Multi-Robot Coordination for a Heterogeneous Fleet of Robots

Diogo Pereira[1,2], Diogo Matos[2], Paulo Rebelo[2], Fillipe Ribeiro[3], Pedro Costa[1,2], and José Lima[2,4,5]

[1] Faculty of Engineering of University of Porto, Portugal,
diogo.a.pereira@inesctec.pt,
pedrogc@fe.up.pt
[2] INESC-TEC, Centre for Robotics in Industry and Intelligent Systems, Portugal
{diogo.m.matos,paulo.m.rebelo}@inesctec.pt
[3] JPM Industry, Zona industrial do Rossio,
3731-901 Vale de Cambra, Portugal
Fillipe.Ribeiro@jpm.pt
[4] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Portugal
[5] Laboratório para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Portugal
jllima@ipb.pt

Abstract. There is an increasing need for autonomous mobile robots (AMRs) in industrial environments. The capability of autonomous movement and transportation of items in industrial environments provides a significant increase in productivity and efficiency. This need, coupled with the possibility of controlling groups of heterogeneous robots, simultaneously addresses a wide range of tasks with different characteristics in the same environment, further increasing productivity and efficiency. This paper will present an implementation of a system capable of coordinating a fleet of heterogeneous robots with robustness. The implemented system must be able to plan a safe and efficient path for these different robots. To achieve this task, the TEA* (Time Enhanced A*) graph search algorithm will be used to coordinate the paths of the robots, along with a graph decomposition module that will be used to improve the efficiency and safety of this system. The project was implemented using the ROS framework and the Stage simulator. Results validate the proposed approach since the system was able to coordinate a fleet of robots in various different tests efficiently and safely, given the heterogeneity of the robots.

1 Introduction

With the advances in the characteristics of autonomous vehicles, such as their autonomy or weight, along with the reduction in their cost, there has been an increase in their usage in industrial environments for a wide range of tasks. Furthermore, using systems made up of a group of vehicles with differences in their brands or morphological structures is becoming more common. The ability

to coordinate a multi-robot system (MRS) with heterogeneous robots improves the efficiency of these vehicles in tasks where different robots are needed to perform them. An MRS is composed of multiple robots capable of executing tasks that require movement from one place to another. The coordination between various robots with the same morphology is a complex task that is still being improved today. However, a system capable of controlling a fleet of robots that are not homogeneous adds another layer of complexity to the problem and is increasingly a necessity for today's industry, whether it is controlling robots of different brands or controlling robots that perform entirely different roles in the same system. Path planning is a crucial aspect required for achieving coordinated and safe navigation in a system composed of multiple vehicles. With this, it is necessary to develop algorithms that can resolve the need for a vehicle/robot to reach a specific location without human interaction and without causing any collisions with obstacles or other robots in the MRS or deadlocks. In addition, reaching the destination is not the only goal. Finding an efficient solution is an essential aspect of these algorithms with crucial factors such as distance traveled, energy spent, or time spent. Because of this, many of the path planning methods are based on various algorithms that, on the one hand, aim to find the set of safe paths to the goal and, on the other hand, aim to link which of these possibilities lead to the most efficient solution [3]. The main goal of this work is to implement a system capable of coordinating heterogeneous robots based on the TEA* graph search algorithm. This goal is achieved by implementing a graph decomposition algorithm that will decompose the graph used by the TEA* algorithm before planning the paths and make the system more stable and robust when using a fleet of heterogeneous robots. The implemented path planning is applied to a real-time system that already exists, developed by [4] and [6] which already has the tools needed for graph based path coordination systems, such as position detection and controllers to move the robots along specific lines. Additionally, the ROS framework will be used to implement the entirety of the system. Finally, a real-time supervisor will also be implemented. This supervisor will be used to verify that all robots in the system are following the path that was planned and ensure that they do not pose safety concerns.

