machines

Article

Multi-Robot Task Scheduling for Consensus-Based Fault-Resilient Intelligent Behavior in Smart Factories

Vivian Cremer Kalempa [1,2,*], Luis Piardi [2,3], Marcelo Limeira [2] and Andre Schneider de Oliveira [2]

1 Department of Information Systems, Universidade do Estado de Santa Catarina (UDESC), Luiz Fernando Hastreiter St., 180, São Bento do Sul 89283-081, Brazil
2 Graduate Program in Electrical and Computer Engineering, Universidade Tecnológica Federal do Paraná (UTFPR), Av. Sete de Setembro, 3165, Curitiba 80230-901, Brazil
3 Research Center in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança (IPB), Campus de Santa Apolónia, 5300-253 Bragança, Portugal
* Correspondence: vivian.kalempa@udesc.br

Abstract: In smart factories, several mobile and autonomous robots are being utilized in warehouses to reduce overhead and operating costs. In this context, this paper presents a consensus-based fault-resilient intelligent mechanism called Consensual Fault-Resilient Behavior (CFRB). The proposed approach is based on three hierarchical plans: imposition, negotiation, and consensus. Fault resilience is achieved using the collective behavior of a multi-robot system that applies ternary decisions based on these plans. The difference between this paper and our previous work is on the consensual level. As it is suitable for the analysis and design of coordinated behavior between autonomous robots, the consensus plan is restructured and enhanced. The proposed approach is tested and evaluated in a virtual warehouse based on a real environment. In addition, it is compared with other current approaches, and the results are presented, demonstrating its efficiency.

Keywords: multi-robot task allocation; smart factories; warehouse logistics; consensus

check for updates

Citation: Kalempa, V.C.; Piardi, L.; Limeira, M.; de Oliveira, A.S. Multi-Robot Task Scheduling for Consensus-Based Fault-Resilient Intelligent Behavior in Smart Factories. *Machines* **2023**, *11*, 431. https://doi.org/10.3390/machines11040431

Academic Editor: Dimitris Mourtzis

Received: 8 February 2023
Revised: 9 March 2023
Accepted: 22 March 2023
Published: 28 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Multi-robot systems (MRSs) are widely utilized to increase the efficiency of Industry 4.0 applications [1,2]. MRSs consist of a group of robots designed with the aim of performing collective behavior [3]. This cooperation between multiple robots is aimed at accomplishing multiple tasks in parallel, which would be unattainable for a single robot. In addition, this collective behavior improves the performance, capability, and robustness of task completion. This type of collaboration has applications in several areas, such as search and rescue [4], fire emergency management services in smart factories [1], scheduling in smart factories [5], disaster environments [6], industrial warehouses [7], formation controllers for unmanned aerial vehicles [8], among others.

Another advantage of working with MRSs is that there is no single point of fault in the system. Thus, if one robot fails, another robot can take over its task. Furthermore, it presents flexibility because if more tasks need to be attended to, the group size can be increased. In addition, an MRS is efficient, as different robots can perform various tasks simultaneously [3].

One of the challenges when working with MRSs is multi-robot task allocation (MRTA), that is, determining which set of robots will perform a task and when [9]. It is also necessary to draw up a plan in case faults occur in some robots, so that another can be used as a replacement under acceptable conditions. In this paper, this behavior is referred to as fault resilience, that is, the ability of a system to recover and continue operations, despite experiencing faults [10].

This paper presents an improved version of the previous work presented by [10], where an approach to fault-resilient collective ternary-hierarchical behavior was presented to manage and adapt an MRS, and obtain fault resilience. The method proposed in this paper has three levels of decision: imposition, negotiation, and consensus. Resilience is achieved by considering choices at these three hierarchical levels: global process, groups of robots, and individual robots. The novelty of the paper is represented by the way in which the robot group makes its decision at the consensual level. In the version presented by [10], a consensual decision was made by an election. In the solution presented in this paper, the decision is made based on a set of characteristics and by presenting a consensus algorithm with two fuzzy controllers. Consensus control is a trend in MRSs, and is becoming increasingly popular among researchers because of its applicability in the analysis and design of coordination behaviors among autonomous robots [11]. With this change, an improved and more mature solution is obtained by the ternary-hierarchical decision-making process.

Thus, with this method, called Consensual Fault-Resilient Behavior (CFRB), a fault is restored as a consequence of the collective behavior obtained through ternary decisions, with the distinct feature that consensual decisions consider the individual characteristics of each robot to achieve resilient behavior to fault. Consequently, autonomous MRS can be more efficiently employed in smart factories to attain greater reliability and robustness in handling faults, thereby improving production. The presented solution allows the MRS to obtain fault resilience through the solutions generated to manage its collective behavior in smart factories.

The remainder of this paper is organized as follows. Section 2 presents the related works. Section 3 presents the problem statements and assumptions. Section 4 presents the improved fault-resilient collective ternary-hierarchical behavior. Section 5 explains warehouse logistics. Section 6 presents the proposed scheduler. Section 7 discusses the experimentation and evaluation of the proposed approach. Section 8 presents a comparison of the present approach with other approaches. Finally, Section 9 presents the conclusions and future work.

2. Related Work

The various works listed as advanced references in this paper indicate the allocation of tasks to an MRS. A few works have also addressed fault resilience, the utilization of consensus at some point in decision making, and the scenario of a smart factory for experiments. The main characteristics of these works are described in this section, with an emphasis on associating them with the proposals of this paper.

An MRTA algorithm for dynamic task allocation is proposed by Das et al. (2015) [12] to address the task allocation challenge specific to an MRS in a healthcare facility such as a care home. This approach is called Consensus-Based Parallel Auction and Execution (CBPAE) and is based on the principles of auction and consensus. In CBPAE, robots resolve bid conflicts in tasks using communication between the robots and a consensus process in each of them before a task is assigned; however, robots do not exhibit fault-resilient behavior, and there is no concern about the amount of battery available in each robot.

Hönig et al. (2018) [13] proposed a task allocation with collision-free paths, based on a Conflict-Based Search (CBS) algorithm called Conflict-Based Search with Optimal Task Assignment (CBS-TA). In this case, the CBS-TA approach extends the CBS to assign the tasks. Furthermore, Hönig et al. (2018) [13] proposed two approaches called Enhanced Conflict-Based Search (ECBS) and Enhanced Conflict-Based Search with Optimal Task Assignment (ECBS-TA), which are, respectively, trajectory solution CBS-inspired multi-agent paths and an ECBS extension that includes task assignment. Finally, Hönig et al. (2018) [13] presented a solution called prioritized planning using Safe Interval Path Planning (SIPP), which is a solution that also targets a collision-free path configuration; however, the solution of Hönig et al. (2018) [13] does not cover resilience to faults and some consensus decision making as in this paper.

Afrin et al. (2019) [1] proposed a multi-robot system based on Edge Cloud to overcome the limitations of remote cloud-based systems in the exchange of delay-sensitive data. The allocation of resources for the workflow of the multi-robot system is modeled as a restricted multi-objective optimization problem, and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) is adopted to solve it. In addition, the authors redesigned the NSGA-II algorithm by defining a new chromosomal structure, a pre-classified initial population, and a mutation operator. As a use case, the authors considered allocating resources for the workflow of a multi-robot system in an emergency management service in a smart factory. Despite the sophistication of the model, there was no concern regarding the treatment of faults or display of fault-resilient behavior.

Gregory et al. (2019) [14] presented a new heuristic for task allocation that incorporates realistic state and uncertainty modeling to improve performance. In this case, resilience was achieved using heuristics that proactively consider the probability of fault.

A mathematical model of linear programming that considers the task time window, energy consumption of the robot, and execution capacity of each robot, was proposed by Xue et al. (2019) [7]. The experiments were conducted in an intelligent warehouse picking system; however, they did not present any treatment in the case of a fault of any robot.

Our previous work published by [10] presented a solution similar to the one presented in this paper; however, with a deliberate consensus through election among the members of the group that must give up a robot, each robot voted for the robot in the group farthest away from it, so that the robot did not have to assume its responsibilities. However, in the current proposal, the consensus considers more characteristics, and a robot can reach the decision that it should leave the group if that is the consensus determined by its group. In other words, the decision is no longer an individual decision, but is based on what is best for the group.

Zitouni et al. (2020) [4] presented a distributed approach to solve the MRTA problem. The field of search and rescue of survivors was adopted, where the survivors were the tasks. The approach comprised two phases: inclusion and consensus. In the inclusion phase, each robot used the ant colony system to build a package for survivors. In the consensus phase, the robots utilized an appropriate coordination mechanism to resolve conflicts in their survivor bundle; that is, one survivor was chosen by more than two robots. In this approach, there is no concern regarding possible faults.

Mayya et al. (2021) [15] proposed a resilient and centralized allocation of heterogeneous robots for tasks in difficult environmental conditions, such as climatic events or attacks by opponents. The main objective is to ensure that each task receives the necessary level of resources, measured as the aggregate capacities of the robots allocated to the task. By tracking task performance deviations under external disturbances, the approach quantifies the extent to which the robot's capabilities (for example, visual detection or mobility) are affected by environmental conditions. This allows an optimization-based framework to flexibly relocate robots to tasks, based on the most degraded resources of each task. Notwithstanding resource constraints and adverse environmental conditions, the approach relaxes resource constraints corresponding to certain tasks, thereby exhibiting acceptable performance degradation; however, this approach has the characteristic of being totally centralized.

In Choudhury et al. (2022) [16], an algorithm is presented to solve the problem of dynamic task allocation for multiple robots, considering time constraints and uncertainty in task completion, called Stochastic Conflict-Based Allocation (SCoBA). The algorithm's efficacy was tested in two scenarios: multi-arm conveyor belt pick-and-place and multi-drone delivery dispatch in a city. Throughout these tests, SCoBA consistently outperformed baseline methods in terms of scalability with regard to the number of tasks and robots. However, the algorithm does not have any mechanism for resilience to faults or achieving consensus in the event of faults.

Kim and Lee (2023) [17] propose a practical method for multi-robot task scheduling in Antarctic environments using ant colony optimization. The method was tested in simulated and real Antarctic environments and showed that the proposed method outperforms other algorithms by finding more efficient multiple paths with lower costs. The authors justify their focus on Antarctic environments due to the risks associated with operating robots in such environments. However, the proposed method does not address any potential issues related to robot faults in these environments.

