESTful web services are among the most common web services used today due to their simplicity. REST services are robust, use minimal bandwidth and are suitable for connecting to cloud services. Similar to HTTP, the device can handle server requests with four request methods (GET, PUT, POST and DELETE). In the study, RESTful web services were used to perform the following operations:

- Saving, updating, listing and deleting of geographic information of assets in a selected map environment,
- Adding fabric orders to the cloud system with its processes,
- Listing of fabrics transferred to the delivery point through all processes.
- Receiving statistical data on fabrics and orders daily, weekly, monthly and yearly.

4.2.2. Position Tracking

To correctly direct AGVs in the working environment, its current position has to be known. There are many studies suggested in the literature to detect the current position of AGVs. In these studies, RFID, QR-code, 3-colors, Bluetooth and Wi-Fi-based technologies have been used. In the position tracking module, standard data structures that can be integrated into all positioning systems have been created. This data structure includes the 2D location information and qualitative features of the AGV on the known map. In addition, by creating grid-shaped nodes on the map, position information is matched with the position of these nodes. The number of nodes for each map can be determined dynamically. These nodes are used in path planning and multitasking algorithms. Figure 6 shows the path planning simulation in an environment with an area of 20 × 14 m^2 and 280 nodes.

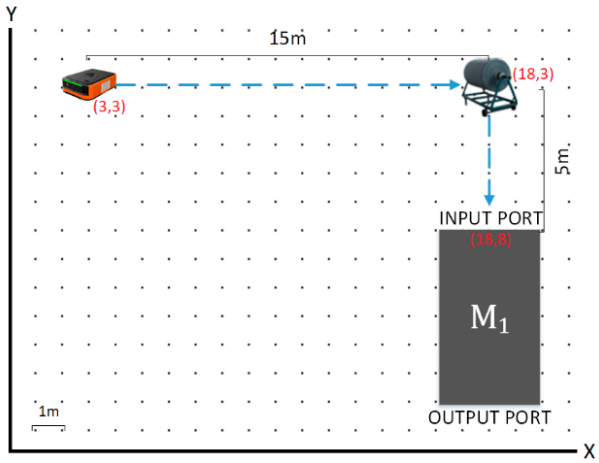


Figure 6. Path planning with instant position tracking in an environment of 20 × 14 m^2.

Position information is small size data, and the transmission of this data should be triggered as soon as the position of the AGV changes. The most suitable technology for this process is WebSocket that provides a two-way communication channel between clients and servers via the internet or local network. This channel is established on a TCP socket and remains open for the uninterrupted exchange of data between endpoints. WebSockets use their own protocols that were standardized by Internet Engineering Task Force (IETF) [RFC 6455] in 2011. When a WebSocket request is sent to the server, the client requests to change the protocol from HTTP to WS, and if the server supports it, this change

will be made. Thus, a two-way WebSocket connection is created that allows real-time messages to be sent between the client and the server. These messages have been stored on REDIS, so clients on all terminals of the WebSocket channels could access these messages at the same time. The copies of the messages have also been stored to the relational database simultaneously. REDIS is an open-source key-value-based NoSQL storage software developed and growing in popularity. It provides great advantages in CPU and memory usage as it stores data in JSON format in the memory instead of the disk. When the channel is closed, the messages created on REDIS are automatically deleted.

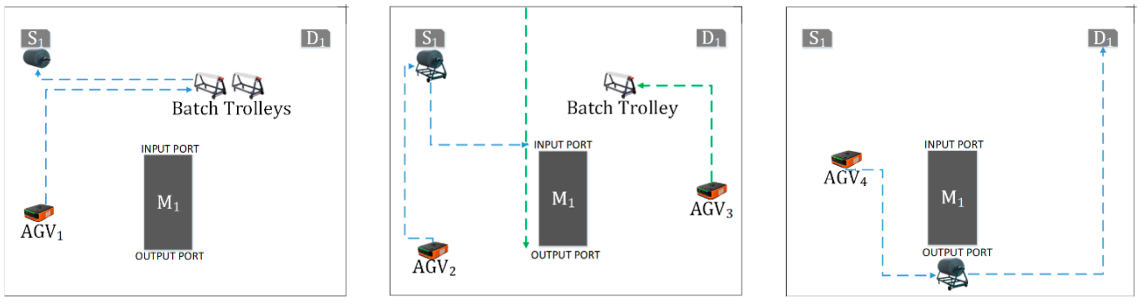
4.3. Collaboration Module for Multi-AGVs and Multi-Tasks

The collaboration of multiple AGVs is one of the major challenges to making indoor logistics as autonomous as possible. The proposed working model plays an important role in the healthy implementation of the collaboration module. The collaboration module consists of task definition, battery management and optimization, path planning, selection of the most suitable batch trolley, optimum task allocation, and collision avoidance stages.

4.3.1. Task Definition

To make the task allocation properly, the behavior patterns of the tasks must be defined properly. The making ready of the fabrics by processing them on multiple machines requires multiple tasks. The types of these multiple tasks are as follows and illustrated in Figure 7:

- When a fabric is included in the system, the process of transmitting an empty batch trolley to this fabric: *starting-task*.
- The process of transmitting the fabric to the input port of any machine where it will be treated: *input-task*.
- When the fabric enters the finishing machine and starts to process it, the fabric sections that are finished should be wrapped in another batch trolley at the exit of the machine. The process of transmitting an empty batch trolley simultaneously to the output port of the machine when an input task begins: *output-task*.
- When the machine to which the fabric will be transmitted is busy, the process of transmitting the batch trolley to the nearest waiting station: *waiting-task*
- The process of transmitting the product from the last machine to the delivery point after all processes are finished: *delivery-task*.
- When an AGV's battery drops below the critical level, the transmission process of the AGV to the nearest charging station: *charging-task*



(**a**) (**b**) (**c**)

Figure 7. The scenario of the fabric included in the system at the S_1 point, processed on the (M_1) machine and transmitted to the (D_1) delivery point. (**a**) Starting task. (**b**) Input and output tasks for processing in M_1. (**c**) The delivery task that transmits the fabric from the output port of M_1 to D_1 point.

4.3.2. Battery Management and Optimization

The battery state information of AGVs was sent to the cloud robotic system by using WebSockets. Thus, when the battery state of AGV drops below a critical level (l_c), AGV is closed to assignments of tasks and navigated to the nearest charging station. If this situation happens when AGV performs a task, the task is assigned to another AGV and discharged AGV is navigated to the nearest charging station. The l_c critical battery level is the minimum amount of energy that will allow an AGV to move along the maximum unobstructed path between the two furthest locations in the environment with the maximum amount of load it can carry. If new static obstacles are added to the environment, this changes the length of the maximum path, and l_c also changes. Thus, critical battery level is determined automatically according to the map status and is used in the optimum task allocation. The l_c critical battery level is calculated as in Equation (1).

$$l_c = \delta * P_{max} \; + \; \mu * P_{max} * W_{max} \tag{1}$$

In Equation (1), δ is the amount of energy that an AGV spends at a distance of 1 m. μ indicates the amount of energy that an AGV needs to spend to carry a 1 kg weight at a distance of 1 m. W_{max} is the maximum weight a batch trolley can transfer. μ and W_{max} values are defined as parametrically in the cloud system. P_{max} represents the maximum length path on the environment map and is obtained by summing the Euclidean distances between two furthest points in Equation (2).

$$P_{max} = \sum_{i=1}^{k-1} \sqrt{(x_i - x_{i+1})^2 + \; (y_i - y_{i+1})^2} \tag{2}$$

In Equation (2), k is the number of elements of the $[(x, y)]$ array in the 2-dimensional plane of the optimum path, which is determined by the path planning algorithm. The array $[(x, y)]$ is created with the longest collision-free path plan in the environment.