2 Related Works

Various heterogeneous MRS coordination methods have already been explored in the field of MRS coordination. The methods may vary a lot in terms of approaches. However, the decision-making process, or rather, whether a centralized or a decentralized group architecture is used, seems to be the most significant point of contention. For example, [2] uses a fully distributed method to coordinate the various robots in the system which communicate with each other and use a coordination protocol to coordinate with each other and a monitor connected to the communication network to supervise the behavior of the robots. A typical MRS that uses a centralized architecture is the GOFER [1] which uses a central task planning and scheduling system by viewing all the tasks that need

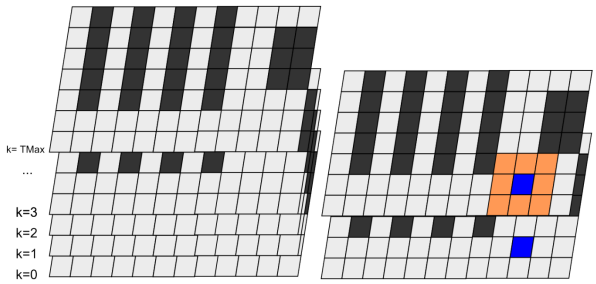
to be resolved and what robots are available to perform those tasks. Furthermore, a hybrid approach is also possible, but the solutions they reach are often sub-optimal, as seen in the following study about MRS coordination [8]. The algorithm used in this paper's project will focus on a more centralized approach due to the nature of the TEA* algorithm.

3 Path Planning

Path planning is one of the most crucial aspects of autonomous movement systems. Path planning algorithms aim to establish paths between start and end points allowing mobile robots to navigate autonomously and safely in a working environment. They generate an efficient solution for a system to change from an initial state to a final state by avoiding static and dynamic obstacles in the environment [3]. As such, a big focus of this paper will be on the implementation of a robust path planning system capable of coordinating heterogeneous robots. To plan the paths for the robots in the system, a graph was used to define within the environment where the robots could travel or not. Using this distinction, it is possible to use graph search algorithms to plan paths for the robots. This paper will use the TEA* algorithm to plan paths within that graph and use it to coordinate paths between the robots. The main advantage of the TEA* algorithm comes from introducing a third temporal layer capable of planning a path that does not collide with any other robot in time or reach other blocking problems such as deadlocks. This feature, coupled with the capabilities of obtaining optimal paths from the A* algorithm, the TEA* accomplishes all the necessary requisites for a robust path planning algorithm.

3.1 Time Enhanced A* (TEA*)

Taking the A* graph search algorithm and adding a third temporal layer enables the neighbor node that will be explored to belong to the next temporal layer, as seen in Figure 1. With this, it is possible to associate each link to the instant of time it will be occupied, preventing collision between robots or paths being blocked by other robots through deadlocks and creating far more efficient routes. Furthermore, in addition to exploring the neighbor nodes in the next temporal layer, the node itself is also "explored" and re-added to the list with a small increment to the cost. This addition is helpful for cases where the robot staying in place is more efficient than taking a longer route, such as having a robot wait for another robot to pass through instead of immediately attempting to take a longer route that would take longer. The run-through of the TEA* algorithm is pretty similar to the A* algorithm. Firstly, instead of a single graph and states for each node, there is an array of graphs, and each node has a specific state and stored cost at a specific time K which is known as "step". The algorithm runs for as many times as many robots that exist in the system, and, following a priority order, each robot plans its path through time and blocks that path to the lower priority robots by setting the nodes that belong to its path to



(a) Input Map over the time. (b) Analysed Neighbor Cells

Fig. 1. Input map over the various temporal layers (a) and the analyzed neighbor cells focused on the AGV position (b) [7].

