Degree project in Computer Science and Engineering
First cycle, 15 credits

You May Pass

A Comparative study of robot deadlock resolution strategies in corridors

ANNA AKAPYAN, KRISTIN ROSEN

You May Pass

A comparative study of robot deadlock resolution strategies in corridors.

Anna Akopyan, Kristin Rosen

Degree Project in Computer Science and Engineering, First Cycle 15 credits
Date: May 21, 2024
Supervisor: Prof. Jana Tumova
Examiner: Prof. Pawel Herman
Swedish title: Du Får Passera: Strategier för lösning av dödlägen bland robotar i korridorer
School of Electrical Engineering and Computer Science

Abstract

The steady increase of autonomous robots in the workplace has lead to an escalating need for these robots to be able to coordinate their movement in shared enclosed spaces. When robots drive along a corridor, they are likely to meet on a collision course, which can lead to a deadlock where the robots cannot continue forward without resolution of the issue. Inspired by this development we study the performance of three different deadlock resolution strategies to compare their advantages and disadvantages. The strategies each demonstrate a different style of deadlock resolution: basic traffic rules, randomization and priority based resolution. Tests were performed in a simulated corridor environment, where the priority based resolution showed the altogether best results, although it also possessed major disadvantages. Based on our findings, we therefore designed an improved strategy, which in testing outperform all original three.

Keywords

Deadlock Resolution, Robot Coordination, Decentralized Robot Systems, Deadlock in Corridors, Autonomous Robots.

Sammanfattning

Den konstanta ökningen av autonoma robotar på arbetsplatser har lett till förhöjda krav på dessa robotars förmåga att koordinera sina rörelser i delade avgränsade områden. När robotar kör längs en korridor är det troligt att de möts, vilket kan leda till dödlägen där robotarna inte kan fortsätta framåt utan lösning av konflikten. Inspirerade av denna utveckling studerar vi prestandan hos tre olika strategier för lösning av dödlägen, i syfte att jämföra deras styrkor och svagheter. Varje strategi visar på en egen variant av dödlägeslösnng: grundläggande trafikregler, randomisering och prioritetsbaserad lösning. Tester utfördes i en simulerad korridorsmiljö, där den prioritetsbaserade strategin uppvisade sammantaget bäst resultat, dock inte utan betydande svagheter. Baserad på våra iakttagelse utvecklade vi därför en förbättrad strategi, som i testning överträffade alla tre originalstrategier.

Nyckelord

Hantering av Dödlägen, Robotkoordination, Decentraliserade Robotsystem, Dödlägen i Korridorer, Autonoma Robotar.

Contents

1 Introduction

In the 21st century, robots are moving further and further from science fiction stories into real life usage. With applications ranging from our homes to tech-heavy industries to social human-centered environments, a wide variety of robots have emerged. As robots become increasingly present in everyday work scenarios, their movements require more coordination to avoid conflicts and collisions. While some robot systems can quite easily make mutual plans, other robots might need to communicate their movement with each other.

Workspaces that incorporate robots in their daily working environment are already encountering conflicts in both human-robot and robot-robot interactions. For example when two robots face each other in a corridor - who should pass and who should wait for the other? Failure to handle conflicts may cause deadlock, as in the case of the hospital robots installed at Karolinska Sjukhuset in Stockholm in 2016 [1]: The Karolinska robots were so-called AGV:s (Automatic Guided Vehicles), unmanned vehicles guided by wires built into the floor.[2] The AGV:s transported deliveries and material around the building and shared their environment with the hospital staff, which meant that possible collisions with humans had to be detected and avoided. Upon meeting humans, these robots would stop and ask the human to move, the robots staying still until their path was free. But this strategy had a major problem: When two robots blocked *each other*, they lacked strategy for handling this. They became stuck in an endless cycle of repeating "You are in the way, please step aside" to each other, neither moving – i.e. a *deadlock*. (See Appendix A for a video depicting the situation and the mentioned robots)

1.1 Problem statement

The real life example above inspired us to closer investigate this issue. How should robots best handle this type of scenario? These problems of coordination need solving for smooth cooperation and work. Our study explores and compares possible deadlock-resolving strategies, and could be used as a stepping stone for studying movement coordination of decentralized robot systems in non-planned scenarios and meetings. A functioning control program for handling coordination conflicts can be indispensable in multi-robot environments, especially if the robots are of different types and models. There are several aspects that could define an efficient coordination strategy, such as the time needed to solve a conflict, or the amount of excess space used. Which aspect is the most important could vary depending on context and situation. Therefore, it is interesting to explore different conflict resolution strategies, and how they fare regarding different aspects.