4.3.3. Path Planning with D* Lite

In this study, the D* Lite algorithm, which is widely used in robotic applications, one of the effective methods in this field and derived from the D* algorithm, was used to solve the path planning problem. D* Lite is an incremental heuristic algorithm, which combines A* and Dynamic SWSF-FP ideas by Sven Koenig and Maxim Likhachev and it uses LPA* to find a new best path of a unit as the graph changes [50]. D* Lite searches for the collision-free path, starting from the goal node s_{goal} to the start corner s_{start}. The algorithm stores the estimated distances between each vertex to the $g(s)$ and $rhs(s)$ values. The *rhs* values are estimated by the correlation in Equation (3).

$$rhs(s) = \begin{cases} 0 & s = s_{goal} \\ min_{s' \in Succ(s)}(g(s') + c(s, s')) & otherwise \end{cases} \tag{3}$$

In Equation (3), s' is the neighbor of s and the $Succ(s)$ set is the vertex formed by expanding from s. $c(s, s')$ is the cost of the transfer from s to s'. D* Lite uses a priority queue (U) structure to store all inconsistent vertices. The algorithm performs the following steps to achieve the optimal path plan:

- Initially, all values of g and *rhs* parameters are set to infinite, and s_{goal} with $rhs = 0$ is put in U.
- First, u vertex with the least priority k value is selected from U.
- If $g(u) > rhs(u)$, the *rhs*-value of u is set as g-value and u is removed from U.
- Then all *rhs* values of the precursors of u are changed and added to U if locally inconsistent.
- If $g(u) \leq rhs(u)$, u is set to infinite and inserted into U if they are locally inconsistent.

- These steps continue until the s_{start} point is locally consistent and the k value of all the vertexes remaining in U is greater than the s_{start} value.

The flowchart of D* Lite algorithm is shown in Figure 8.

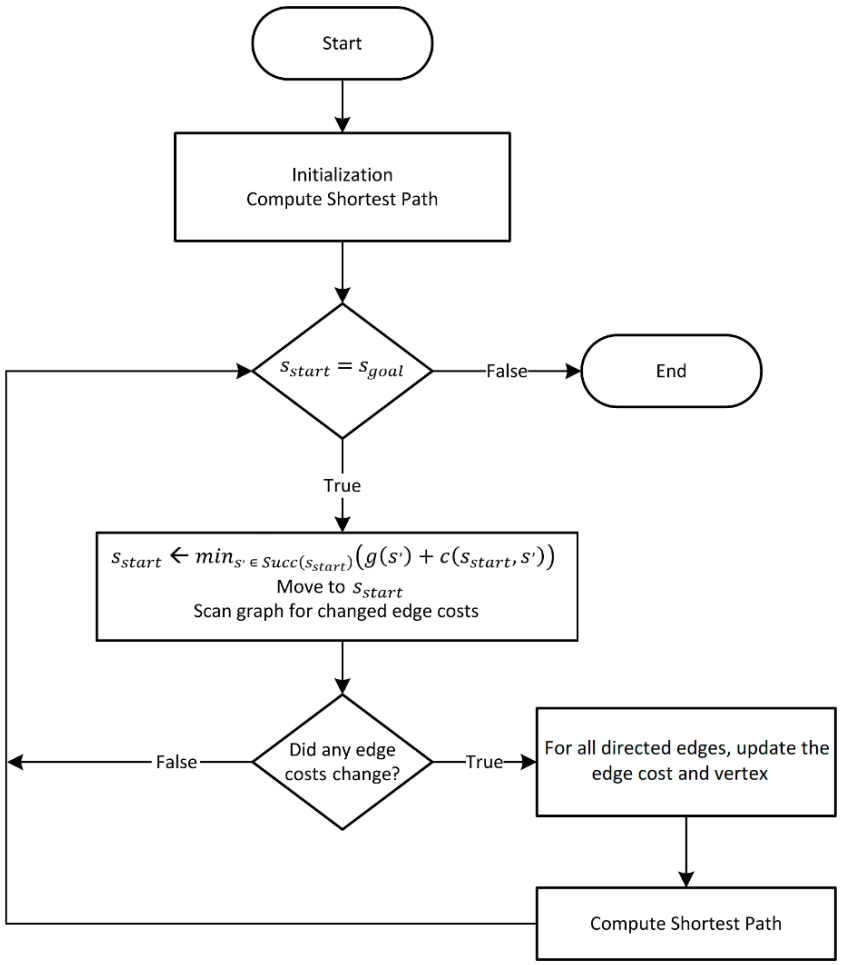


Figure 8. D* Lite algorithm flow chart.

4.3.4. Choosing the Most Suitable Batch Trolley

When choosing a batch trolley to be assigned to a task, the one closest to the location where this task will begin should be preferred. However, the busy state of this batch trolley should also be checked. If the selected batch trolley is busy, it may be necessary to choose another one, but if the busy time is short, choosing the first batch trolley can produce an optimal result. In this case, the busy time must be calculated. The busy time (t_m) is calculated in seconds by the Equation (4).

$$t_m = w/h_m - (t_c - t_s) * h_m \quad (4)$$

In Equation (4), w is the length of the fabric on the busy batch trolley that is processed in the machine m. The parameter h_m is the length of the fabric processed in the machine m in 1 s. t_c is the current time and t_s is the time when the m machine starts to become busy. t_m is also used when a fabric is transmitted to the machine to be processed. In this case, if the machine's busy time is more than the transmit time, the fabric is transmitted to the nearest waiting station.

While selecting the batch trolley to be assigned to $task_i$, only the empty ones were not selected. It may be more advantageous to select any batch trolley that already performs another task if its deadline is short. Therefore, the transmit times of all batch trolleys to the start position of $task_i$ should

be calculated. T_i, the time that batch trolley will transmit to the start position of $task_i$ is calculated by the Equation (5).

$$T_i = Len(s_{current},\ s_{goal_{old}}) * \frac{1}{v} + t_m + Len(s_{start},\ s_{goal}) * \frac{1}{v} \tag{5}$$

In Equation (5), $s_{current}$ is the current position of the i^{th} batch trolley. $s_{goal_{old}}$ is the arrival position of the $task_c$ (the current task of bach trolley that it has not yet completed). $Len(s_{current},\ s_{goal_{old}})$ is the length of the optimum path plan from the $s_{current}$ to the $s_{goal_{old}}$. Using the [(x, y)] points from the path plan created with D* Lite, v is the speed of the AGV. t_m is the time that the batch trolley will spend in the finishing machine of the $task_c$ it is completing and is calculated as in Equation (4). s_{goal} is the starting position of $task_i$.

If the batch trolley is busy for $task_i$; then $s_{start} = s_{goal_{old}}$. If the transfer tool is not busy, s_{start} is the current position of the batch trolley, as is $Len(s_{current},\ s_{goal_{old}}) = t_m = 0$. This selection method is given in Algorithm 1.