the state *OBSTACLE*. The next robots plan their path, considering the new *OBSTACLE* nodes in time. It is essential to note that this algorithm is hard to scale. This problem comes from the necessity of storing a lot of memory regarding the array of graphs used to plan the path in time [5]. On top of that, the algorithm also requires a lot of computational resources, namely read/write speed to and from memory, as the algorithm has to look for the next neighbor node to be explored constantly and update the values of the costs/states. One of the steps to reduce this potential read/write speed problem was utilizing a low-time complexity algorithm to store the graph's data. As such, in a graph, the nodes and links were stored in a C++ map container from the C++ standard library, where the keys of that map were the ID of the node/link, and the graph itself was stored in a vector of graphs. The map container was helpful because the time complexity, in Big-O notation, of searching for a member given the key is $log(n)$. Using such a container is better than using more usual containers such as C++ vectors or double-ended queues (deque), which would have at least $O(n)$ temporal complexity. Regarding using a vector to store the temporal layers of the graph, since the steps go from 0 to K_{max} sequentially, the temporal complexity of accessing a specific temporal graph in that vector is $O(1)$.

To further ensure safety, it is also common to give robots a bigger space gap between them. For example, instead of setting only the current node that the robot is, in time, as *OBSTACLE*, the neighbor nodes around that node are also set as *OBSTACLE*. Additionally, it is also common to set as *OBSTACLE* a few previous nodes in time as *OBSTACLE*. For example, if it is defined that three nodes in time are to be set as *OBSTACLE* then if a robot has a path $P_A = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5\}$, when planning the path for that robot, in the step K that the node (4) is expected to be *OBSTACLE*, both the previous nodes (3) and (2) are also set as *OBSTACLE* to ensure further safety. These gaps are implemented in a way to depend on a safety parameter called *SAFETY_GAP*, which sets both the number of previous path nodes that are set to *OBSTACLE* but also the depth of the neighbors of the current path node to be set as *OBSTACLE*. For example, if

SAFETY_GAP is 4 and path is $P_A = \{1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5\}$, in the step where node (4) is going to expected to have the robot, the nodes (4), (3), (2) and (1) are set to *OBSTACLE* and the neighbor nodes of the node (4) and the neighbors or those neighbors are also set to *OBSTACLE*. Additional examples can be seen in Figure 2. A relevant note is that the smaller the *SAFETY_GAP* value, the more efficient the planned path is, but it also becomes less safe. With this, the TEA* algorithm can plan efficient and collision-free paths for the various robots in the system.

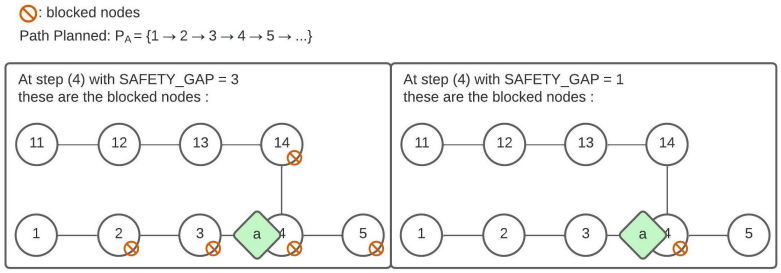


Fig. 2. Additional examples of the path planning *OBSTACLE* setup

3.2 Robot Priority Swapping

Even though the base TEA* algorithm can plan paths for all the robots in the system, extra tweaks exist to optimize it further. One of these can be the addition of a priority swap for the robots. This swap is an essential feature as sometimes the base algorithm cannot find a path for the current setup of robot priorities. For example, if there is a robot priority setup where the higher priority robot, after planning its path, blocks any possible path for the lower priority robot that means it is impossible to complete that order. But, if the priorities are switched, there is still a possibility of paths being found for all robots. Thus, having a list of possible priority setups can help eliminate this problem and find paths that the base algorithm would be unable to.

3.3 Moving idle robots

Another relevant feature to incorporate into the base algorithm is moving other robots when idle and in the way of another robot. This happens, for example, when there is a robot idle in a position that another robot needs to complete its path. By creating an extra order to that idle robot, if available to move, it is possible to move that robot and enable the passage of the robot that was being blocked. This solution can be achieved by first detecting if a path is not found

because another lower priority robot blocks the destination by being stationary and plans the path ignoring that stationary robot. Then, by looking into the nearby stations of the stationary robot, it is possible to attempt to move the idle robot to the closest unused station, thus unblocking the way. This feature should only be applied to robots of lower priority, so combining this method with the previous process of priority swapping helps achieve more possible paths.