Finally, Martin et al. (2023) [18] proposed a cooperative game theory framework to solve multi-robot task allocation problems. A large-scale experiment was carried out to demonstrate the solution's benefits, comparing the results with those of a genetic algorithm. The proposed approach outperformed the genetic algorithm solution in performance and computation time for various problem instances. However, problems related to faults are not addressed.

Such task allocation problems for MRSs served as an inspiration for the approach developed in this paper. In this case, the fault-resilient behavior, together with the consensus decision making of each group of robots, aim to improve the allocation of tasks for MRSs in a real environment. In addition, the utilization of a Coppeliasim (V-REP) simulation environment, ROS framework, and experimental architecture are different factors that make the system more realistic. Tables 1 and 2 list the comparison of the main aspects presented in this paper with those of the papers cited.

Table 1. Comparison with related works—Part 1.

Main Features	This Paper	Das et al. (2015) [12]	Hönig et al. (2018) [13]	Afrin et al. (2019) [1]	Gregory et al. (2019) [14]	Xue et al. (2019) [7]
Approach	Collective ternary-hierarchical behavior resilient to faults	Based on auction and consensus principles	Task allocation with collision-free paths, based on the Conflict-Based Search algorithm	Based on the Non-dominated Sorting Genetic Algorithm II	Task allocation heuristic that incorporates realistic state and uncertainty modeling to improve performance	Based on the mathematical model of linear programming
Fault Resilience	✓				✓	
Consensus	✓	✓				
Target Application	Warehouse/ARENA	Health area	Uninformed	Emergency fire management service in a smart factory	Disaster environment	Industrial plant warehouse picking system
Implementation	ROS, Rviz, Coppeliasim (V-REP)	Own Python Simulator, ROS, and Stage	Own C++ Simulator	Matlab	Own Python Simulator	LINGO11
Number of Robots	8	50	100	50	5	10
MRS Type	Homogeneous	Heterogeneous	Homogeneous	Homogeneous	Heterogeneous	Homogeneous
Considers Priority	✓			✓		✓
Considers Energy	✓	✓				
Considers Distance Traveled	✓	✓	✓	✓	✓	✓

Table 2. Comparison with related works—Part 2.

Main Features	This Paper	Kalempa et al. (2020) [10]	Zitouni et al. (2020) [4]	Mayya et al. (2021) [15]	Choudhury et al. (2022) [16]	Kim and Lee (2023) [17]	Martin et al. (2023) [18]
Approach	Collective ternary-hierarchical behavior resilient to faults	Collective ternary-hierarchical behavior resilient to faults	Use of ant colony system and Consensus-Based Bundle Algorithm	Optimization-based	Stochastic Conflict-Based Allocation (SCoBA)	Based on Ant Colony Optimization	Use of cooperative game theory framework
Fault Resilience	✓	✓		✓			
Consensus	✓	Consensus based on election	✓				
Target Application	Warehouse/ARENA	Warehouse/ARENA	Search and rescue environment	Environment with difficult environmental conditions	Pick-and-place and delivery	Antarctic environments	Industrial plant
Implementation	ROS, Rviz, Coppeliasim (V-REP)	ROS, Rviz, Coppeliasim (V-REP)	Jade Framework and Java language	Coppeliasim (V-REP)	Julia programming language	Own Python Simulator	Matlab
Number of Robots	8	8	20	10	30	40	15
MRS Type	Homogeneous	Homogeneous	Homogeneous	Heterogeneous	Homogeneous	Homogeneous	Homogeneous
Considers Energy	✓	✓		✓			✓
Considers Priority	✓	✓	✓	✓			✓
Considers Distance Traveled	✓	✓	✓	✓		✓	✓

3. Problem Statement and Assumptions

In agent networks (or dynamic systems), according to Olfati-Saber et al. (2007) [19], a consensus means reaching an agreement that depends on the state of all agents. Thus, a consensus protocol is a communication rule that specifies the exchange of information within a network, between an agent and all its close neighbors [20], as well as the internal processing of information obtained by each agent [21]. Thus, to converge to a single decided value, a consensus protocol makes the processes change their proposed values [22]. In this sense, a parallel can be drawn between agents and autonomous robots in an MRS, where the definition of consensus can also be applied in an MRS.

A consensus algorithm ensures that a single value among the proposed values is chosen. The security requirements for consensus are as follows [23]:

- Solely one value that has been proposed can be chosen;
- Solely a single value is chosen;
- A process never determines that a value was chosen unless it really was.

Thus, only the proposed values can be chosen as the sole value, and once this value is decided, all the agents know it.

Assuming there are a group of N agents in a simple consensus algorithm, each agent sends its value to all members of the group and waits until it receives $N - 1$ messages. The definition of the value to be chosen is obtained by a function on the set of all received values, for example, a function of maximum or minimum, average, or majority; however, if the agents fail, the consensus is compromised. In this sense, a few fault-tolerant consensus algorithms have been developed, such as the Paxos [23] and Raft [24] algorithms; the Raft algorithm is easier to understand than Paxos, and provides a better basis for building practical systems [24].

The Raft algorithm decomposes the consensus problem into three relatively independent sub-problems [24]:

- Leader election: a new leader must be chosen when an existing leader fails;
- Log replication: the leader must accept log entries from clients and replicate them to other servers;
- Safety: if any server has applied a particular log entry to its state machine, then no other server can utilize a different command to a similar log index.

A cluster in the Raft algorithm contains several servers. Each server is in one of three states: leader, follower, or candidate, as illustrated in Figure 1. In regular operations, there is precisely one leader, and all other servers are followers. The followers are passive; they do not issue requests independently, but respond to requests from leaders and candidates. The leader handles all the customer requests; however, if a customer contacts a follower, the follower redirects them to the leader. The third state, the candidate, is used to elect a new leader. A follower becomes a candidate when it does not receive any communication. A candidate who receives votes from the majority of the cluster becomes the new leader. Leaders typically operate until they fail [24].

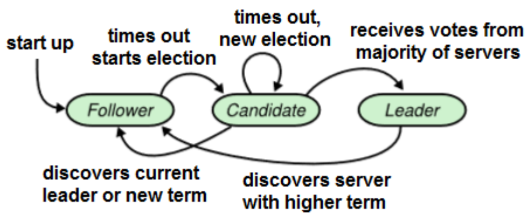


Figure 1. Server states in Raft algorithm [24].

The Raft algorithm divides time into terms of arbitrary lengths, as illustrated in Figure 2. The terms are numbered using consecutive integers. For each term, there is an election with one or more candidates. The candidate who wins the election will be the leader of the term for which it was elected. The term will be leaderless in the case of a tie.

In this case, a new election is initiated. The Raft algorithm guarantees that there is at most one leader in a given term [24].

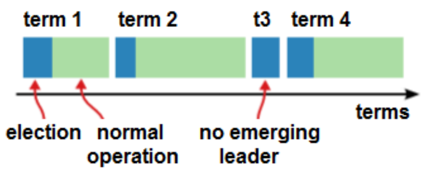


Figure 2. In the Raft algorithm, time is divided into terms, and each term starts with an election [24].

Upon being elected, the leader can serve the customers. All system changes are the responsibility of the leader. Each client request must be replicated across all servers. Therefore, to confirm an entry from a customer, the leader first replicates it to the follower nodes, and waits until most nodes have written the entry. Upon receiving feedback from most followers, the entry is confirmed to the leader, and the leader notifies the followers that the entry has been confirmed. This process is called log replication [24].

The Raft algorithm inspires the consensus approach proposed in this paper. It aims to obtain collective behavior to achieve fault resilience in MRSs, leveraging the algorithm's main advantage of ensuring information consistency and availability, even in individual component faults. The proposal is not intended to discuss the detection and classification of faults, but to provide a solution that does not interrupt the operations of an industry in production.

4. Consensual Fault-Resilient Behaviour (CFRB)

The proposed ternary-hierarchical behavior is an expansion of the work presented by [10]. The approach, called Consensual Fault-Resilient Behavior (CFRB), is a method for an MRS to present a fault-resilient behavior via a community decision. The decision is obtained by processing at three hierarchical levels: imposition, negotiation, and consensus. The imposition level is performed by the scheduler; the negotiation level by the groups of processes that execute the tasks at a given moment of time; and the consensus by the robots of each process group. The processes and tasks are detailed in Section 6, where a process is a set of tasks, to be executed by a set of robots. An illustration of the hierarchical layers is illustrated in Figure 3. The difference between the work presented in [10] and that presented in this paper is in the consensus layer. The main reason for improving the consensus layer in the approach proposed in this paper is that it allows the system to achieve collective fault-resilient behavior, allowing robots to reach a common agreement on their actions.

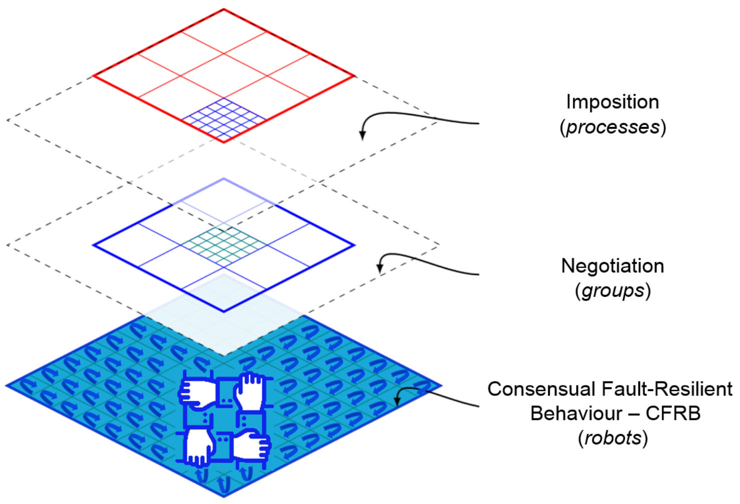


Figure 3. Layers of the ternary-hierarchical approach.

Fault recovery occurs because of the decisions made in these three planes, as illustrated in Figure 3. That is, the imposition plan sets the initial conditions for decision making in the negotiation plan. The negotiation plan provides the conditions for the decision making in consensus by the group of robots for each process that will be required to release a robot to deal with a fault.

4.1. Imposition

The imposition plan is the first level of the ternary-hierarchical decision-making process. At this level, the scheduler indicates which process/processes failed one or more robots and initiated the recovery procedure. The scheduler does not indicate which process will have to release robots; it solely indicates the priority of the process that failed and the number of robots, as well as determines which process will be the decision-making manager. The manager will always be the process with the highest priority, or in the case of a tie, the oldest.