Algorithm 1 Choosing the most suitable batch trolley

```
01: function Find Most Suitable Batch Trolley ()
02:     V ← list of all vehicles
03:     B_v ← ∅
04:     B_p ← P_max * v
05:     for all v ∈ V do
06:         time = 0
07:         if v is busy then
08:             s_local ← position(v)
09:             //Calculate path from current position to the end point of task (s_task)
10:             time ← GetOptimumPath (s_local, s_task) *v
11:             if v have a work station task then time ← time + t_m
12:                 s_start ← s_task
13:         else
14:             s_start ← position(v)
15:         time ← time + GetOptimumPath (s_start, s_goal) *v
16:         if B_p > time then B_v ← v, B_p ← time
17:     return B_v
```

4.3.5. Optimum Multi-Task Allocation

To optimally assign AGVs to multi-tasks, the total cost of all assignments must be minimal. This cost is calculated with a multi-fitness function that minimizes total path and time. Usually, minimizing the path also minimizes time in multi-task robot allocation. However, in some cases, minimizing the path may increase the time. For example, when assigning a task, the AGV in the most suitable position may be performing another task at that moment. In this study, a multi-fitness function is presented, in which both the total path and the total time are minimized. This multi-fitness function is shown in Equation (6). The function has a significance coefficient (λ) that can be determined dynamically for path and time.

$$f(R,\ G) = \lambda_1 * \sum_{i=1}^{a-1} Len\left(s_{r_i},\ s_{g_i}\right) + \lambda_2 * \sum_{i=1}^{a-1} T_i \tag{6}$$

In Equation (6), $R = \{r_1, r_2, r_3, \ldots, r_n\}$ and it is the array of AGVs. $G = \{g_1, g_2, g_3, \ldots, g_m\}$ and it is the array of goals that consist of the start positions of tasks. The sum of λ significance coefficients equals 1 and can be defined parametrically in the cloud system. a is the number of task assignments of AGVs. $Len\left(s_{r_i},\ s_{g_i}\right)$ is the length of the optimum path between the AGV and the start position of the task. T_i is the time AGV spends to arrive at the starting position of the task and is calculated as in Equation (5).

For the optimum allocation of multi-AGVs to multi-tasks simultaneously, the value of $f(R,\ G)$ should be minimal. To minimize f, an assignment matrix was created, containing all possible mappings of AGVs and goals. The purpose of this matrix is to detect the row with minimum cost and suitable match. Before matching between R and G, these two sets were kept in C_{left} and C_{right} intermediate sets as follows:

$R = \{r_1, r_2, r_3, \ldots, r_n\}$, $G = \{g_1, g_2, g_3, \ldots, g_m\}$
If $n \geq m$ then $C_{left} = R$, and $C_{right} = G$
Else $C_{left} = G$, and $C_{right} = R$

Since the C_{left} determines the number of assignments that can be made simultaneously, the above operations have been performed. This means: If the number of AGVs is lower than the number of goals, the number of matching sets can be as many as the number of AGVs, otherwise it can be as much as the number of goals. Thus, a combination matrix was created with the C_{left} to the left of the matching and the C_{right} to the right of the matching. Then, a permutation matrix was created using all matches with unique assignments in each row. Figure 9 shows the creation of the permutation matrix for the $R = \{r_1,\ r_2,\ r_3,\ r_4\}$ and $G = \{g_1,\ g_2,\ g_3\}$ given as examples.

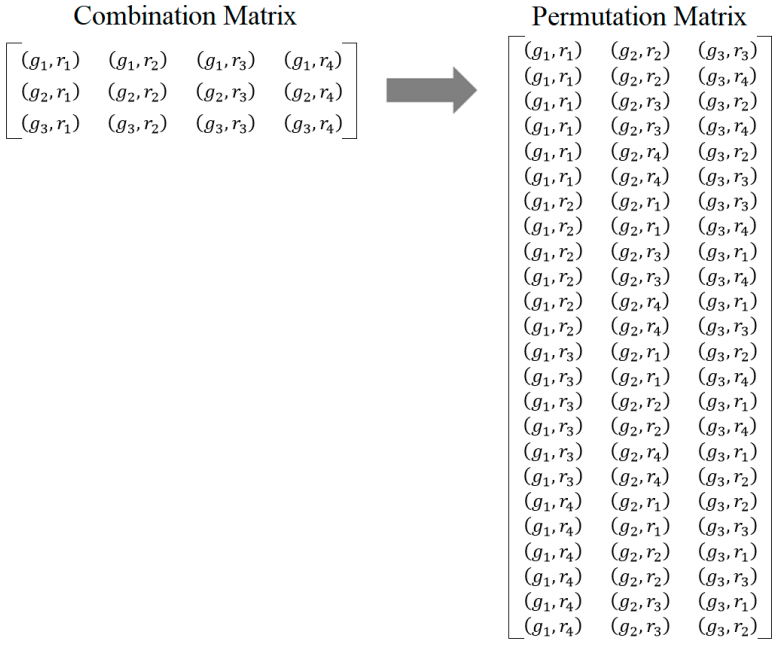
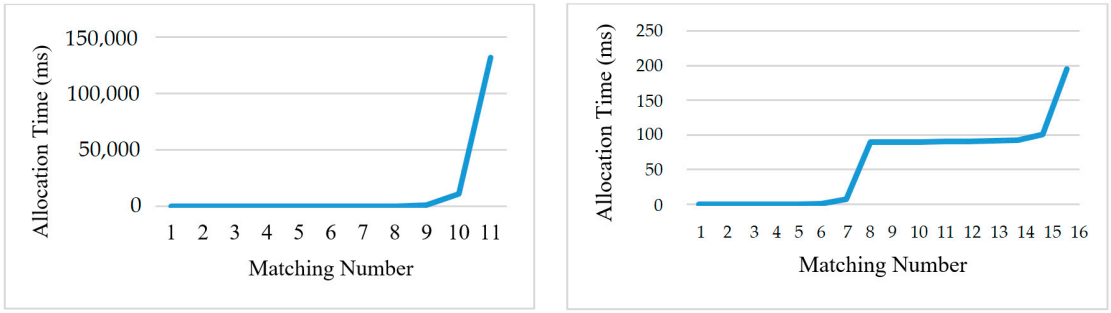


Figure 9. Creating permutation matrix for $R = \{r_1,\ r_2,\ r_3,\ r_4\}$ and $G = \{g_1,\ g_2,\ g_3\}$.

Along with the permutation matrix, all matching possibilities for R and G are also determined. Thus, the row with the minimum sum of the lengths of each matching can be selected. However, as the number of AGVs and goals increases, the computational complexity of the permutation matrix also increases. To overcome this problem, AGVs and goals were classified into φ element clusters using the k-NN method. When the φ value is kept too high, the assignment time may be longer depending on the computing ability of the computer. When this value is too low, the optimum solution is avoided. Therefore, it is necessary to keep this value at an optimum level. φ value can be changed dynamically in the cloud system since the calculation times are variable in computers with different hardware. As can be seen in the graph in Figure 10, assignment time increases exponentially when AGVs and goals are not classified. However, when they are classified with k-NN, the assignment time is much shorter. In Figure 10, the number of matching indicates the number of AGVs and goals. For example, if the number of matching is 5, the assignment time was calculated in the environment with 5 AGVs and 5 goals.



(**a**) (**b**)

Figure 10. Assignment times by the number of matching. (**a**) The assignment time obtained without the classification. (**b**) When taken as $\varphi = 8$, assignment times obtained by K nearest neighbor (k-NN) classification.