4 System Implementation

In order to implement and test a system capable of coordinating a group of robots, it was necessary to recreate an industrial-like environment where the conditions are similar to a real one. Furthermore, this system needs to work with the ROS framework, which is a requirement imposed. Using the *Stage* simulator, various environments were set up to host the robots that were going to be coordinated by the implemented system. With this simulation, it was possible to simulate a nearly real industrial environment by creating walls, corridors and even objects that could collide with the robots. Furthermore, these objects could be moved to create obstacles for the robot's paths and test safety features. The robots themselves were also simulated in this simulator. The robots could localize themselves using simulated LIDAR sensors on their front and back, along with collision detectors that could detect if the robots had collided with anything, forcing them to stop.

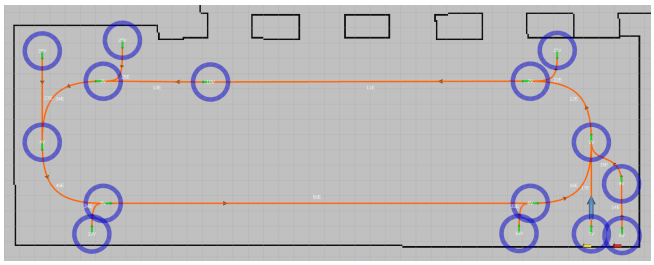


Fig. 3. An undecomposed small map and its graph used in simulations.

To view all the data being circulated, such as the current robot's estimated location and the graph points, the RVIZ visualization tool from the ROS framework is used. With this tool, it is possible to visualize the robots running through the graph and have the option to send orders directed through the RVIZ tool for robots to move from one point to another. This tool can also be used to edit and save graphs instead of manually inserting coordinates and values in the manual definition. In Figure 3 an example of a RVIZ visualization is presented where it is possible to see the various graph links as red lines, the nodes as blue circles

and the robot's position as the big blue arrow. In figure 4 a block diagram of the implemented system is presented.

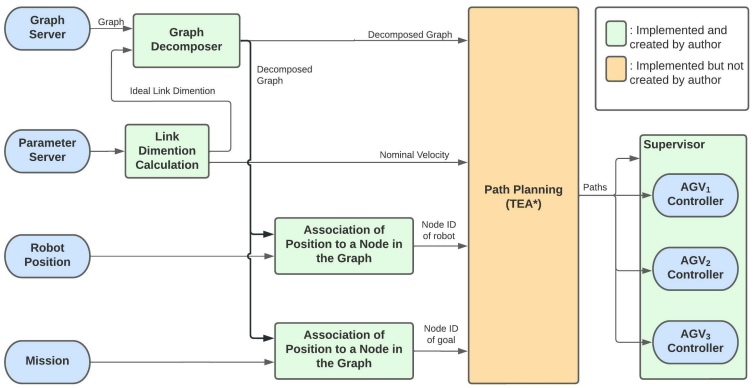


Fig. 4. Diagram of the system architecture implemented

In sum, a short description of the modules of the system implementation can be given as such:

- Firstly, a graph is created. In this project a graph is created using a list of nodes and links that describe the various characteristics of these (curves, length, ID, etc.) and storing them into a YAML file.
- Given a parameter server that holds various characteristics (size and traction type), an ideal link dimension is calculated for the decomposed graph.
- Before the system starts, these graphs can be optionally decomposed. This decomposition means that the graph links are subsequently divided so that all links have a similar length.
- Once the system is started and given an order for a robot to move, both the robot's current position and the order's destination position are associated with their position relative to a node in the graph.
- Finally, a path can be calculated by knowing the node to which the robot belongs and its destination node. This path planning also considers notable exceptions in an exception table. Once a path is calculated, it is sent to each corresponding robot.
- On top of all, a supervisor also exists to ensure that the robots are following the paths planned without the risk of problems such as collisions and deadlocks.