This information is sufficient for the lower levels to follow the hierarchical fault recovery plan, without affecting other processes.

4.2. Negotiation

The negotiation plan is the second level of ternary-hierarchical decision making. At this level, all processes are analyzed to verify which one would best serve the process that suffered any fault. For decision making at this level, each process must inform the following:

- Redundancy degree: that is, the number of surplus robots that the process has, in addition to the number required to service the process, according to its priority. If more than one process has extra robots, a process with lower priority is preferred;
- Laxity: the laxity of each process is calculated according to the Least Laxity First (LLF) algorithm [25]. Thus, if two processes have equal priorities and a similar number of robots, the process that presents greater laxity will be considered the lowest priority.

After receiving the information from each of the processes, the plan manager analyzes them. First, the manager checks if any process has extra robots. If solely one process has an extra robot, this process will be chosen to release the robot. If more than one process has extra robots, then the one with the lower priority will be chosen. In the event of a tie, the process with the greatest laxity will be chosen to release the robot(s).

If there is no possibility of replacement, owing to the fact that the process is the one with the lowest priority or the one with the highest laxity, and the others do not have extra robots to release, this process will suffer preemption. Thus, there is no reversal of priorities.

However, the process that releases robots does not lose its priority, even if it does not have the required minimum number of robots. This is the advantage that the process gets from having assigned robots to a more important process. This process retrieves the robot as soon as the process with the highest priority completes processing.

4.3. Consensus

The third and final level of the ternary-hierarchical approach is the *consensus plan*. After the negotiation plan has decided which process will release robots, the group decides which robot will be released to fill the fault in another process. The consensus plan presented in this paper differs from that presented by [10], as in the plan presented in [10], the decision in the consensual plan was taken by voting. However, in the improved approach presented in this paper, the decision is made based on a set of characteristics by all members of the group.

In the consensual plan, a member of the process robot team that needs to hand over robots is chosen as the robot manager. The role of the robot manager is to determine among all members which ones are eligible to participate in the consensus, that is, the robots that do not have faults. This list is then transmitted via messages to the other robots on the team so that consensus is initiated, and a response from the majority of the followers is awaited.

Upon receiving feedback from the majority of team members, the robot manager notifies the other team members regarding which members are chosen. This stage of the consensual plan was inspired by the Raft algorithm presented by Ongaro and Ousterhout (2014) [24].

The following characteristics are raised by each robot in the process group that will assign a member, in relation to the other members of the group that have no fault: distance from the robot to complete the current task; distance from the robot to complete the task of the failed robot; the robot's Wi-Fi signal; the time left for the next robot maintenance; and the robot battery percentage.

Two fuzzy controllers were developed using the Mamdani model to determine the choice of the best robot to be voted on by each group member. Suppose the priority of the process that requires robots is higher than the current process of the group of robots that reaches a consensus in determining which robot will leave the group. In this case, the consensual approach makes use of a fuzzy controller for higher priorities. Otherwise, the approach uses a fuzzy controller for the lower priorities, as illustrated in Figure 4. Fuzzy controllers are a practical alternative for several challenging control applications because they provide a convenient approach to building controllers using heuristic information [26].

The inputs and outputs of the two fuzzy controllers are similar, as illustrated in Figure 5. The first input, called the current task distance, analyzes the distance information in the ARENA [27] position graph from the task being executed. ARENA is a small-scale warehouse utilized in this work to conduct the experiments. The warehouse state machine, illustrated in Figure 6, displays transitions between states representing the direction in which the robots can travel, avoiding collisions between them. Each state can solely be occupied by one robot. This entry has the following linguistic variables: Near (NC), Medium (MC), and Far (FC), as illustrated in Figure 5a. The second input, called the fault task distance, is utilized to analyze the distance of the robot from the task of the failed robot. Its linguistic variables are as follows: Near (NF), Medium (MF), and Far (FF), as illustrated in Figure 5b. The third input is the robot's Wi-Fi signal, which has the linguistic variables: Low (LW), Medium (MW), and High (HW). The input for the Wi-Fi signal is illustrated in Figure 5c. ARENA has two wireless routers placed in strategic locations in the warehouse to have ample coverage, as illustrated in Figure 7. The closer to the router, the better the signal; hence, the signal will be considered high if it is between −40 dBm and 0 dBm; medium if it is between −62 dBm and −38 dBm, and low if it is between −100 dBm and −60 dBm.

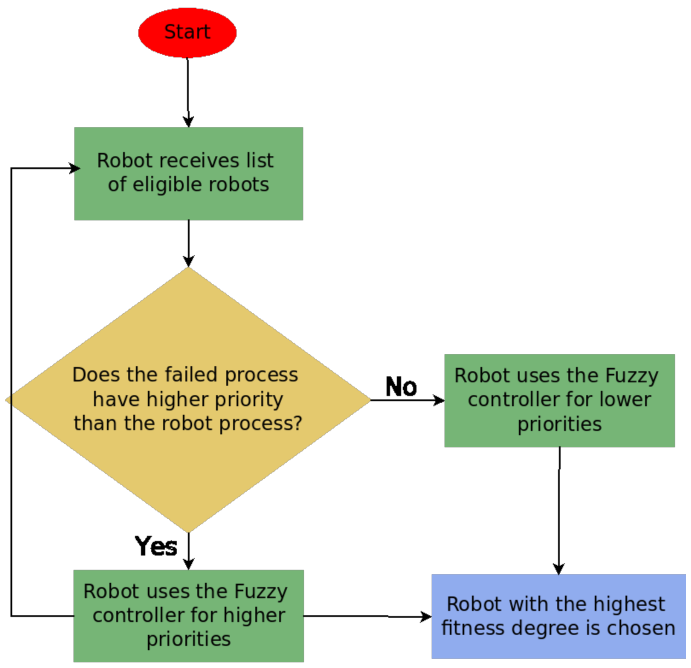
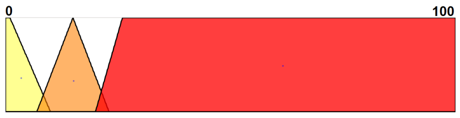
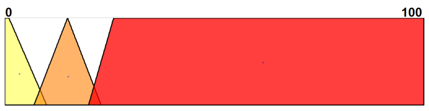

Start
Robot receives list of eligible robots
Does the failed process have higher priority than the robot process?
No
Robot uses the Fuzzy controller for lower priorities
Yes
Robot uses the Fuzzy controller for higher priorities
Robot with the highest fitness degree is chosen

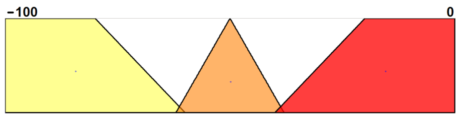
Figure 4. Flowchart referring to Fuzzy controllers for higher and lower priorities.



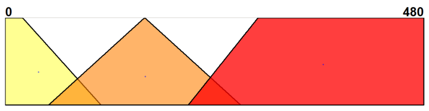
(**a**) Fuzzy input for current task distance.



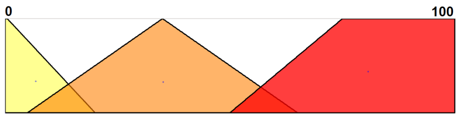
(**b**) Fuzzy input for fault task distance.



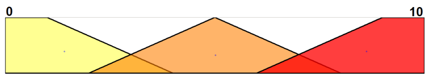
(**c**) Fuzzy input for the robot's Wi-Fi signal.



(**d**) Fuzzy input for robot maintenance time.



(**e**) Fuzzy input for robot battery percentage.



(**f**) Fuzzy output to the robot's fitness degree.

Figure 5. Inputs and outputs for the Fuzzy consensus controller.

The fourth entry of the two fuzzy controllers is for the maintenance time information that the robot has, in minutes. The linguistic variables for the maintenance time are as follows: Low (LM), Medium (MM), and High (HM), as illustrated in Figure 5d. The last input for the fuzzy controller is the battery percentage information of the robot; the corresponding linguistic variables are Low (LB), Medium (MB), and High (HB). The input for the robot battery percentage information is illustrated in Figure 5e. Finally, the fuzzy output is the robot's degree of ability to perform substitution in another process group. The fuzzy output is illustrated in Figure 5f, and has Low (LD), Medium (MD), and High (HD) as its linguistic variables.

The fuzzy controller for higher and lower priorities has a set of 243 rules each. Additionally, the technique used for defuzzification was centroid. What differs between them is the way in which the rules are drawn up. For both, if the robot battery, maintenance time, or Wi-Fi signal receives the Low linguistic variable, the result of the rule will be Low.

For the fuzzy controller for higher priorities, if the distance of the failed robot task is Near, the result of the rule will be High. Specifically, it will give a better fitness degree to the robot that is faster, to fulfill the task with the highest priority. If the distance of the failed robot task is Medium, the result will be Medium. Finally, if the distance is Far, the result will be Low, as this robot will be too far from the failed robot's task.

For the fuzzy controller for lower priorities, the distance from the robot's current task will be prioritized; that is, the farther it is from the current task, the higher its grade will be. Therefore, if the distance of the current task is Near, the output of the rule will be Low. If the distance is Medium, the output will be Medium. Finally, if the distance is Far, the output will be High.

Thus, at the end of the processing, each robot must reach the indication of a robot to leave the group. As the group's decision is consensual, a robot may indicate to leave the group itself, if this is best for the process that is handing over the robot and for that which will receive it. In the event of a tie, if the task that requires attention is of lower priority, the robot with the greatest distance to complete the current task will be released. If the task that needs to be serviced is of higher priority, the robot with the shortest distance to the task to be serviced will be assigned.