The frequency of and difficulty in solving deadlocks varies greatly between *centralized* and *decentralized* robots. In a centralized system, robots either "share a brain", each having access to the same information, such as route plans and positions; or the less informed robots communicate with a more "all-knowing"

coordinator agent, which handles their coordination. A decentralized control program on the other hand, is built on the robots having individuality. Here each robot is an independent agent that makes its own decisions and since the robots do not share memory or any control panel, they must resolve a deadlock through interaction. For this reason, deadlock-resolution of the aforementioned kind is of the most interest for decentralized robot systems, which is why we have chosen to study such systems.

1.2 Aim and Research Question

The aim of this study is to compare and analyze deadlock resolution strategies. It handles a situation of two decentralized robots moving on a collision course towards each other in a corridor. The corridor is wide enough for two, but not three, robots to drive alongside each other. A simple deadlock scenario in such a situation is illustrated in Fig.1 below, with red and blue rectangles representing the two robots.

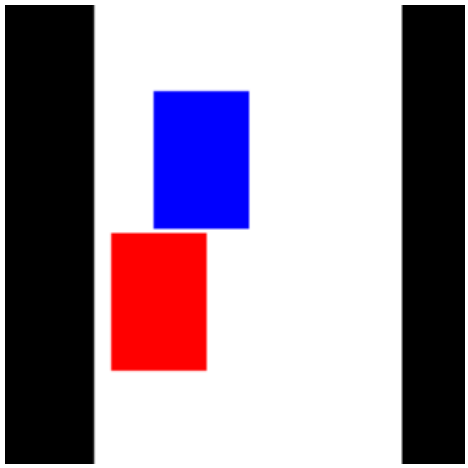


Figure 1: Two robots in deadlock.

1.2.1 Research Question

Our research question is divided into two parts:
1) "How do different deadlock resolution strategies compare in terms of effectiveness and time and space efficiency, in meetings between two decentralized robots in a corridor?"

2) "How could we design a more optimized strategy based on our analysis of the results from part 1."

"Effectiveness" in part 1 refers to the robots' success rate at solving the deadlock. It essentially boils down to how often the deadlock is actually resolved. In the first part of the study we will test and analyze three different deadlock resolution

strategies, after which we will attempt to design a more optimized strategy based on our analysis of the previous three.

1.2.2 Time and Space parameters

We compare 3 different strategies for coordination and evaluate them from the following two aspects, to better understand their usefulness and applications.

- Time Efficiency - defined by how long it takes the robots to resolve the deadlock. Completion time is measured from the moment the robots start moving until both have reach the ends of the corridor. This value is directly dependent on deadlock resolution time.
- Space Efficiency - defined by how much horizontal space the robots need to resolve the deadlock. Space usage is measured as horizontal space occupied by the robots during the deadlock resolution.

In a real world corridor, the robots will not be aimlessly going up and down empty corridors, instead the corridor will be used by humans or other robots, and our robots will likely be busy completing some task. In these cases deadlock resolution is critical, and efficient resolution is incredibly important. Minimizing time becomes of value since the faster the deadlock is resolved, the faster the robots can get back to completing their tasks, and thus their work will be more efficient. Minimizing space usage is also relevant, since a corridor in real life is not always empty and fully available to move in. Using too much space could be very problematic since it could either lead to a collision or block the way for both humans and other robots with their own tasks to carry out. Avoiding "traffic jams" is also dependent on time. The longer time a deadlock resolution takes, the higher the risk of others arriving and potentially being blocked.

1.2.3 Deadlock resolution strategies

In this study, the robots always begin on a course forward from opposite ends of a corridor. When they meet, one of the following coordination strategies is applied. The strategies will be further explained in Section 3.3, where the exact algorithms are also described.

- Right-Way Strategy - This strategy is the traffic rule of keeping to the right. This is the baseline strategy.
- Impatience Strategy - This strategy assigns a randomized waiting time to each robot. Upon meeting, both stop. After the shortest waiting time steps the associated robot steps aside, before both robots keep moving straight forward.
- Give Way Strategy - This strategy assigns a priority to one of the robots. The robot with the lower priority steps aside, giving way for the higher-priority robot. Both then continue their route forward.

1.3 Delimitations

In this study we define the "deadlock scenario" as follows: Two robots going forward in a corridor meet face-to-face, blocking the way for each other, thereby both becoming suspended indefinitely since neither is able to continue moving forward. The corridor is empty except for the two robots, meaning that there are no other agents.

The robots are fully decentralized, meaning they have no knowledge of each other beyond that which they communicate or observe. The robots are not centralized and will have no control panel that can tell them what to do, so if they want to share information they must share it manually.

This study will not have any obstacles in the robots' way, meaning the corridor will be fully empty except for the two robots. The robots will also not take proactive actions, and will not observe each other until they meet in a deadlock. This means that the robots' cannot simply see each other from far away and solve a deadlock before actually meeting.