4.1 Graph

The graphs are defined using the YAML serialization language and specifying various fields that describe the characteristics of various nodes and links that

create the graph. For nodes the fields are: ID, angle and position. For links the fields are: ID, origin and destination IDs, velocity forward/backward and other parameters for the description of a Bézier curve.

4.2 Graph Decomposition

Due to how the TEA* algorithm works, the path planning returns a sequence of edges or vertices that define a path. Therefore, if the various edges' sizes are very different, the time it takes for a robot to travel an edge may be substantially different from another robot traveling to another edge, which is even more substantial when the fact that different edges can have different velocities is taken into account. This difference can be dangerous as TEA* assumes that the time it takes to go through an edge is the same for all edges, and if that is not true, the path planned cannot ensure that there is no collision between robots or that potential deadlocks can occur. In Figure 5 presents an example of this problem.

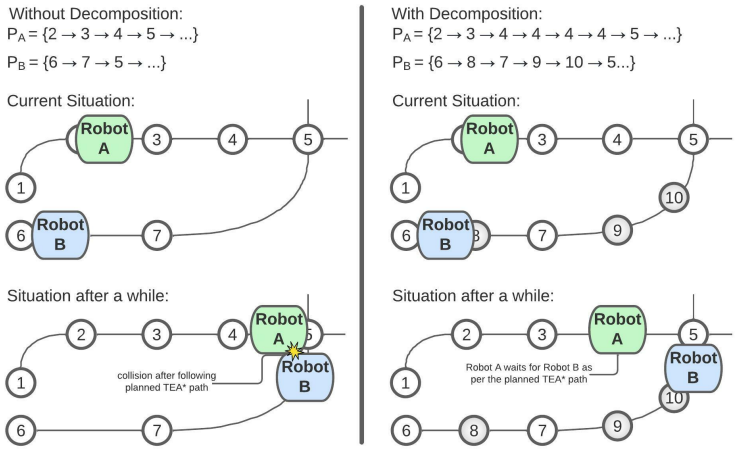


Fig. 5. Example of collision due to lack of graph decomposition

Because of these problems, to ensure that the robots are proceed safely through their path, it is necessary to decompose longer links into smaller links to minimize the differences of time that it takes to travel between the different edges. Furthermore, this decomposition also optimizes the routes for the robots, as decomposing a lengthier edge into smaller ones allows for more than one robot to travel that edge sequentially. To implement this, firstly, a decomposition algorithm was created. This algorithm's goal was to, given a specific step length,

divide all links in a graph in a way that minimizes the error of the new division of the links when comparing to the specific step. The errors of this division are all taken and the average of this error is calculated. Using this error, and cycling between an interval of values, it is possible to use this algorithm to obtain the value of specific step length that minimizes the average of the error when decomposing the links. After obtaining that value the graph is decomposed using the initial algorithm.

4.3 Supervision

Because in an industrial environment, many variables exist at all times, a supervisor for such a system as a path planner is crucial to ensure safety and efficiency. As such, a supervisor was created to observe the planned path by the system and ensure that all robots within the system are following that path.

Assuming a worst-case scenario that all links are the same size as the robots (and never smaller than the robots), and two robots are following each other in a path, as long as they are not over one step ahead of the other robot regarding the route planned, they will not collide. If such a thing happens, the supervisor acts and triggers a new path planning. In Figure 6 this is presented.