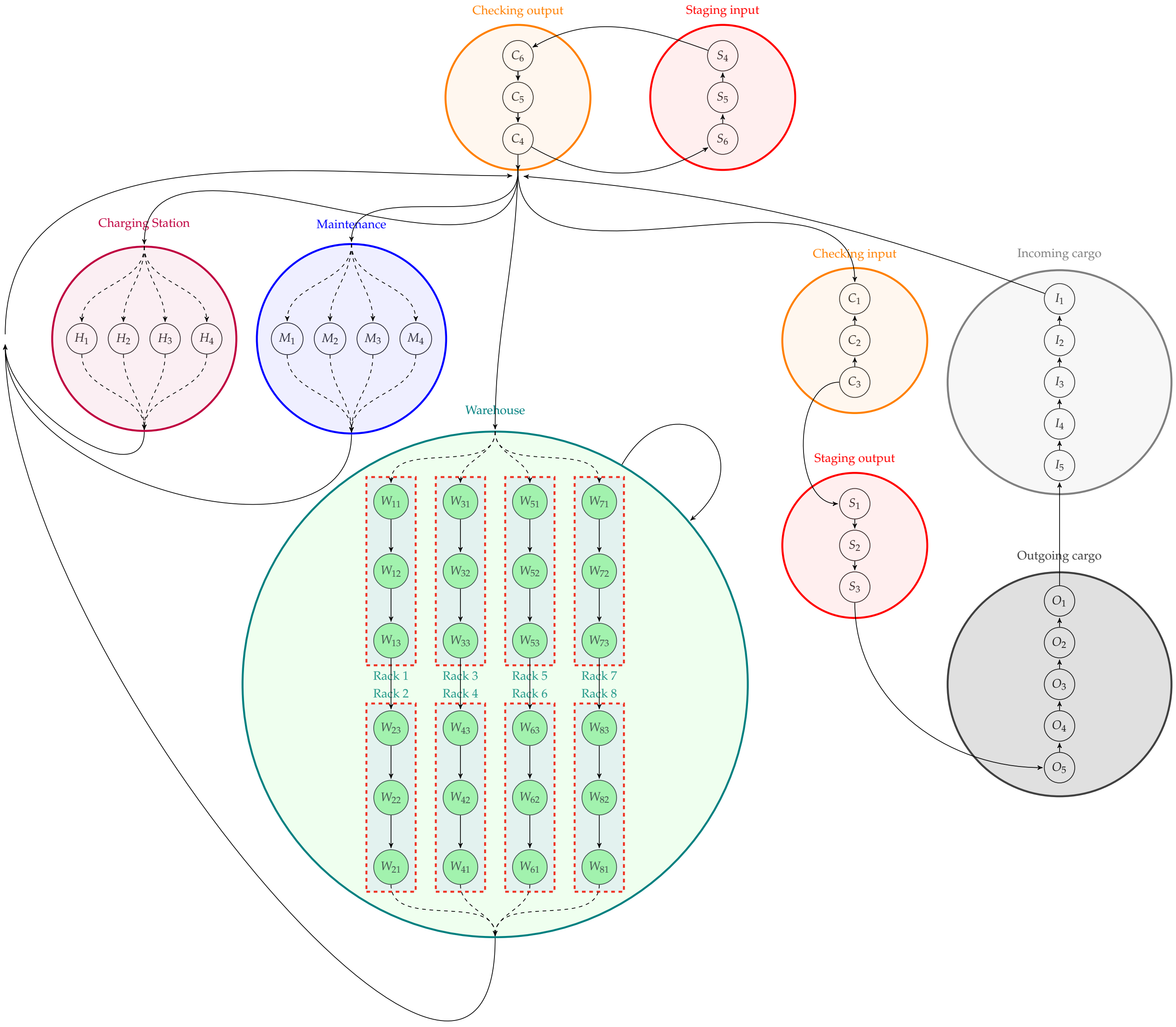


Figure 6. State machine of warehouse logistic process [27].

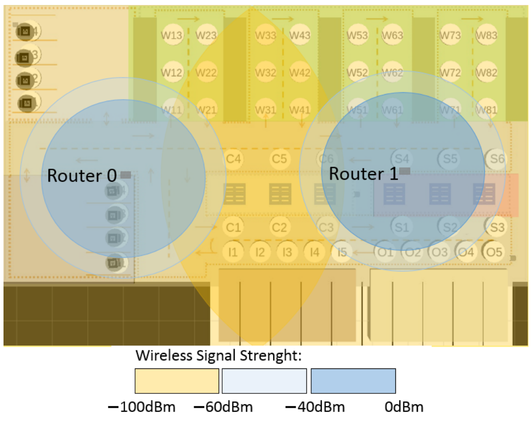


Figure 7. Wi-Fi signal coverage from two routers.

The sequence diagram illustrated in Figure 8 presents in detail the procedure performed by the CFRB approach. The process starts with the scheduler determining who the process manager will be, and the priority and number of robots in the failed process. With this information, the process manager requests information regarding the priority, number of extra robots, and laxity of each process. Upon receiving the process data, the manager decides which process will assign the robots. This decision is transferred to processes which then indicate the choice of the robot manager. The robot manager is responsible for preparing the list of eligible robots, and transmitting them to other members of the group. Each robot analyzes the list of eligible robots and makes its choice according to each robot's feature set. The decision of each robot is sent to the robot manager, which, upon receiving the response from the majority of agents, makes its decision and communicates it to the other members of the group. The managing robot also communicates to its process that the decision has been taken, and this information is confirmed to the scheduler. At this point, the fault recovery process ends.

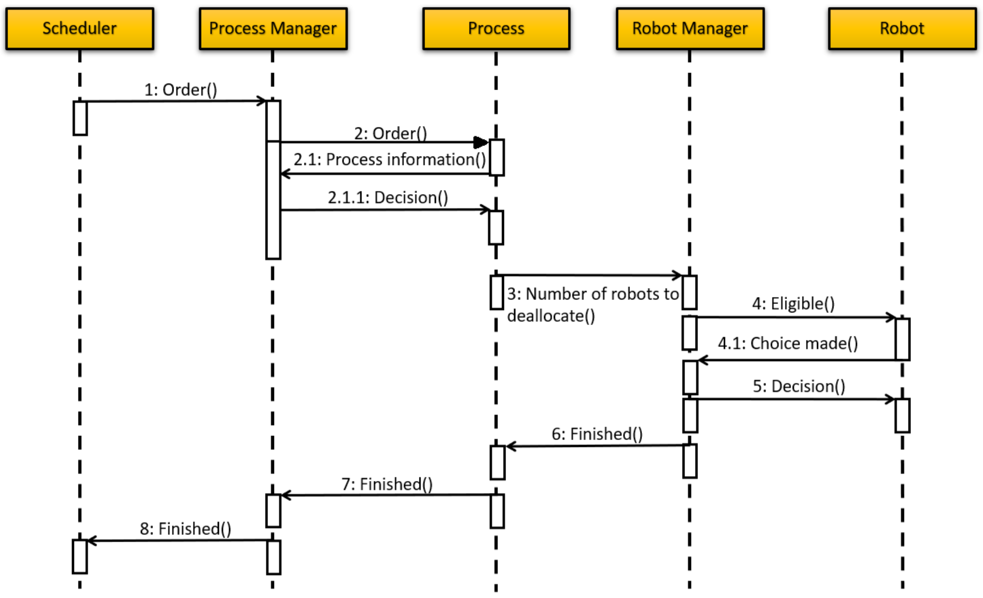


Figure 8. Sequence diagram of the CFRB approach.

5. Warehouse Logistics

Logistics refer to the transport and storage of materials, parts, and products in a supply chain [28]. Warehouse logistics collect, store, distribute, deliver, and manage the inventory of items. Logistics comprise several specific processes and tasks such as receiving, checking, sorting, storage, picking, staging, and delivering tasks.

The receiving task is responsible for managing the receipt of goods. According to Koster et al. (2007) [29], this task includes unloading products from trucks, updating the stock register, and inspecting for any inconsistencies in quantity or quality. In addition, it can include repackaging, for example, from full pallets to boxes. Then, when a receiving event occurs, a pick-up process is started to unload goods from the trucks parked in the goods receipt area. In this case, the forklifts must handle goods from the goods receipt area and take them to the checking area to be verified and recorded in the inventory database, designated as a checking task. After being revised, the cargo must be organized and stored, which is a sorting task. Finally, the storage task deals with storing cargo in the warehouse. In this paper, these tasks are part of the incoming cargo process.

The picking task is the process of removing products from a warehouse in response to a specific customer request, and placing them in the staging sector [29]. The staging task involves the preparation or packaging of goods for delivery. Finally, cargo dispatch occurs in the outgoing cargo sector, which is a function of the delivery task. In this paper, these tasks are a part of the outgoing cargo process.

The logistics of a smart warehouse can be organized as illustrated in Figure 9, with the maintenance, warehouse, charging station, checking, staging, incoming cargo, and outgoing cargo sectors. In addition, Figure 9 identifies as a white circle each space that a single robot can occupy, and the arrows indicate the direction in which a robot can move to avoid collisions.

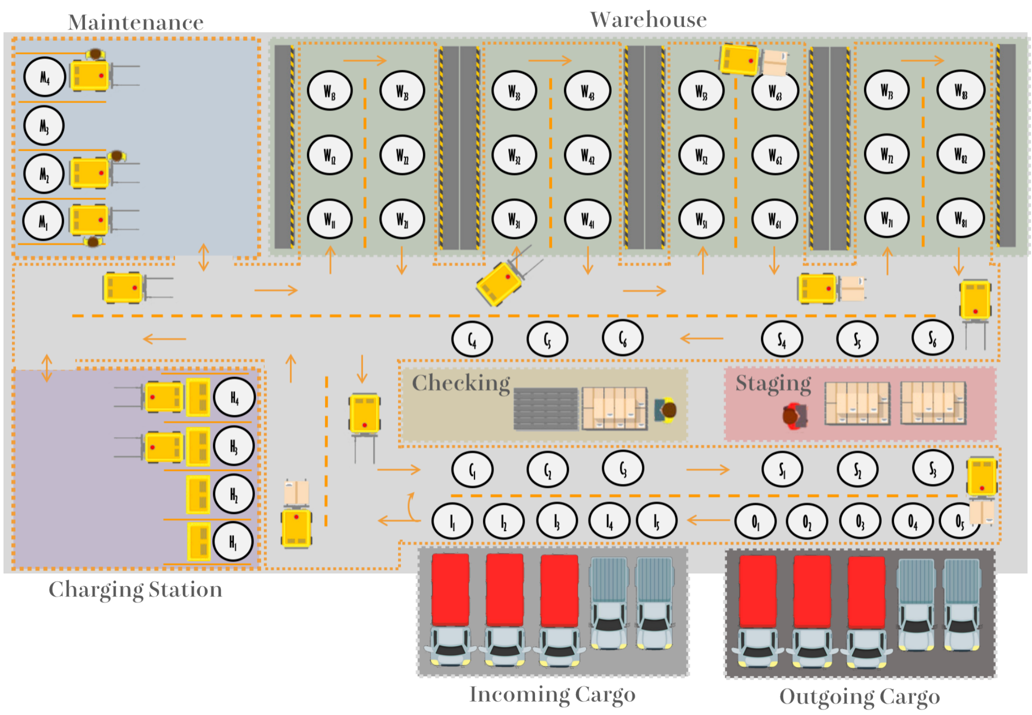


Figure 9. Warehouse logistics organization [27].