Choosing the row with the minimum total path in the assignment matrix is not sufficient alone for optimum task allocation. Whether the energy that the AGVs will spend along the path to the goal they match is sufficient is also checked. If any row in the assignment matrix has an AGV whose battery level is not sufficient for the distance it matches, this row is deleted from the assignment matrix. Thus, the allocation of tasks is realized by searching at the minimum one among the remaining rows. The pseudo-code of the optimum multi-task allocation is presented in Algorithm 2.

Algorithm 2 Optimum Multi-Task Allocation

```
function getOptimumAllocation ()
    C ← ∅, P ← ∅, C_left ← ∅, C_right ← ∅
    if length(R) > length(G) then C_left ← G, C_right ← R
    else C_left ← R, C_right ← G
    D ← Create sub-matrices with length φ according to nearest neighborhoods
    for d in D do
        C ← Create a combination matrix from d_left and d_right
        P_d ← Create a permutation matrix from C
        P.Append(P_d)
    Remove rows from P if energy level does not meet the cost
    Calculate the total cost of each row of P by summing the costs of each assignment in rows.
    Return first row with the lowest total cost in matrix P
```

For the collaboration of multi-AGVs according to the working model, it is necessary to determine the task states first. These task states play an important role in the collaboration of AGVs. Task states are as follows:

- S_1: *Task created*
- S_2: *First task for initializing*
- S_3: *Task expected to run,*
- S_4: *Task was run.*
- S_5: *AGV moves towards the batch trolley,*
- S_6: *AGV loads the batch trolley,*
- S_7: *AGV moves towards the goal,*
- S_8: *AGV arrived at the destination. It leaves the load,*
- S_9: *Task Completed.*

When a fabric is registered in the cloud system and a motion plan like in Figure 4 is created, a task is defined for each part of the motion and their status is set to S_1. A sequence number is given for each task. Then the status of the first task is set to S_2. When the status of this task is S_9, the status of the next task is set to S_2. This process continues until all the tasks are completed. For tasks with a status S_2, the "choosing the most suitable batch trolley" algorithm is run. When the nearest batch trolley is assigned to the fabric, the status of the task is changed to S_3. To transmit the fabrics in the S_3 state towards the goal, the most suitable AGV is selected. If there is more than one S_3 task at the same time, the optimum multi-tasks allocation algorithm ensures that the most suitable AGVs are selected for all tasks. When the most suitable AGV for the task is selected, the status of the task is updated to S_4. The path plan is sent to the selected AGVs via WebSockets. If the AGV transmits a batch trolley to a finishing machine and this machine is busy, it must be transmitted to the nearest empty waiting station. In this case, the status of the current task is updated to S_1, and a new task is created to take the AGV to the nearest waiting station, and the status of this task is set to S_4. When the batch trolley reaches the waiting station by AGV, the status of the previous task is updated to S_3. The pseudocode of this module that provides this collaboration is given in Algorithm 3.

Algorithm 3 The Collaboration of Multi-AGVs and Multi-Tasks

```
01: function main ()
02:     F ← task list of status is S2
03:     for all t ∈ F do
04:         v ← FindMostSuitableVehicle (position(t))
05:         Create a new task for taking transport vehicle to task point and set status S3
06:         Update v as busy
07:         Update status of t to S1
08:     W ← Task list that status field is S3
09:     R ← AGV list which is not busy and the battery level is above the critical value
10:     L ← ∅ //Allocation list
11:     for all t ∈ W do
12:         for all r ∈ R do
13:             if the type of task is "transfer" then
14:                 v ← FindMostSuitableVehicle (position(output port of workstation))
15:                 assign v to t
16:                 update v as busy
17:                 path ← GetOptimumPath (position(r), position(v))
18:                 L.Insert(r, t, path)
19:     A ← getOptimumAllocation(L)
20:     for all a ∈ A do
21:         if the goal of a is a busy work station then
22:             p ← find nearest waiting station to the position of the batch trolley
23:             Update goal position as p
24:             a.path ← GetOptimumPath (position(vehicle), p)
25:             Update status of the task as S3
26:             Update start position of the task
27:             Increase the priority level of the task
```

4.3.6. Collision Avoidance

One of the biggest challenges in mobile robots is the problem of avoiding collisions with these obstacles. Static obstacles can be registered to map in the cloud system with their location and size. Active obstacles are defined/undefined moving objects for the cloud system. An active obstacle can be an employee, a vehicle or another AGV. Since the positions of static obstacles in the environment are known by the cloud system, a collision-free path can be created before the task execution. However, moving obstacles are unexpectedly included in the path during task execution. If this moving obstacle is an AGV, then the collision can be prevented automatically. Thus, when more than one AGV faces at the same time, the collision avoidance module is activated. When they enter a secure circular area with a radius r, which can be defined dynamically, a new local path is created for AGVs. If the moving obstacle is not an AGV, it waits for the obstacle to disappear within a default time. When the obstacle does not disappear after this period, it sends an alarm message to the cloud system via WebSockets. In Figure 11, an example of the collision avoidance system is given.

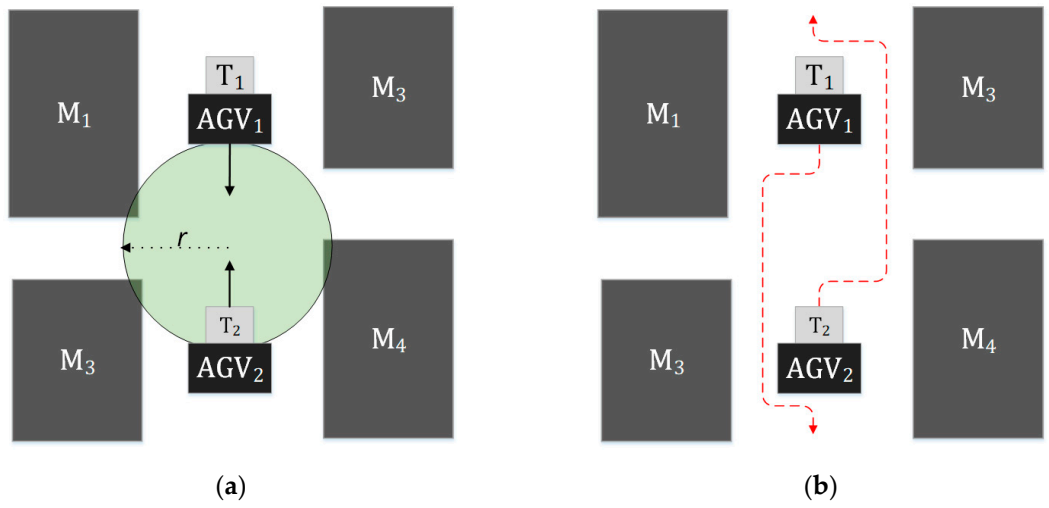


Figure 11. Collision avoidance for Automated Guided Vehicle $(AGV)_1$ and AGV_2 moving towards each other in the same direction by the cloud system (**a**) Detecting AGVs entering the same radius, (**b**) Creating a local path and re-entering the old path when it leaves the circle.