2 Background

2.1 Decentralized vs. centralized systems

In Section 1.1: Problem Statement, we described the concepts of decentralized and centralized robot systems. Regarding coordination, these two each have their different strengths and weaknesses. For the purpose of efficient coordination and deadlock resolution, it may seem natural to choose a centralized system - if the robots automatically "share a brain" the route planning seems like it should go smoothly. So why choose another approach? . In many cases, decentralized is a more realistic approach to putting these systems to real life use. Decentralized robot systems are more flexible for changes, and more easily allow for a large number of robots of varying models.[3][4]

2.2 Decentralized robot coordination

As many studies have shown before, decentralized robot coordination and deadlock resolution can be solved in many ways with many different algorithms. Depending on situation, these methods bring different advantages and disadvantages.

For this small literature study, we compiled some studies on, and adjacent to, decentralized robot coordination and deadlock resolution. We consider the most relevant work, at the intersection of the three following areas:

- The study handles robot coordination in a decentralized manner
- The study includes some sort of meeting conflict/ deadlock resolution
- The study handles includes a closed area, preferably a corridor or passage

(Note that these guidelines were not very strict, and a study must not adhere to all of them perfectly to be relevant)

Coordination of multiple robots requires handling deadlocks, where two or more robots simultaneously block each other from performing their tasks, and collisions, where the robots run into each other. From originally being studied in other computer systems, deadlock handling in robot coordination has been examined by several researchers. Xing et al. [5] describe three strategies: Deadlock detection and resolution, deadlock prevention, and deadlock avoidance. In the first strategy, deadlocks are identified and handled after forming, in deadlock prevention, possible deadlocks are identified and prevented before forming, and in deadlock avoidance, the system is designed to avoid deadlocks altogether. Through experiments, the authors explore their own different strategies, and conclude that avoiding impending deadlock is the most efficient approach. In this study, the area is modeled using a graph of nodes, where a control center must authorize each path a robot wants to take, and keeps track of which nodes are covered by which robots. The control center identifies problem areas with possible deadlocks, and is thereby able to avoid these areas.

With more decentralized robots however, which do not have a control center overseeing the entire system, avoidance becomes more difficult. Deadlock handling requires other strategies. Several studies handle decentralized robots in narrow spaces. S. Szominski et al [6] use the behavior of moving people to model algorithms for mobile robot coordination in a range of environments. One of their algorithms applies "give way to the right" traffic rules, while another has the robots assign "respect factors" to each other, which determine who has priority over another. In a dissertation by M. Cáp [7], the "corridor swap scenario" is specifically mentioned. As in the scenario we will be exploring later, this refers to a meeting of two robots in a narrow corridor, blocking each other's paths and causing a deadlock. To resolve this situation, the authors propose what they call the "kPM" strategy, where the robots iteratively plan new paths until an acceptable one is found. Thomas and Vaughan [8] study doorway negotiation between two robots. The method used models socially acceptable human behavior: When meeting in a doorway, both robots take a step back and wait. Each of them have an individual level of "assertiveness", which determines their amount of waiting time. When its waiting time is up, the robot makes a new attempt at moving forward. If a robot moves while the other is waiting, the waiting robot politely stays still and lets the more assertive one through.

As Szominski, Turek, and Byrski [6] point out, no strategy is useful in every situation, which is why a robot ideally should be able to recognise the situation it is in and choose a method to use accordingly, something they as well as Changyun Wei Koen and Hindriks [9] suggest methods for. In Changyun Wei Koen and Hindriks study, decentralized robots make real-time responses to several dynamic environments, and can resolve a set of deadlock situations using what they call "altruistic" coordination: "The robot is willing to make concessions in congested situations, even if it may result as a disadvantage to

itself." The robots can employ a range of strategies, such as waiting, moving-forwards, dodging, retreating and turning-head, to make local adjustments. The study differs from the corridor situation however, in that it is very much based on a grid-like environment. Its methods mostly require an environment with intersections to be applicable.

2.3 The challenge of communication

Besides deciding on a coordination strategy, following through with a strategy often also comes with the challenges of communication. Communication can be implicit (the robots gather information through observation) and/or explicit (the robots directly exchange information). Communication could be done through for example sound or light signalling, or as in [10], through Bluetooth connection. The more sophisticated communication required, the more complex the situation becomes, and explicit communication brings extra challenges. The robots require the necessary hardware functionality such as e.g. sensors for "seeing", or ways to transmit and receive online signals. There could be challenges like the limitations of bandwidth, delay in communication - either due to slow online connection, or the delay when a robot has to listen or observe and process what another tries to signal - or the way online communication might only be able to support a limited number of robots.[11]

Several researchers discuss these challenges, and some suggest solutions. The researchers of [12] propose an algorithm for better communication despite limited bandwith, while [13] discusses the advantages and disadvantages of limiting the number of other robots each individual communicates with in a large system.