In this logistical scenario, several unpredictable situations can occur, altering its execution, such as the breakdown of a forklift or a forklift not having enough battery. In addition, it is a process that requires good planning, as several tasks with different priorities occur simultaneously.

6. Process Scheduling

The scheduler of the proposed approach works with two types of processes: incoming and outgoing cargos. The receiving, checking, sorting, and storage tasks are a part of the incoming cargo process. Picking, staging, and delivery tasks are a part of an outgoing cargo process. Each process, and consequently its tasks, has a treatment priority, which also determines the minimum limit of robots that will be a part of its group of workers. Each process to be handled by the warehouse MRS must have one of the following priorities:

Priority 1. *Minor: for less important warehouse processes. These processes can be performed when possible, and do not require a minimum number of robots.*

Priority 2. *Normal: for processes that do not have special requirements; however, they need to be executed when created. This type of process requires at least one robot.*

Priority 3. *Major: for processes with significant impacts on the warehouse's MRS, that is, a process with several boxes, for example. This type of process requires at least two robots.*

Priority 4. *Critical: for processes that need to be completed quickly, for example, processes with refrigerated products or with time limits imposed by contracts. This type of process requires a minimum of three robots.*

The operation of the scheduler, together with the proposed approach, is illustrated in the flowchart of Figure 10. The scheduler works with the limit of three processes in parallel to allow the optimal flow of autonomous robots in the warehouse, and to utilize autonomous robots in a better-distributed manner. Process priority constraints determine the allocation of the robots; however, if a process requires fewer robots than required by its priority, the remaining robots are ceded to other processes in preference to those with higher priority or older ones.

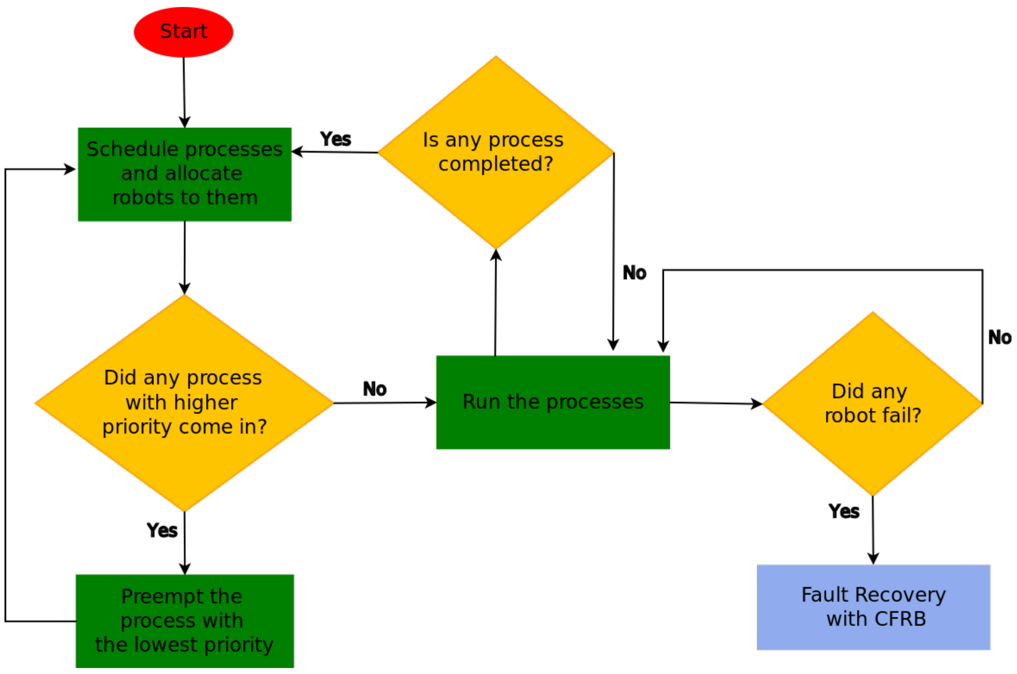

Start
Schedule processes and allocate robots to them
Yes
Is any process completed?
No
No
Did any process with higher priority come in?
No
Run the processes
Did any robot fail?
Yes
Yes
Preempt the process with the lowest priority
Fault Recovery with CFRB

Figure 10. Process scheduling.

In addition, if a process arises with a higher priority than the ones that are running, the least important among them is preempted. Preemption ensures that processes with higher priorities are serviced first. The preempted process has its status saved and returns to the process queue. When resumed, it resumes its execution from where it left off. Finally, if any robot fails, fault recovery is performed utilizing the proposed CFRB approach.

7. Experimental Evaluation of CFRB

The CFRB proposal is evaluated in a small-scale warehouse called ARENA. ARENA was presented in the work conducted by [27]. ARENA is a small-scale representation of a real Brazilian warehouse. This warehouse does not have automation, but has human operations and controls. In this work, the intention is to use ARENA with augmented reality (AR) to demonstrate the utilization of an autonomous warehouse system. Figure 11 illustrates the ARENA view, with and without AR elements.

To reproduce the movements and actions of real forklifts, mobile robots called WsBots were developed. The work presented by [30] presents the WsBot micro robot in detail, which is also illustrated in Figure 11. WsBots also allow the assessment of smart behavior in smart factories.

The evaluation of the CFRB approach was planned based on two experiments. The first experiment demonstrated a situation in which a robot from a process with a lower priority received a robot from a process with a higher priority. This happened because the process with the highest priority had two extra robots. In the second experiment, the priority of the process that lost a robot was higher than the priority of the process of the group that gave up a robot.

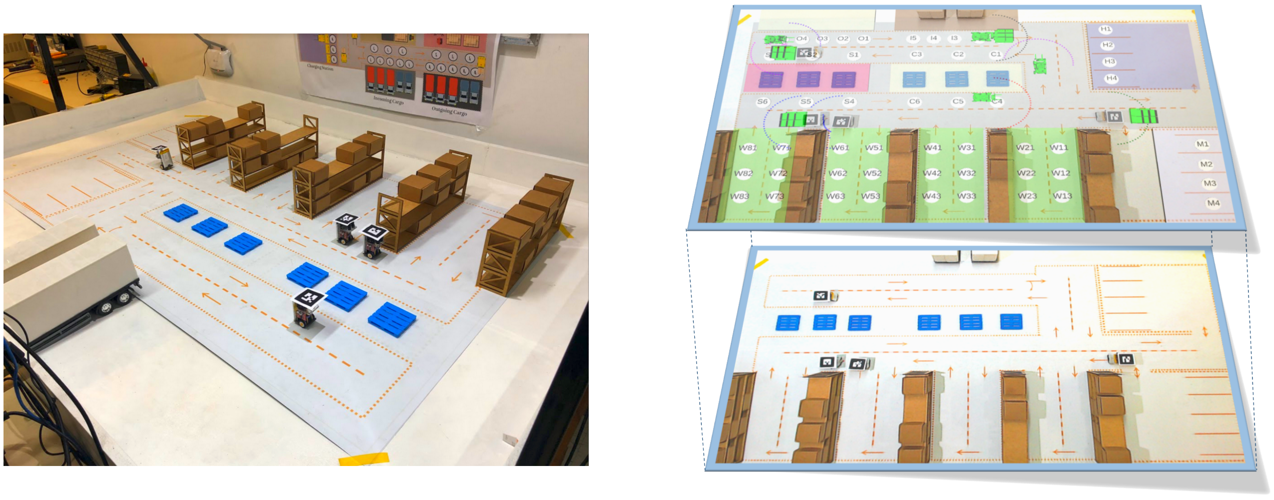


Figure 11. The ARENA with only real elements on the (**left**) and augmented reality elements on the (**right**).

7.1. Experiment 1: Consensual Decision Using Fuzzy Controller for Lower Priorities

The first experiment is an example in which a process with a higher priority gives one robot to a process with a lower priority. This happens because the process with the highest priority has two more robots than necessary for its priority. In this experiment, the fuzzy output is utilized using the fuzzy controller for lower priorities. The first experiment has three types of *incoming cargo*, as described in Table 3.

Table 3. Processes of experiment 1.

#	Type of Process	Priority	Number of Boxes	Initial State	Warehouse Aisle
Process-1	Incoming Cargo	3	4	I1	4
Process-2	Incoming Cargo	4	3	I2	2
Process-3	Incoming Cargo	2	1	I3	1

The first scene from experiment 1, illustrated in Figure 12, shows the three processes executed. The group of robots from the first process (robots 0–3) receives a yellow color and heads toward state I1 to move the boxes from the incoming cargo sector to the checking sector. The group of robots from the second process (robots 4–6) receives a dark blue color and heads toward the I2 state, as illustrated in Figure 12. Finally, robot 7 is the sole robot in process 3 with a light blue color, and heads toward state I3.