5. Simulation Results and Analysis

The simulation environment was carried out using C# language. By adding mapping features to the simulation software, it was provided to add the finishing machines, the waiting stations, the charging stations, starting and delivery ports to this map. These added static assets were uploaded to the cloud system using Restful APIs with their polygonal dimensions and their positions on the map. More than one map can be added to the simulation environment. A map environment similar to the map in the case study was created and experimental results on this map were observed. When simulated AGVs were added to the system and on the move, their positions were instantly transmitted to the cloud system using WebSockets. The dynamic parameters required for the initialization of the simulation were defined as follows:

- It was assumed that all AGVs have the same model and features. Batch trolleys were also considered to have the same features.
- The δ parameter, which is the amount of energy an AGV spends in 1 m, was set to 1 joule.
- The μ parameter, which determines the effect of the weight carried by the AGV over 1 m on energy consumption, was set to 0.001.
- The W_{max} used to calculate the critical battery level was set to 1000 kg.
- The energy spent when AGVs turns were set to $c * 0.01$. c is the amount of rotation angle of AGV.

- The connection of an AGV to batch trolley was set to spend 2 joules of energy.
- The wrapping time of the fabric to an empty batch trolley at the starting ports was set to 60 s.
- In each machine, 1 m of fabric was adjusted to be processed in 1 s.
- The weight of 1-m fabric was assumed to be 1 kg.
- It was assumed that an AGV battery has 250 joules while it is full.
- The radius of the safety area used for collision avoidance was set at 5 m.
- The φ value used in the classification of the assignment matrix was set to 8.

An example study for the simulation test is shown in Figure 12. In this example, a map with 9 finishing machines, 20 AGVs, and 30 batch trolleys was simulated and 10 fabrics were placed to the starting ports. AGVs were randomly located at the charging stations. Empty batch trolleys were also located randomly in the waiting stations.

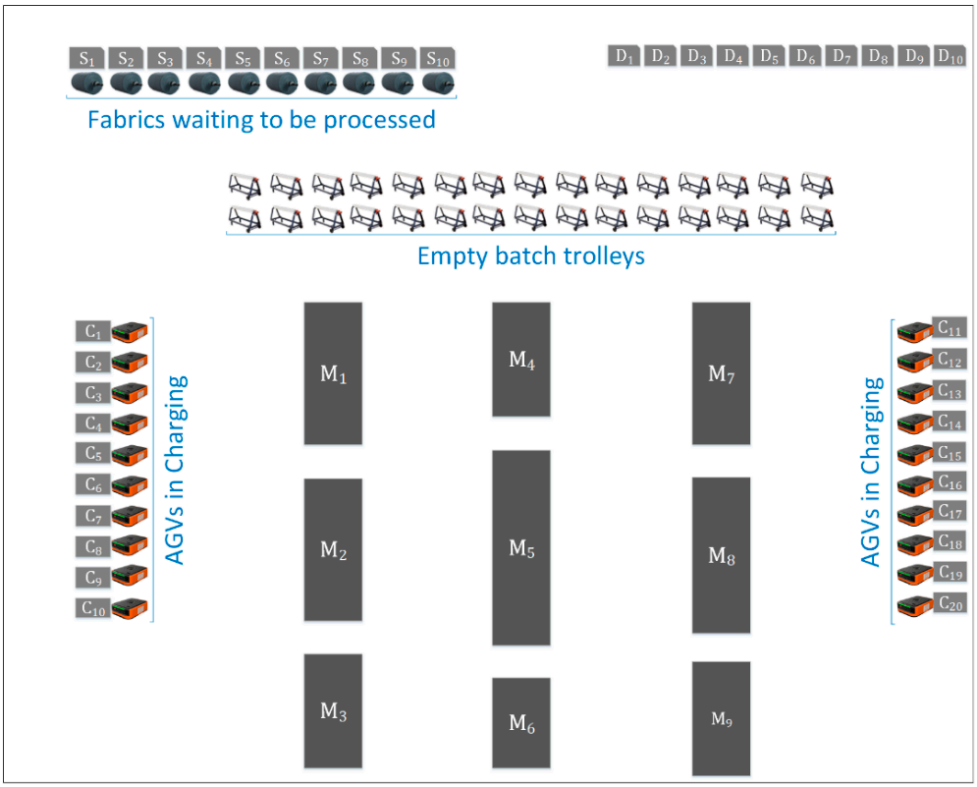


Figure 12. An example simulation map with 10 fabrics, 9 finishing machines, 20 AGVs, and 30 batch trolleys.

Machines for processing each fabric in the example given in Figure 12 are defined as tasks, sequentially as in Table 2. For example, the K_1 fabric will depart from the S_1 starting port, will be processed on M_1, M_4, M_5, and M_8 machines sequentially and then will transmit to the D_1 delivery port.

Table 2. Fabrics and their tasks.

Fabric	Sequential Tasks
K_1	$S_1 \rightarrow M_1 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow D_1$
K_2	$S_2 \rightarrow M_2 \rightarrow M_3 \rightarrow M_7 \rightarrow M_6 \rightarrow M_9 \rightarrow D_2$
K_3	$S_3 \rightarrow M_3 \rightarrow M_6 \rightarrow M_4 \rightarrow M_5 \rightarrow M_8 \rightarrow D_3$
K_4	$S_4 \rightarrow M_1 \rightarrow M_7 \rightarrow M_8 \rightarrow D_4$
K_5	$S_5 \rightarrow M_3 \rightarrow M_2 \rightarrow M_7 \rightarrow M_9 \rightarrow M_6 \rightarrow D_5$
K_6	$S_6 \rightarrow M_4 \rightarrow M_5 \rightarrow M_2 \rightarrow M_3 \rightarrow M_7 \rightarrow D_6$
K_7	$S_7 \rightarrow M_2 \rightarrow M_7 \rightarrow M_3 \rightarrow M_5 \rightarrow D_7$
K_8	$S_8 \rightarrow M_4 \rightarrow M_1 \rightarrow M_3 \rightarrow M_6 \rightarrow M_9 \rightarrow M_8 \rightarrow D_8$
K_9	$S_9 \rightarrow M_5 \rightarrow M_6 \rightarrow M_4 \rightarrow M_8 \rightarrow D_9$
K_{10}	$S_{10} \rightarrow M_5 \rightarrow M_6 \rightarrow M_4 \rightarrow M_8 \rightarrow D_{10}$

According to the working model, when a fabric is transmitted to the input port of a finishing machine, a batch trolley must also be transmitted to the output port of the machine simultaneously. According to this rule, one *output-task* has been created for each *input-task*. Therefore, the number of tasks determined for this fabric is increased by as many as the number of machines. If the machine to which the batch trolley is to be transmitted is busy, it is transmitted to the waiting stations as a rule. In this case, a *waiting-task* is created to transmit the batch trolley to the waiting station. This means that the number of tasks created for the fabric increases again. The results of the sample study above can be seen in Table 3.

Table 3. Results of the simulation example.