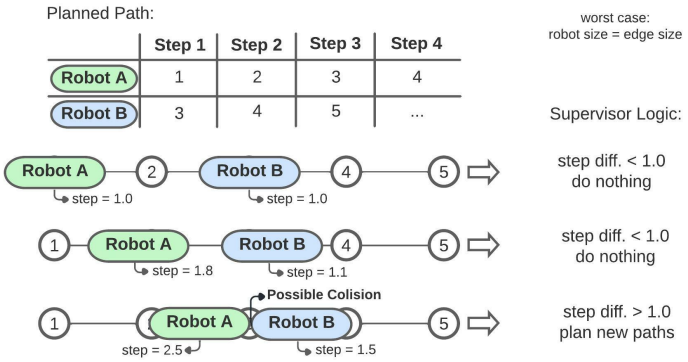


Fig. 6. Supervisor interventions in the implemented system

Because of how the system works, in the path planner, a robot belongs to a node until it reaches another. Looking at Figure 6 when the replanning is triggered, the robot (a) that is still detected as being in node (2) will have a path that will make the robot (a) go back to node (2) and stays there for one step. In contrast, the other robot is detected as being in node (3) and plans a new path from that node, which means the robot (b) keeps advancing as he was, thus eliminating possibilities for collisions.

5 Results and discussion

To ensure that the other tests could be done safely, the basic features of the TEA* were tested along with the extra features added to complement it[6]. Once all these tests were completed and working as intended, the supervisor was tested. To test the supervisor four robots were set on a path to follow each other in a straight line to verify the system's response when subjected to a supervisor re-planning of the paths[7]. The robots were set in a specific order where the blue robot was at the front, followed by the red, yellow and orange robots in this order. When in movement, one of the robots was manually blocked from moving so it could trigger a new path planning and verify its effects. The test results are shown in the table 1.

Line Test Blocking Yellow Robot				
Distance traveled by:	Blue Robot (m)	Red Robot (m)	Yellow Robot (m)	Orange Robot (m)
Decomposed	22,27	22,34	23,85	24,53
Undecomposed	22,41	22,39	24,01	26,16

Line Test Blocking Red Robot				
Distance traveled by:	Blue Robot (m)	Red Robot (m)	Yellow Robot (m)	Orange Robot (m)
Decomposed	22,29	22,91	24,22	25,09
Undecomposed	22,56	23,13	27,54	31,00

Table 1. Effects of supervisor interventions in decomposed vs undecomposed graphs

Looking at these results, it is possible to conclude that the decomposition of the graph not only makes the system safer but also faster since the distance traveled by the robots affected by the supervisor's re-planning of the paths is lower when the graph is decomposed. This is because the robots affected have to take much longer routes if their paths are re-planned when the graph is not decomposed and, as such, the orders take quite a bit longer to complete.

Some tests were also made to test the graph decomposition when attempting to coordinate a fleet of heterogeneous robots. These tests are composed of generating orders to send four heterogeneous robots simultaneously to different places in the map making them follow approximate equivalent paths with the same number of nodes[8].

In sum, the heterogeneity of the robots affects the system in two major aspects. On one hand, the size of the robots and their morphologies affect the decomposition parameters, which affects the whole decomposition. On the other hand, the traction type affects the irregularity of the velocity parameters, making them have different behaviors when moving, such as slippage or different behaviors when turning. While the first problem can be solved by re-decomposing the

[6] Video of basic robot coordination tests: `https://youtu.be/EEKb_r0efJY`
[7] Video of supervisor tests: `https://youtu.be/8qBK5eCUi4o`
[8] Video of heterogeneous robot coordination tests: `https://youtu.be/RDjN2HEko40`

graph with the different parameters, the second one cannot be directly solved the same way and must depend on the supervisor to act when needed to ensure that everything is kept safe.

In the tests, the supervisor only seemed to act about 1-2 times when the system was coordinating four robots at the same time with long paths composed of about 50 nodes. These tests were an overall success as the amount of times the supervisor had to act was pretty low. Furthermore, the supervisor was acting in a strict mode where no matter where the robots were in the map, the difference limit for it to act was always 1.0 steps. This, in a more real scenario, would have reduced the number of interventions by the supervisor substantially, potentially to zero. Comparing to some tests done with using homogeneous robots, as seen in table 2, the amount of times the supervisor had to act with the heterogeneous robots was a bit higher.

	Homogeneous Robots		Heterogeneous Robots	
	Undecomposed	Decomposed	Undecomposed	Decomposed
Avg. Actuation of Supervisor	4.4	1	5.2	1.8

Table 2. The average number of actuations by the supervisor when attempting to coordinate a fleet of heterogeneous vs homogeneous robots.