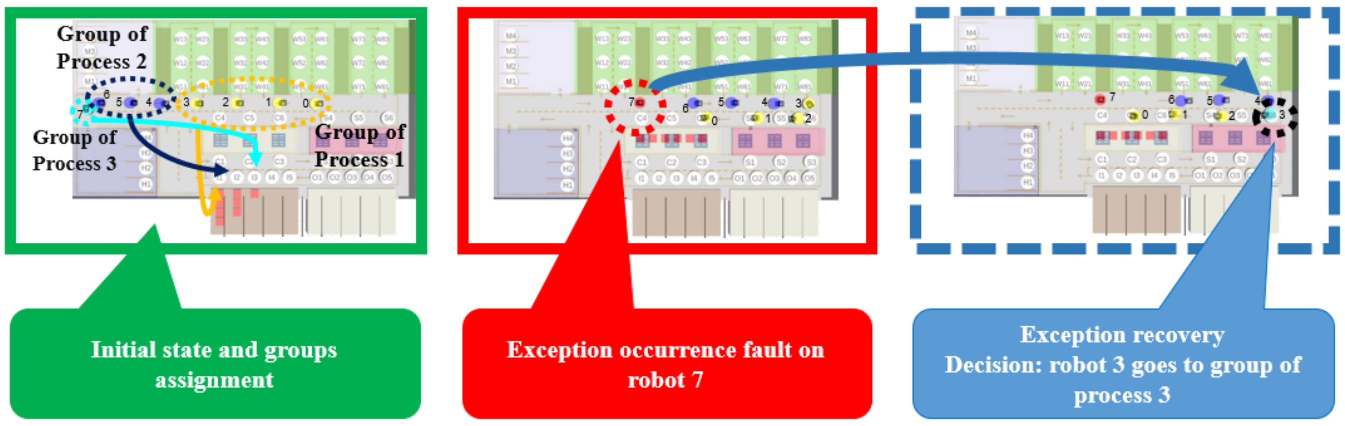
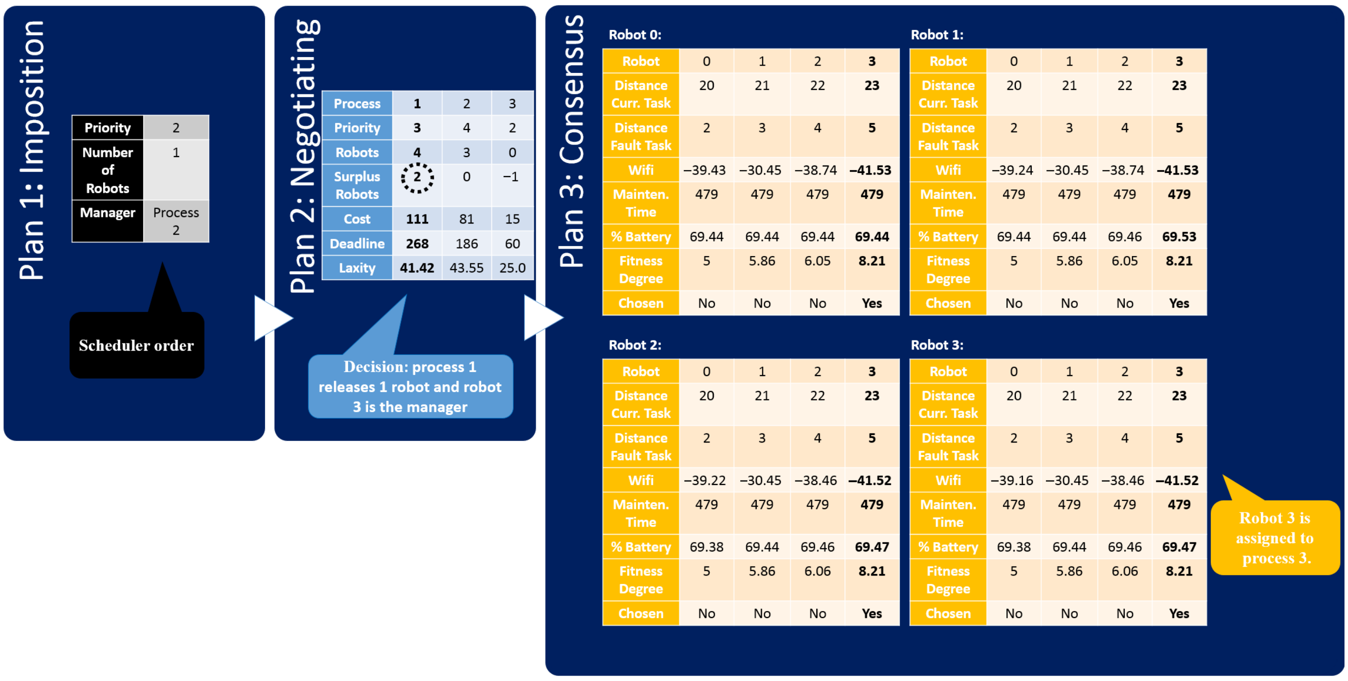

Group of Process 2
Group of Process 3
Group of Process 1
Initial state and groups assignment
Exception occurrence fault on robot 7
Exception recovery Decision: robot 3 goes to group of process 3

Figure 12. Process of experiment 1 running in ARENA.

The second scene of experiment 1 presents a later moment in the experiment, where the robots have already loaded the boxes to the checking sector, and are moving toward the exit of the checking sector to move these boxes to the warehouse; however, robot 7 fails, and starts to be indicated by a red color. The scheduler then signals that a priority 2 process requires a robot, and determines process 2 as the manager because it has the highest priority among the three processes. This order is then passed to the negotiation level, as illustrated in Figure 13.


Plan 1: Imposition
Scheduler order
Plan 2: Negotiating
Decision: process 1 releases 1 robot and robot 3 is the manager
Plan 3: Consensus
Robot 3 is assigned to process 3.

Priority	2
Number of Robots	1
Manager	Process 2

Process	1	2	3
Priority	3	4	2
Robots	4	3	0
Surplus Robots	2	0	−1
Cost	111	81	15
Deadline	268	186	60
Laxity	41.42	43.55	25.0

Robot 0:

Robot	0	1	2	3
Distance Curr. Task	20	21	22	23
Distance Fault Task	2	3	4	5
Wifi	−39.43	−30.45	−38.74	−41.53
Mainten. Time	479	479	479	479
% Battery	69.44	69.44	69.44	69.44
Fitness Degree	5	5.86	6.05	8.21
Chosen	No	No	No	Yes

Robot 1:

Robot	0	1	2	3
Distance Curr. Task	20	21	22	23
Distance Fault Task	2	3	4	5
Wifi	−39.24	−30.45	−38.74	−41.53
Mainten. Time	479	479	479	479
% Battery	69.44	69.44	69.46	69.53
Fitness Degree	5	5.86	6.05	8.21
Chosen	No	No	No	Yes

Robot 2:

Robot	0	1	2	3
Distance Curr. Task	20	21	22	23
Distance Fault Task	2	3	4	5
Wifi	−39.22	−30.45	−38.46	−41.52
Mainten. Time	479	479	479	479
% Battery	69.38	69.44	69.46	69.47
Fitness Degree	5	5.86	6.06	8.21
Chosen	No	No	No	Yes

Robot 3:

Robot	0	1	2	3
Distance Curr. Task	20	21	22	23
Distance Fault Task	2	3	4	5
Wifi	−39.16	−30.45	−38.46	−41.52
Mainten. Time	479	479	479	479
% Battery	69.38	69.44	69.46	69.47
Fitness Degree	5	5.86	6.06	8.21
Chosen	No	No	No	Yes

Figure 13. Ternary-hierarchical decisions of the experiment 1.

Among the three processes, the sole one that has robots left is process 1, with two robots. Thus, the manager's decision is for process 1 to release one robot to process priority 2, with robot 3 as the manager. Each robot in process 1 performs its analysis to determine which robot is best to handle the process fault with priority 2. As the process priority that failed is lower than process 1's priority, process 1 robots will utilize the fuzzy controller for lower priorities. In this case, the group will not give up its best robot to make up for the lack of process with priority 2, but the one that is farther from ending process 1, and also according to other characteristics, such as distance from the task that the failed robot was running, the robot's Wi-Fi signal, time for next maintenance, and robot's battery. In this case, the robot chosen by consensus to leave process group 1 is robot 3, as illustrated in Figure 13.

The replacement of robot 7 by robot 3 is illustrated in the third scene of Figure 12, where robot 3 is now indicated by light blue. At that moment, all robots go to the *checking* sector to load the boxes into the *warehouse* to complete the processes.

The Gantt chart for experiment 1 is illustrated in Figure 14. Process 3 was the last to start, and the first to finish. This happened because process 3 had solely one box to transport, and although the robot failed, the replacement was made by a robot better positioned in the scenario. Process 2 was the second to start and the second to finish as well, with no faults and without needing to attend to other processes, as process 2 had a higher priority than both processes 1 and 3. Finally, the most time-consuming process, because of the greater number of boxes and having attended a process that suffered faults, was process 1.

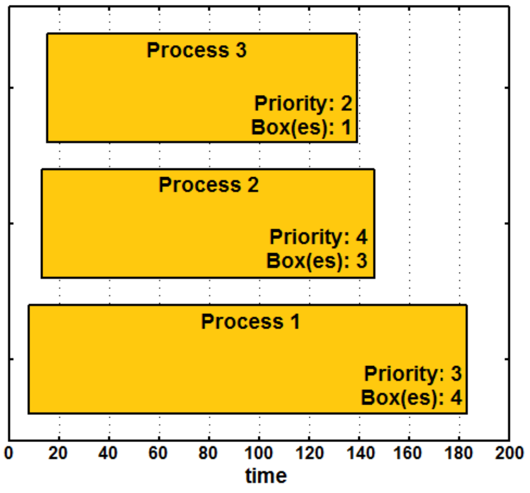


Figure 14. Gantt chart of experiment 1.

7.2. Experiment 2: Consensual Decision Using Fuzzy Controller for Higher Priorities

The second experiment presents a situation in which a process with a lower priority is required to hand over a robot to a process with a higher priority. Unlike Experiment 1, in experiment 2, the fuzzy output is utilized using the fuzzy controller for higher priorities. For experiment 2, it is assumed that there are two boxes in aisle 1, four boxes in aisle 3, and two boxes in aisle 4, loaded into the warehouse by processes 1, 2, and 3, executed previously. Experiment 2 has two processes of the outgoing cargo type and one process of the incoming cargo type, as presented in Table 4.

Table 4. Processes of experiment 2.

#	Type of Process	Priority	Number of Boxes	Initial State	Warehouse Aisle
Process-4	Outgoing Cargo	4	2	O1	4
Process-5	Outgoing Cargo	2	2	O2	1
Process-6	Incoming Cargo	4	4	I4	1

The first scenario of experiment 2, illustrated in Figure 15, indicates where each process group is going. Robots 0 and 1, from process 4, have a yellow color. These robots take two boxes from aisle 4 of the warehouse to the staging sector. Robots 2 and 3, from process 5, are dark blue and take two boxes from aisle 1 of the warehouse to the staging sector. Finally, robots 4, 5, 6, and 7, indicated in light blue, are process 6 robots, and head to state I4 of the incoming cargo sector.