Fabric	Length (m)	Machine Count	Tasks	Waiting Tasks	Collision Avoidance	Total Path (m)	Time (s)	Energy (Joules)
K_1	200	4	10	0	2	152	242	346
K_2	250	5	13	1	5	296	368	548.4
K_3	375	5	12	0	3	228	357	623.65
K_4	300	3	9	1	0	131	196	309
K_5	280	5	14	2	2	317	423	639.8
K_6	325	5	12	0	4	222	349	541.32
K_7	400	4	14	2	1	224	343	545.55
K_8	415	6	16	2	6	368	516	881.4
K_9	340	4	12	2	1	226	341	586.85
K_{10}	950	4	15	3	4	303	372	636.9

In Table 3, the length of each fabric is indicated in the "Length" column. In the "Machine Count" column, there is information on the number of machines by which the fabric is processed. There is the total number of tasks created to transmit the fabric to the delivery point in the "Tasks" column. These task numbers refer to the sum of the starting-tasks, input-tasks, output-tasks, waiting-tasks and delivery-tasks created for the fabrics. For example, it is seen that 10 tasks were created for product K_1 and it was formed as in Figure 13. "Waiting Tasks" is the number of batch trolleys being transmitted to the waiting station when the machine to which the fabric is transmitted is busy. The "Collision Avoidance" column is the number of encounters and collision avoidance with other AGVs on the path during the total time. In the "Total Path" column, there is the total distance in meters travelled for the fabric to be processed on all machines and delivered to the delivery point. In the "Time" column, there is the total duration of all processes for the fabric. In the "Energy" column, the total energy spent is given in joules.

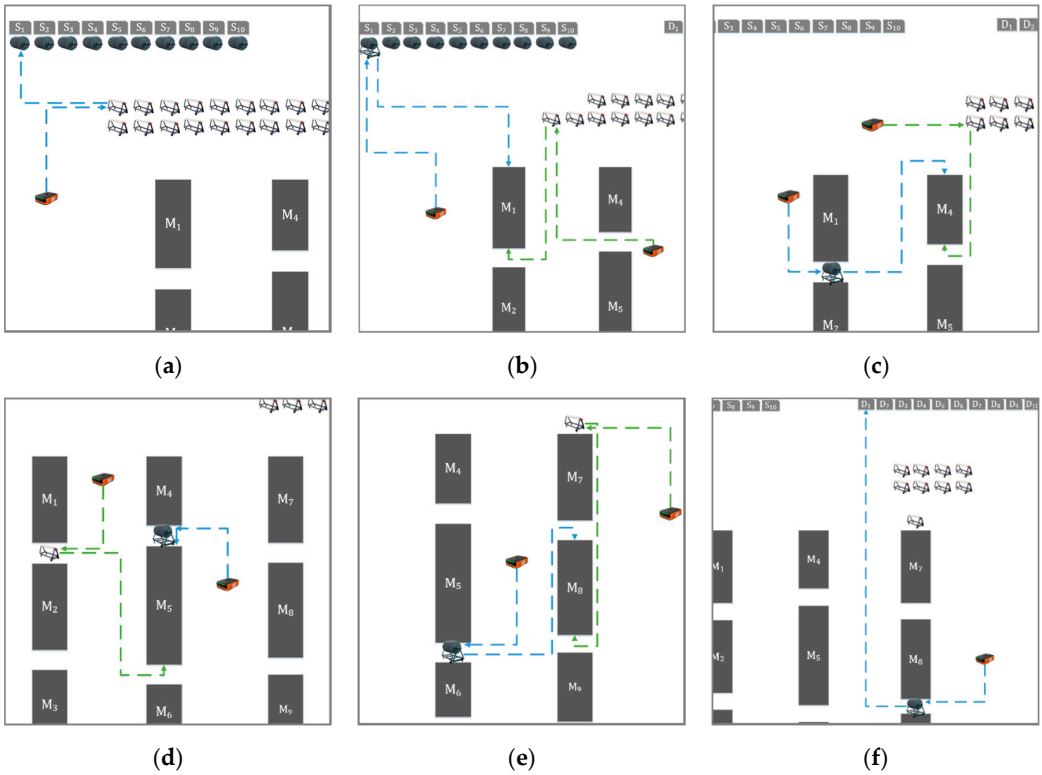


Figure 13. 10 tasks created by the cloud system to transmit the K_1 semi-manufactured fabric to the delivery point. (**a**) Starting task an empty batch trolley will be transmitted to the S_1 port, where the fabric is located. (**b**) Two separate tasks that will transmit the fabric to the input port of the M1 machine and simultaneously transmit an empty batch trolley to the output port of the machine. (**c**) Input and output tasks for M_4. (**d**) Input and output tasks for M_5. (**e**) Input and output tasks for M_8. (**f**) The delivery task for the fabric.

In the simulation example, it was seen that all the fabrics started processing simultaneously. Table 3 shows that the fabric with the longest delivery time is K_8. In this case, K_8 is the last fabric to reach the delivery point. The running time of the simulation is also equivalent to the delivery time of K_8. In Figure 14, the timelines of the delivery time of 10 fabrics, which are given in the simulation example, are shown. As can be seen from Figure 14, all tasks are completed at the end of 516 s.

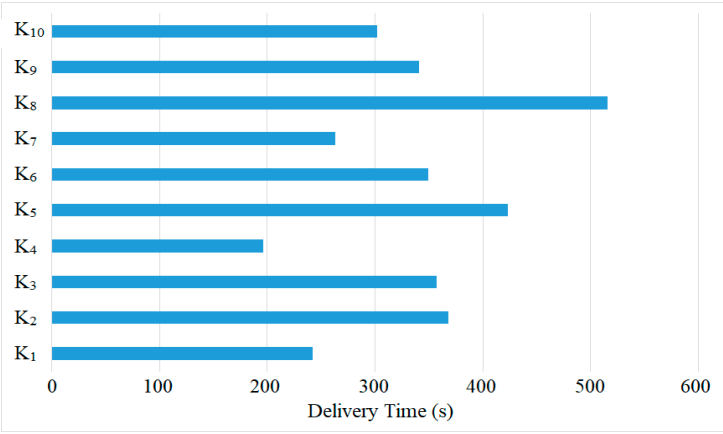


Figure 14. Timelines of the delivery of 10 fabrics in the simulation example.

In the simulation example, it was observed that the batteries belonging to 12 of the AGVs used fell below the critical level. The charging tasks were created for these AGVs and forwarded to the nearest charging station. The total energy spent to take all the fabrics to the delivery point was 5658.87 joules.

All the tests, including the above example carried out in the simulation, have shown that all tasks created in the cloud system have been successfully fulfilled and the fabrics have been transmitted to the delivery point after all finishing processes.

6. Discussions and Conclusions

In this study, a cloud-based collaboration system that autonomously controls AGVs used in industrial indoor logistics is proposed. As a case study, indoor logistics in fabric finishing enterprises have been considered. In the study, firstly, all the assets in the environment were defined and a working model was created and the role of each asset in the environment was described. It was observed that the created model also affected the working strategy of the optimum multi-task allocation and path planning algorithms.

AGVs are used in the transportation of batch trolleys, which are specified in the "Case Definition" section and provide mobility to the fabrics. In the study, a cloud system was developed that provides centralized control of the collaboration of these AGVs. The cloud system is compatible with the proposed working model for the case study. In the cloud system, communication and coordination between AGVs were ensured by using WebSocket and Restful technologies. The developed cloud system has a collaboration module that moves AGVs within the lowest-cost principle and can complete tasks successfully. This lowest-cost principle is based on a multi-fitness function that minimizes the total path and time. The collaboration module consists of task definition, battery management-optimization, choosing the most suitable batch trolley, optimum multi-task allocation and collision avoidance stages.

At the task definition stage all operations to be performed on the fabric are defined. These operations are the tasks that take the fabrics from their location and transmit them to the machines which will be processed sequentially, and finally, transmit them to the delivery port.