This is expected, not only due to the different traction type explained earlier but also because of the difference in sizes of the robots. The bigger the robots, the worse the decomposition of the graph. This is because the decomposition algorithm aims to have a link sizes always higher than the largest robot in the system so that the supervisor works as expected. With this, the higher the minimum size of the links, the higher the error of decomposition will be, and, as such, results are expected to be worse.

6 Conclusions and future work

Looking at the results obtained during the project's test phase, it is possible to conclude various aspects regarding the accomplishment of various goals. The first one is the implementation of the TEA* algorithm was done successfully allowing for the other modules to be tested without problems arising. Regarding the implemented supervisor, it was able to keep the robots from deviating too much from the expected temporal planned path by the TEA* algorithm by triggering a path re-planning for all the robots in the system to prevent extreme deviations that could lead to collisions or deadlocks. Even though the supervisor sometimes had to act four or five times to complete a full order, in no case the supervisor failed to plan new paths and complete all orders. Regarding the graph decomposition algorithm, the results proved to be quite favorable. The algorithm was able to decompose the graph's links leading to the robot movement deviating way less from the expected path generated by the path planning system. The

supervisor had to act a considerably low amount of times when the graph used to plan the paths of the robots was decomposed. Considering the extreme case the supervisor was subjected to act by removing the feature that only has the supervisor work on a tight step difference when the paths of two robots intercept, these results were pretty good.

These results together make the system quite robust to the heterogeneity of the group of robots. Although the system proved quite effective in a simulation environment, it was impossible to test it in a real scenario due to time constraints. As such, it is hard to verify that the system would be able to work as expected using real robots. Furthermore, although considerably efficient and safe, the paths obtained by the implemented system are still not fully optimal. Finally, the system also suffered from some common vulnerabilities of a centralized system, such as having a single point of failure.

Acknowledgements

This work is financed by the ERDF - European Regional Development Fund, through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme under the Portugal 2020 Partnership Agreement, within project CrossLog, with reference POCI-01-0247-FEDER-039895.

References

1. Caloud, P., Choi, W., Latombe, J.C., Le Pape, C., Yim, M.: Indoor automation with many mobile robots. In: EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications. pp. 67–72 vol.1 (1990). https://doi.org/10.1109/IROS.1990.262370
2. Iocchi, L., Nardi, D., Piaggio, M., Sgorbissa, A.: Distributed coordination in heterogeneous multi-robot systems. Autonomous Robots **15**, 155–168 (01 2003). https://doi.org/10.1023/A:1025589008533
3. Ángel Madridano, Al-Kaff, A., Martín, D., de la Escalera, A.: Trajectory planning for multi-robot systems: Methods and applications (2021). https://doi.org/10.1016/j.eswa.2021.114660
4. Matos, D., Costa, P., Lima, J., Costa, P.: Multi agv coordination tolerant to communication failures. Robotics **10**(2) (2021). https://doi.org/10.3390/robotics10020055, https://www.mdpi.com/2218-6581/10/2/55
5. Moura, P., Costa, P., Lima, J., Costa, P.: A temporal optimization applied to time enhanced a*. vol. 2116 (2019). https://doi.org/10.1063/1.5114225
6. Rocha, C., Sousa, I., Ferreira, F., Sobreira, H., Lima, J., Veiga, G., Moreira, A.P.: Development of an autonomous mobile towing vehicle for logistic tasks. In: 4th Iberian Robotics Conference, ROBOT 2019. pp. 669– 681 (2020). https://doi.org/10.1007/978-3-030-35990-4$_5$4
7. Santos, J., Costa, P., Rocha, L.F., Moreira, A.P., Veiga, G.: Time enhanced a: Towards the development of a new approach for multi-robot coordination. vol. 2015-June (2015). https://doi.org/10.1109/ICIT.2015.7125589
8. Yan, Z., Jouandeau, N., Cherif, A.A.: A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems **10**(12), 399 (2013). https://doi.org/10.5772/57313