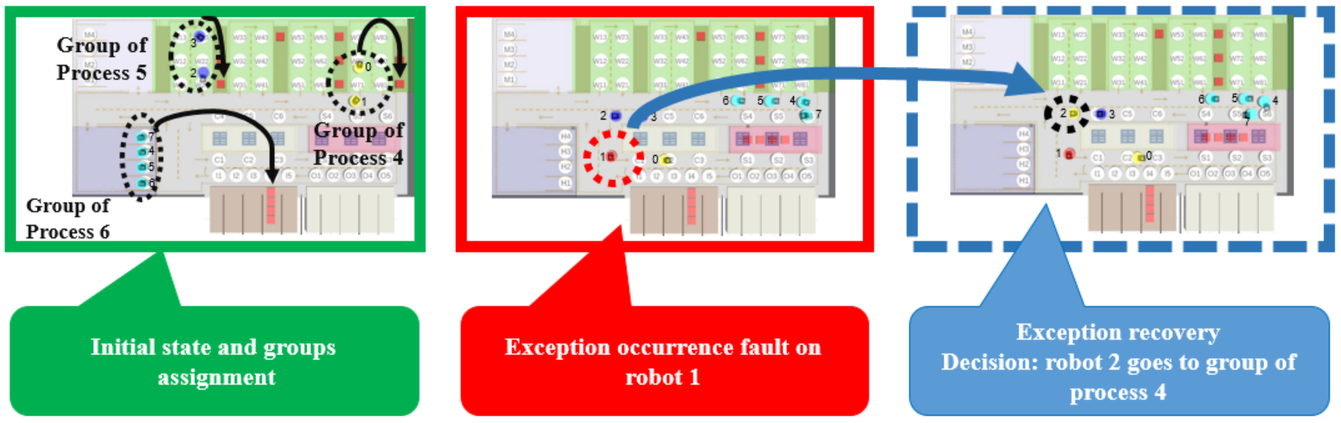
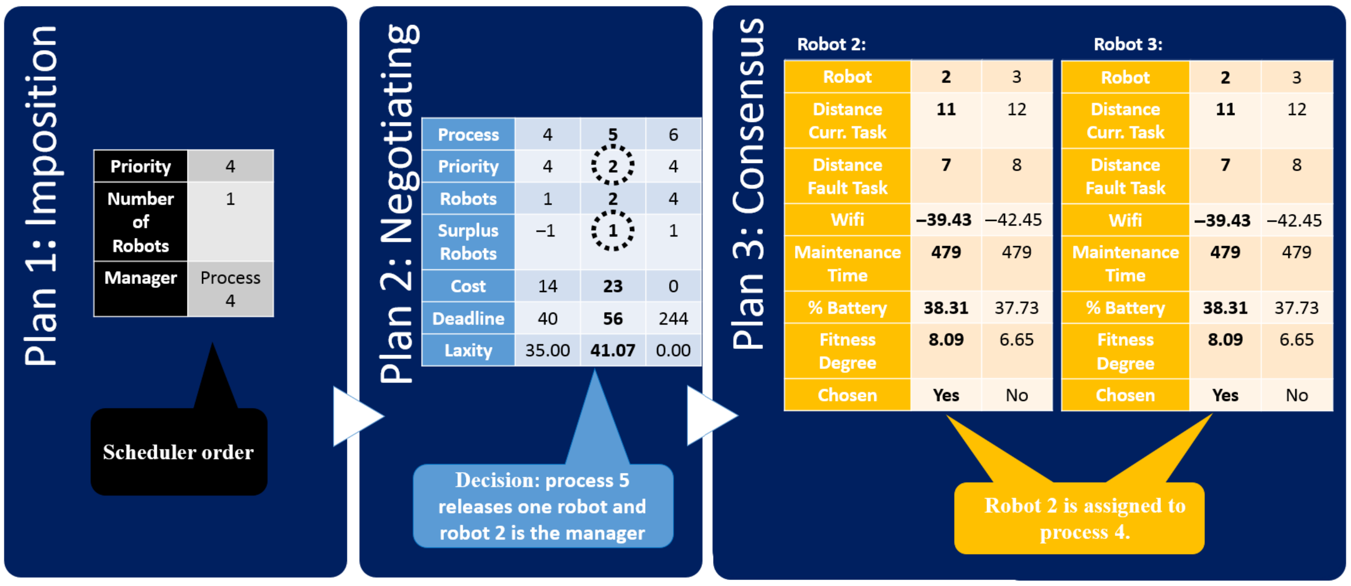

Group of Process 5
Group of Process 4
Group of Process 6
Initial state and groups assignment
Exception occurrence fault on robot 1
Exception recovery
Decision: robot 2 goes to group of process 4

Figure 15. Process of experiment 2 running in ARENA.

After the robots from processes 4 and 5 deliver their boxes in the staging sector, they head to the exit of the staging sector to remove the boxes to take to the Outgoing Cargo sector; however, as illustrated in the second scenario in Figure 15, robot 1 has a fault, and the fault resilience process begins. The scheduler determines that a process with priority 4 has failed and needs one robot, with the manager being process 4, because it is the one with the highest priority and the oldest, as illustrated in Figure 16.


Plan 1: Imposition
Scheduler order
Plan 2: Negotiating
Decision: process 5 releases one robot and robot 2 is the manager
Plan 3: Consensus
Robot 2 is assigned to process 4.

Priority	4
Number of Robots	1
Manager	Process 4

Process	4	**5**	6
Priority	4	**2**	4
Robots	1	**2**	4
Surplus Robots	−1	**1**	1
Cost	14	**23**	0
Deadline	40	**56**	244
Laxity	35.00	**41.07**	0.00

Robot 2:

Robot	**2**	3
Distance Curr. Task	**11**	12
Distance Fault Task	**7**	8
Wifi	**−39.43**	−42.45
Maintenance Time	**479**	479
% Battery	**38.31**	37.73
Fitness Degree	**8.09**	6.65
Chosen	**Yes**	No

Robot 3:

Robot	**2**	3
Distance Curr. Task	**11**	12
Distance Fault Task	**7**	8
Wifi	**−39.43**	−42.45
Maintenance Time	**479**	479
% Battery	**38.31**	37.73
Fitness Degree	**8.09**	6.65
Chosen	**Yes**	No

Figure 16. Ternary-hierarchical decisions of experiment 2.

In the negotiation plan, among the three processes, process 5 fulfills the need because it has one extra robot, and also because it has a lower priority than process 6. The manager determines that process 5 must release one robot from its group, with robot 2 as the manager. In the consensus plan, each robot in process 5 performs its analysis to determine which robot is best to handle the fault of the process with priority 4. As the priority of the process that has failed is higher than that of process 5, the robots of process 5 will utilize the fuzzy controller for higher priorities. In this case, the group will hand over the robot that is closest to the task of the failed robot, in addition to analyzing other characteristics such as distance from the current task, the robot's Wi-Fi signal, time for the next maintenance, and the robot's battery. In this case, the robot chosen by consensus to leave process group 5 is robot 2, as illustrated in Figure 16.

The third scenario in Figure 15 illustrates the moment when robot 2 leaves process group 5 and becomes part of process group 4, altering its color to yellow.

The Gantt plot for experiment 2 is illustrated in Figure 17. The graph illustrates processes 1, 2, and 3, executed before processes 4, 5, and 6, which are of interest in this analysis. Process 4 ended its execution first, despite having a failed robot. This is because the replacement robot was close and did not lead to further delays. Process 5, which was part of the process with solely one robot, took a little longer than process 4 to complete its tasks. Finally, process 6 took the longest time to complete its tasks, as it is an *incoming cargo*-type process, which needs to go through more states before completion.

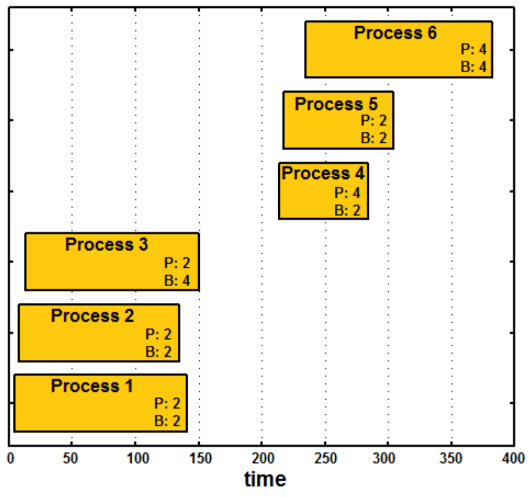


Figure 17. Gantt chart of experiment 2.

8. Comparison with Other Works

This section aims to compare the approaches proposed by Hönig et al. (2018) [13] and Das et al. (2015) [12] with the approach presented in this paper called CFRB. These methods were chosen for comparison because they have available code for evaluation and are easy to adapt to the ARENA warehouse scenario.

Hönig et al. (2018) [13] presented a few solutions based on the CBS algorithm: CBS-TA, ECBS, ECBS-TA, and prioritized planning using SIPP. These solutions aim at a collision-free path configuration and task assignment.

The CBPAE approach of Das et al. (2015) [12] aimed to allocate tasks in a system of multiple heterogeneous autonomous robots deployed in a healthcare institution, based on the principles of auction and consensus.

To compare the works of Hönig et al. (2018) [13] and Das et al. (2015) [12], two objective functions frequently utilized in the literature were considered:

- The sum of the path distances of all robots: the sum of all transitions in the warehouse state machine, as illustrated in Figure 6;
- Makespan: the elapsed time between the completion of the first and last tasks [4]; however, as the work of Das et al. (2015) [12] and Hönig et al. [13] considered the time in milliseconds, in the scheduler cycles of this work, called *ticks*, this comparison is incompatible. Therefore, to have a fair comparison between the methods, we chose to determine the makespan as the longest distance traveled between all robots while performing a task.

Table 5 presents the results of this comparison. The warehouse scenario in this paper was adapted and used for Hönig et al. (2018) [13] and Das et al.'s (2015) [12] approaches. The comparison was performed for the incoming and outgoing cargo processes, covering a total of 42 experiments. Each experiment involved a task that required the transportation of four boxes, and also required four robots.

In Table 5, the scheduling of tasks proposed in this paper presents the sum of the shortest total path distance and smaller makespan for all tasks. This happens because, in the approach of this paper, robots know which collision-free graph they must go through, in addition to the fact that this graph has a single direction.

In summary, the average improvement in CFRB compared to the other methods was −20.43% for the sum of the path distances of the robots and −19.09% for the makespan.

Figure 18 illustrates the average of the sum of the robots' path distances, and the average makespan obtained by comparing the CFRB, ECBS, ECBS-TA, CBS, CBS-TA, and prioritized planning methods using SIPP and CBPAE. In all the situations presented, the CFRB method had the lowest average.

Table 5. Summary of the comparison of CFRB with similar methods.