In the battery management and optimization stage, the critical battery level is calculated and tracked for each AGV. When the occupancy of the batteries drops below this level, the task assigned to the AGV is cancelled and assigned to another one. The AGV which is running out of battery is navigated to the nearest charging station.

At the stage of choosing the most suitable batch trolley, an algorithm that finds the closest batch trolley, taking into account its occupancy, was developed. In this algorithm, the D* Lite was used to calculate the shortest path between fabrics and batch trolleys, avoiding obstacles.

In the optimum multi-task allocation stage, an algorithm with a multi-fitness function was developed based on the minimum path to reduce the total energy cost and time to increase the number of daily production. The algorithm firstly creates a combination matrix between AGVs and positions of tasks and calculates the minimum paths between them. After calculating the minimum paths, a permutational assignment matrix was created, covering one-to-one assignment possibilities for each AGV and task. By calculating the total paths for each row in the matrix, the lowest cost row was selected. Since the assignment matrix is permutational, it has been observed that the computational complexity increased as the AGVs and tasks increased. To solve this problem, combinations classified by k-NN, according to their closest neighborhood were divided into φ number parts. Permutation calculation was calculated separately with these classified data and combined into a final matrix. When choosing the most appropriate row in the final assignment matrix, not only path lengths were taken into account, but also the level of energy to fulfil the task assigned to AGVs. If there was an AGV that did not have enough energy for the task it was assigned in any row, and this row was removed from the assignment matrix. Thus, the allocation of tasks was carried out by selecting the lowest total path among the remaining rows in the assignment matrix.

The path plan was created by avoiding the static obstacles known by the cloud system using D* Lite algorithm. However, moving obstacles are not included in the formation of the path plan.

A mobile obstacle can be a human, any object or an AGV. In the collision avoidance stage, a safe area was created to address this problem and a local and collision-free path plan was created when AGVs encountered this safe area.

Simulation software was developed to test the cloud system. It was observed that the cloud system has successfully organized and completed all the tasks created by simulation. High efficiency was achieved in total path, energy and delivery times. The developed system presents solutions for many industrial indoor logistics systems with complex structures. The modular features in the system are quite promising. These modular features were compared with prominent related works and the results of the comparison are given in Table 1.

The theoretical applicability of the system has been proven by simulation tests. We think that the study will inspire other studies in similar fields with similar processes. In addition, we are currently working on a highly accurate positioning and navigation system to create a practically applicable system. Laboratory experiments of the positioning and navigation system will be conducted in integration with the proposed cloud system and will be published in the future.

Author Contributions: Conceptualization, F.O. and E.D.; methodology, F.O.; software, F.O.; validation, E.D. and A.F.K.; formal analysis, F.O.; investigation, F.O.; resources, F.O., E.D. and A.F.K.; data curation, F.O.; writing—original draft preparation, F.O.; writing—review and editing, F.O., E.D. and A.F.K.; visualization, F.O.; supervision, A.F.K.; project administration, A.F.K.; funding acquisition, F.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This research was supported by the university-industry cooperation project carried out by Çalık Denim R&D Center and Inonu University. Çalık Denim factory finishing enterprise operating in Malatya 2nd Organized Industrial Zone was used to modelling the working environment. We thank Çalık Denim R&D Center, which greatly assists the research and provides expertise.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Llopis-Albert, C.; Rubio, F.; Valero, F. Fuzzy-set qualitative comparative analysis applied to the design of a network flow of automated guided vehicles for improving business productivity. *J. Bus. Res.* **2019**, *101*, 737–742. [CrossRef]
2. Okumus, F.; Kocamaz, A.F. Cloud based indoor navigation for ros-enabled automated guided vehicles. In Proceedings of the 2019 International Conference on Artificial Intelligence and Data Processing Symposium (IDAP 2019), Malatya, Turkey, 21–22 September 2019; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2019. [CrossRef]
3. Okumuş, F.; Fatih, A. Exploring the Feasibility of a Multifunctional Software Platform for Cloud Robotics. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018. [CrossRef]
4. Mellado, M.; Vendrell, E.; Crespo, A.; López, P.; Garbajosa, J.; Lomba, C.; Schilling, K.; Stützle, H.; Mayerhofer, R. Application of a real time expert system platform for flexible autonomous transport in industrial production. *Comput. Ind.* **1999**, *38*, 187–200. [CrossRef]
5. Khamis, A.; Hussein, A.; Elmogy, A. Multi-robot task allocation: A review of the state-of-the-art. *Stud. Comput. Intell.* **2015**, *604*, 31–51. [CrossRef]
6. Korsah, G.A.; Stentz, A.; Dias, M.B. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robot. Res.* **2013**, *32*, 1495–1512. [CrossRef]
7. Tsarouchi, P.; Michalos, G.; Makris, S.; Athanasatos, T.; Dimoulas, K.; Chryssolouris, G. On a human–robot workplace design and task allocation system. *Int. J. Comput. Integr. Manuf.* **2017**, *30*, 1272–1279. [CrossRef]
8. Lee, D.H. Resource-based task allocation for multi-robot systems. *Robot. Auton. Syst.* **2018**, *103*, 151–161. [CrossRef]
9. Wei, C.; Hindriks, K.V.; Jonker, C.M. Dynamic task allocation for multi-robot search and retrieval tasks. *Appl. Intell.* **2016**, *45*, 383–401. [CrossRef]