	Sum of path distances						
	CFRB	**ECBS**	**ECBS-TA**	**CBS**	**CBS-TA**	**PP SIPP**	**CBPAE**
Average Sum of Distances—Incoming Cargo	64.33	+25.39%	+41.45%	+25.39%	+25.39%	+25.39%	+21.76%
Average Sum of Distances—Outgoing Cargo	46.00	+21.74%	+26.09%	+21.74%	+26.09%	+21.74%	+21.74%
Average of Both Processes	55.17	+23.87%	+35.05%	+23.87%	+25.68%	+23.87%	+21.75%
	Makespan						
Makespan Average—Incoming Cargo	17.00	+23.53%	+37.25%	+23.53%	+23.53%	+23.53%	+21.57%
Makespan Average—Outgoing Cargo	12.67	+18.42%	+26.32%	+18.42%	+26.32%	+18.42%	+18.42%
Average of Both Processes	14.83	+21.35%	+32.58%	+21.35%	+24.72%	+21.35%	+20.22%
	Average CFRB Improvement						
Distance	−20.43%						
Makespan	−19.09%						

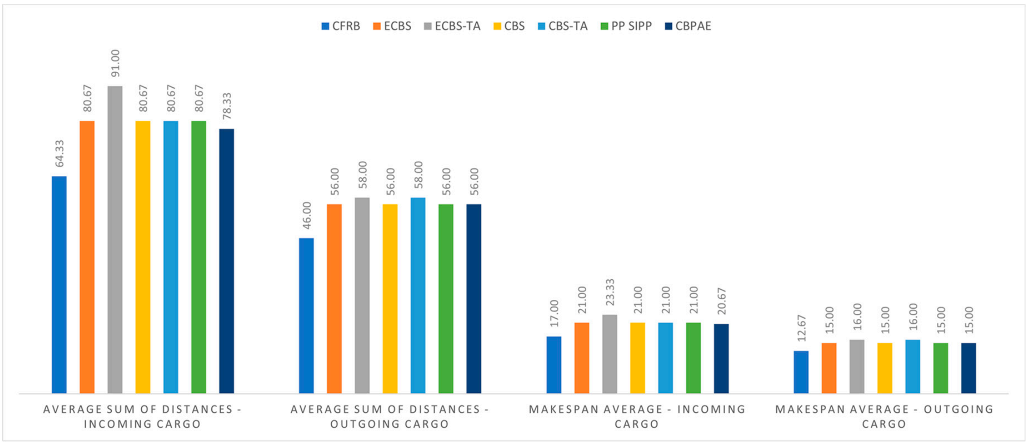


Figure 18. Graphical analysis of the comparison of CFRB with similar methods for incoming cargo and outgoing cargo processes.

9. Conclusions

This paper presented a ternary-hierarchical approach to a consensus-based, fault-resilient behavior called *Consensual Fault-Resilient Behaviour* (CFRB). Thus, fault-resilient collective behavior was obtained by decision making in three hierarchical planes: global, procedural, and individual. This mechanism was useful in smart factories where faults and a fault recovery mechanism were required to avoid stopping their operations.

Two experiments were conducted in a virtual warehouse inspired by a real warehouse to demonstrate how the CFRB works, and how fault-resilient behavior is achieved. Forty-two other experiments were conducted in comparison with the work by Hönig et al. (2018) [13] and Das et al. (2015) [12], which demonstrated the efficiency of the CFRB approach, and yielded better results when dealing with *incoming cargo*-type process tasks, and better or equal results when dealing with *outgoing cargo*-type process tasks.

In future works, it is planned to address the limitations of this work, such as carrying out experiments in more reality-inspired factory scenarios and with a greater number of robots to evaluate the scalability of the proposed approach. Furthermore, the MRS can be expanded to work with heterogeneous robots, considering their different capabilities during process allocation. These improvements aim to increase the robustness and reliability of collective fault-resilient behavior in smart factories.

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

Funding: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001 and in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

ARENA	Augmented Reality to Enhanced Experimentation in Smart Warehouses
CBPAE	Consensus-Based Parallel Auction and Execution
CBS	Conflict-Based Search
CBS-TA	Conflict-Based Search—Task Assignment
CFRB	Consensual Fault-Resilient Behavior
ECBS	Enhanced Conflict-Based Search
ECBS-TA	Enhanced Conflict-Based Search—Task Assignment
MRS	Multi-robot systems
MRTA	Multi-robot task allocation
SIPP	Safe Interval Path Planning

References

1. 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]
2. Nie, Z.; Chen, K.C. Hypergraphical Real-time Multi-Robot Task Allocation in a Smart Factory. *IEEE Trans. Ind. Inf.* **2021**, *18*, 6047–6056. [CrossRef]
3. Khamis, A.; Hussein, A.; Elmogy, A. Multi-robot Task Allocation: A Review of the State-of-the-Art. In *Cooperative Robots and Sensor Networks 2015*; Koubâa, A., Martínez-de Dios, J., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 31–51.
4. Zitouni, F.; Harous, S.; Maamri, R. A Distributed Approach to the Multi-Robot Task Allocation Problem Using the Consensus-Based Bundle Algorithm and Ant Colony System. *IEEE Access* **2020**, *8*, 27479–27494. .. [CrossRef]
5. 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. Inf.* **2018**, *14*, 4548–4556. [CrossRef]
6. Gregory, J.M.; Brookshaw, I.; Fink, J.; Gupta, S.K. An investigation of goal assignment for a heterogeneous robotic team to enable resilient disaster-site exploration. In Proceedings of the 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, China, 11–13 October 2017; pp. 133–140. [CrossRef]
7. Xue, F.; Tang, H.; Su, Q.; Li, T. Task Allocation of Intelligent Warehouse Picking System based on Multi-robot Coalition. *KSII Trans. Internet Inf. Syst.* **2019**, *13*, 3566–3582.
8. Tran, V.P.; Santoso, F.; Garratt, M.A.; Anavatti, S.G. Distributed Artificial Neural Networks-Based Adaptive Strictly Negative Imaginary Formation Controllers for Unmanned Aerial Vehicles in Time-Varying Environments. *IEEE Trans. Ind. Inf.* **2021**, *17*, 3910–3919. [CrossRef]
9. Gerkey, B.P.; Matarić, M.J. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *Int. J. Robot. Res.* **2004**, *23*, 939–954. [CrossRef]
10. Kalempa, V.C.; Piardi, L.; Limeira, M.; De Oliveira, A.S. Fault-Resilient Collective Ternary-Hierarchical Behavior to Smart Factories. *IEEE Access* **2020**, *8*, 176905–176915. [CrossRef]
11. Mahmoud, M.S.; Oyedeji, M. Consensus in multi-agent systems over time-varying networks. *Cyber-Phys. Syst.* **2020**, *6*, 117–145. [CrossRef]
12. Das, G.P.; Mcginnity, T.M.; Coleman, S.A.; Behera, L. A Distributed Task Allocation Algorithm for a Multi-Robot System in Healthcare Facilities. *J. Intell. Robot. Syst.* **2015**, *80*, 33–58. [CrossRef]

13. Hoenig, W.; Kiesel, S.; Tinka, A.; Durham, J.; Ayanian, N. Conflict-Based Search with Optimal Task Assignment. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018.
14. Gregory, J.M.; Al-Hussaini, S.; Gupta, S.K. Heuristics-Based Multi-Agent Task Allocation for Resilient Operations. In Proceedings of the 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Wurzburg, Germany, 2–4 September 2019; pp. 1–8. [CrossRef]
15. Mayya, S.; D'antonio, D.S.; Saldaña, D.; Kumar, V. Resilient Task Allocation in Heterogeneous Multi-Robot Systems. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1327–1334. [CrossRef]
16. Choudhury, S.; Gupta, J.K.; Kochenderfer, M.J.; Sadigh, D.; Bohg, J. Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Auton. Robot.* **2022**, *46*, 231–247. [CrossRef]
17. Kim, S.; Lee, H. Multi-Robot Task Scheduling with Ant Colony Optimization in Antarctic Environments. *Sensors* **2023**, *23*, 751. [CrossRef]
18. Martin, J.G.; Muros, F.J.; Maestre, J.M.; Camacho, E.F. Multi-robot task allocation clustering based on game theory. *Robot. Auton. Syst.* **2023**, *161*, 104314. [CrossRef]
19. Olfati-Saber, R.; Fax, J.A.; Murray, R.M. Consensus and Cooperation in Networked Multi-Agent Systems. *Proc. IEEE* **2007**, *95*, 215–233. [CrossRef]
20. Gulzar, M.M.; Rizvi, S.T.H.; Javed, M.Y.; Munir, U.; Asif, H. Multi-Agent Cooperative Control Consensus: A Comparative Review. *Electronics* **2018**, *7*, 22. [CrossRef]
21. Vasiljević, G.; Petrović, T.; Arbanas, B.; Bogdan, S. Dynamic Median Consensus for Marine Multi-Robot Systems Using Acoustic Communication. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5299–5306. [CrossRef]
22. Brasileiro, F.; Greve, F.; Mostefaoui, A.; Raynal, M. Consensus in One Communication Step. In *Parallel Computing Technologies*; Malyshkin, V., Ed.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 42–50.
23. Lamport, L. Paxos made simple. *ACM Sigact News* **2001**, *32*, 18–25.
24. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In Proceedings of the 2014 USENIX Annual Technical Conference USENIX ATC' 14, Philadelphia, PA, USA, 19–20 June 2014; pp. 305–319.
25. Oh, S.H.; Yang, S.M. A Modified Least-Laxity-First scheduling algorithm for real-time tasks. In Proceedings of the Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No.98EX236), Hiroshima, Japan, 27–29 October 1998; pp. 31–36.
26. Passino, K.M.; Yurkovich, S.; Reinfrank, M. *Fuzzy Control*; Addison-Wesley Reading: Boston, MA, USA, 1998; Volume 42.
27. Piardi, L.; Kalempa, V.C.; Limeira, M.; de Oliveira, A.S.; Leitão, P. ARENA—Augmented Reality to Enhanced Experimentation in Smart Warehouses. *Sensors* **2019**, *19*, 4308. [CrossRef]
28. Zijm, H.; Klumpp, M.; Heragu, S.; Regattieri, A. Operations, Logistics and Supply Chain Management: Definitions and Objectives. In *Operations, Logistics and Supply Chain Management*; Zijm, H., Klumpp, M., Regattieri, A., Heragu, S., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 27–42.
29. de Koster, R.; Le-Duc, T.; Roodbergen, K.J. Design and control of warehouse order picking: A literature review. *Eur. J. Oper. Res.* **2007**, *182*, 481–501. [CrossRef]
30. Limeira, M.A.; Piardi, L.; Kalempa, V.C.; de Oliveira, A.S.; Leitão, P. WsBot: A Tiny, Low-Cost Swarm Robot for Experimentation on Industry 4.0. In Proceedings of the 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), RioGrande, Brazil, 23–25 October 2019; pp. 293–298.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.