10. Nunes, E.; Manner, M.; Mitiche, H.; Gini, M. A taxonomy for task allocation problems with temporal and ordering constraints. *Robot. Auton. Syst.* **2017**, *90*, 55–70. [CrossRef]
11. Nedjah, N.; De Mendonça, R.M.; De Macedo Mourelle, L. PSO-based distributed algorithm for dynamic task allocation in a robotic swarm. *Procedia Comput. Sci.* **2015**, *51*, 326–335. [CrossRef]
12. Zhu, Z.; Tang, B.; Yuan, J. Multirobot task allocation based on an improved particle swarm optimization approach. *Int. J. Adv. Robot. Syst.* **2017**, *14*. [CrossRef]
13. Li, X.; Ma, H.-X. Particle swarm optimization based multi-robot task allocation using wireless sensor network. In Proceedings of the 2008 IEEE International Conference on Information and Automation (ICIA 2008), Changsha, China, 20–23 June 2008. [CrossRef]
14. Chen, J.; Yang, Y.; Wu, Y. Multi-robot task allocation based on the modified particle swarm optimization algorithm. In Proceedings of the 2011 7th International Conference on Natural Computation (ICNC 2011), Shanghai, China, 26–28 July 2011; Volume 3, pp. 1744–1749. [CrossRef]
15. Choudhury, B.B.; Biswal, B.B. A PSO based multi-robot task allocation. *Int. J. Comput. Vis. Robot.* **2011**, *2*, 49–61. [CrossRef]
16. Wan, J.; Chen, B.; Wang, S.; Xia, M.; Li, D.; Liu, C. Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4548–4556. [CrossRef]
17. Mousavi, M.; Yap, H.J.; Musa, S.N.; Tahriri, F.; Md Dawal, S.Z. Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization. *PLoS ONE* **2017**, *12*, e0169817. [CrossRef]
18. Mousavi, S.; Afghah, F.; Ashdown, J.D.; Turck, K. Use of a quantum genetic algorithm for coalition formation in large-scale UAV networks. *Ad Hoc Netw.* **2019**, *87*, 26–36. [CrossRef]
19. Jose, K.; Pratihar, D.K. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robot. Auton. Syst.* **2016**, *80*, 34–42. [CrossRef]
20. Zitouni, F.; Maamri, R.; Harous, S. FA–QABC–MRTA: A solution for solving the multi-robot task allocation problem. *Intell. Serv. Robot.* **2019**, *12*, 407–418. [CrossRef]
21. Suemitsu, I.; Izui, K.; Yamada, T.; Nishiwaki, S.; Noda, A.; Nagatani, T. Simultaneous optimization of layout and task schedule for robotic cellular manufacturing systems. *Comput. Ind. Eng.* **2016**, *102*, 396–407. [CrossRef]
22. Jin, L.; Li, S.; La, H.M.; Zhang, X.; Hu, B. Dynamic task allocation in multi-robot coordination for moving target tracking: A distributed approach. *Automatica* **2019**, *100*, 75–81. [CrossRef]
23. Chopra, S.; Notarstefano, G.; Rice, M.; Egerstedt, M. A Distributed Version of the Hungarian Method for Multirobot Assignment. *IEEE Trans. Robot.* **2017**, *33*, 932–947. [CrossRef]
24. Yao, W.; Qi, N.; Liu, Y.; Xu, S.; Du, D. Homotopic Approach for Robot Allocation Optimization Coupled with Path Constraints. *IEEE Robot. Autom. Lett.* **2019**, *5*, 88–95. [CrossRef]
25. Zhang, J.; Zhou, Y.; Zhang, Y. Multi-objective Robot Path Planning based on Bare Bones Particle Swarm Optimization with Crossover Operation. In Proceedings of the 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Huangshan, China, 28–30 July 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 330–335. [CrossRef]
26. Bae, J.; Chung, W. A Heuristic for Path Planning of Multiple Heterogeneous Automated Guided Vehicles. *Int. J. Precis. Eng. Manuf.* **2018**, *19*, 1765–1771. [CrossRef]
27. Digani, V.; Hsieh, M.A.; Sabattini, L.; Secchi, C. Coordination of multiple AGVs: A quadratic optimization method. *Auton. Robot.* **2019**, *43*, 539–555. [CrossRef]
28. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28. [CrossRef]
29. Dönmez, E.; Kocamaz, A.F. Design of Mobile Robot Control Infrastructure Based on Decision Trees and Adaptive Potential Area Methods. *Iran. J. Sci. Technol. Trans. Electr. Eng.* **2020**, *44*, 431–448. [CrossRef]
30. Okumus, F.; Kocamaz, A.F. Comparing Path Planning Algorithms for Multiple Mobile Robots. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018. [CrossRef]
31. Silveira, L.; Maffei, R.Q.; Botelho, S.S.C.; Drews, P.L., Jr.; Bicho, A.d.L.; Duarte Filho, N.L. Space D*. *J. Braz. Comput. Soc.* **2012**, *18*, 363–373. [CrossRef]
32. Schillinger, P.; Bürger, M.; Dimarogonas, D.V. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *Int. J. Robot. Res.* **2018**, *37*, 818–838. [CrossRef]

33. Turner, J.; Meng, Q.; Schaefer, G.; Whitbrook, A.; Soltoggio, A. Distributed Task Rescheduling with Time Constraints for the Optimization of Total Task Allocations in a Multirobot System. *IEEE Trans. Cybern.* **2018**, *48*, 2583–2597. [CrossRef]
34. Du, L.-z.; Ke, S.; Wang, Z.; Tao, J.; Yu, L.; Li, H. Research on multi-load AGV path planning of weaving workshop based on time priority. *Math. Biosci. Eng.* **2019**, *16*, 2277–2292. [CrossRef]
35. Afrin, M.; Jin, J.; Rahman, A.; Tian, Y.C.; Kulkarni, A. Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory. *Future Gener. Comput. Syst.* **2019**, *97*, 119–130. [CrossRef]
36. Liu, H.; Liu, S.; Zheng, K. A Reinforcement Learning-Based Resource Allocation Scheme for Cloud Robotics. *IEEE Access* **2018**, *6*, 17215–17222. [CrossRef]
37. Chowdhury, M.; Maier, M. Collaborative Computing for Advanced Tactile Internet Human-to-Robot (H2R) Communications in Integrated FiWi Multirobot Infrastructures. *IEEE Internet Things J.* **2017**, *4*, 2142–2158. [CrossRef]
38. Yan, H.; Hua, Q.; Wang, Y.; Wei, W.; Imran, M. Cloud robotics in Smart Manufacturing Environments: Challenges and countermeasures. *Comput. Electr. Eng.* **2017**, *63*, 56–65. [CrossRef]
39. Wang, L.; Liu, M.; Meng, M.Q.H. A Hierarchical Auction-Based Mechanism for Real-Time Resource Allocation in Cloud Robotic Systems. *IEEE Trans. Cybern.* **2017**, *47*, 473–484. [CrossRef] [PubMed]
40. Clark, C.M. Probabilistic Road Map sampling strategies for multi-robot motion planning. *Robot. Auton. Syst.* **2005**, *53*, 244–264. [CrossRef]
41. Solovey, K.; Halperin, D. *K*-color multi-robot motion planning. In *Springer Tracts in Advanced Robotics*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 86, pp. 191–207. [CrossRef]
42. Ma, X.; Jiao, Z.; Wang, Z.; Panagou, D. Decentralized prioritized motion planning for multiple autonomous UAVs in 3D polygonal obstacle environments. In Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 7–10 June 2016; pp. 292–300. [CrossRef]
43. Draganjac, I.; Miklic, D.; Kovacic, Z.; Vasiljevic, G.; Bogdan, S. Decentralized Control of Multi-AGV Systems in Autonomous Warehousing Applications. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1433–1447. [CrossRef]
44. Dewangan, R.K.; Shukla, A.; Godfrey, W.W. Survey on prioritized multi robot path planning. In Proceedings of the 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Chennai, India, 2–4 August 2017; pp. 423–428. [CrossRef]
45. Oh, G.; Kim, Y.; Ahn, J.; Choi, H.L. PSO-based Optimal Task Allocation for Cooperative Timing Missions. *IFAC-PapersOnLine* **2016**, *49*, 314–319. [CrossRef]
46. Guerrero, J.; Valero, Ó.; Oliver, G. Toward a Possibilistic Swarm Multi-robot Task Allocation: Theoretical and Experimental Results. *Neural Process. Lett.* **2017**, *46*, 881–897. [CrossRef]
47. Trigui, S.; Cheikhrouhou, O.; Koubaa, A.; Zarrad, A.; Youssef, H. An analytical hierarchy process-based approach to solve the multi-objective multiple traveling salesman problem. *Intell. Serv. Robot.* **2018**, *11*, 355–369. [CrossRef]
48. Velagapudi, P.; Sycara, K.; Scerri, P. Decentralized prioritized planning in large multirobot teams. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010; pp. 4603–4609. [CrossRef]
49. Narayanan, V.; Phillips, M.; Likhachev, M. Anytime Safe Interval Path Planning for dynamic environments. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 4708–4715. [CrossRef]
50. Koenig, S.; Likhachev, M. D*Lite. In Proceedings of the Eighteenth National Conference on Artificial Intelligence, Edmonton, AB, Canada, 28 July–1 August 2002; American Association for Artificial Intelligence (AAAI): Menlo Park, CA, USA, 2002; pp. 476–483.