As can be discerned from the number and nature of studies however, the solutions are often quite situation constrained. To implement a new robot system with extensive communication needs it is necessary to plan a tailored solution more or less unique to the situation.

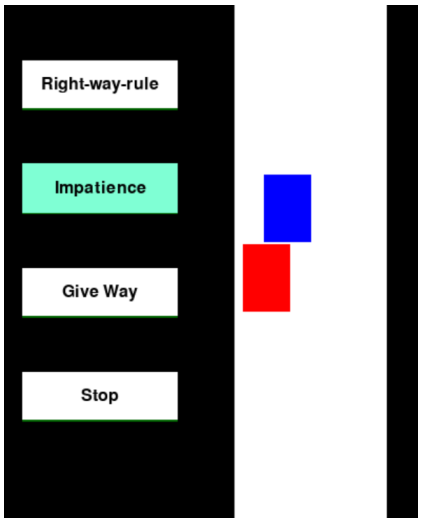
3 Method

3.1 The Simulation

The studied situation is the corridor scenario described in Section 1.2: Aim and Research Question, where two robots try using different deadlock resolution strategies that the study can compare and contrast. This requires testing, and to perform the tests we set up a simple simulation environment in Python. An example of what the simulation looks like is illustrated in Fig. 2a below. The simulation was created using pygame, an open source library for developing games and similar applications in Python.[14] In the simulation the corridor is viewed from above, with the two robots represented by red and blue rectangles that for the purpose of simplicity will be referred to as RED and BLUE. The user can run the simulation with the strategy of their choice by pressing on buttons

with the strategies' names on it. The robots will then, starting from opposite ends of the corridor, move towards each other, stopping when they meet and apply the chosen strategy. The simulation screen uses Cartesian coordinates, with (0,0) being the top left corner, measured with pixels.

We defined five starting positions for the robots, x-wise. Y-wise, the robots always start at the far opposite ends of the corridor. The robots' combined x-wise starting positions will hereafter be referred to in the form of (i,j), with RED in position i and BLUE in position j. That is, position (1,2) for example does not refer to an (x,y) coordinate, and should instead be interpreted as BLUE standing in the x-wise position No. All positionings can be seen in Fig. 2b.



(a) A simulated meeting with the robots standing in position (1,2)

(b) All possible x-axis starting positions. Y-axis starting positions are always bottom (RED) and top(BLUE)

Figure 2: The simulation setup

The positions (1,4), (1,5) and (2,5) allow the robots to freely pass each other (see Fig. 3 below), avoiding deadlock altogether, and these were therefore excluded from this study.

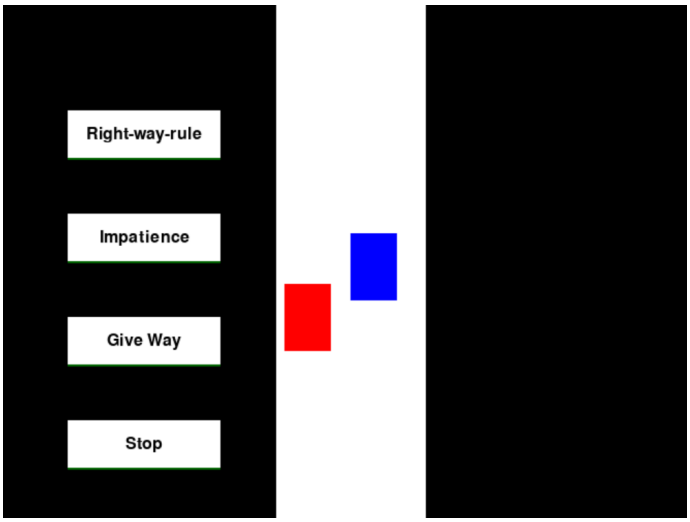


Figure 3: In position (1,4) there is no meeting and thus deadlock never occurs

3.2 Testing Variable Constants

The simulated robots were set to a height and width of 65 pixels and 45 pixels respectively, but with an invisible hit-box around it that is 5 pixels bigger on both sides (70x50 pixels). In the simulation, the deadlock is only triggered once the two robots actually collide, which would be very unfortunate if it were to happen in real life. To counter this we used the hit-boxes, and since they are invisible and bigger than the robots they simulate the way a sensor would work: detecting the other robot before the actual robots collide. When the two hit-boxes collide in the simulation the simulated robots become aware of each other's presence and it mimics how robots would use sensors to find each other and other obstacles in real life.

The corridor was set to a width of 205 pixels, which is 5 pixels less than three times the hit-box width, meaning it cannot fit three robots in a row. The corridor length was 650 pixels.

3.3 Strategy description

The following sections describe our developed and chosen deadlock resolution strategies. It is worth noting that the strategies are all applied in their raw form, meaning that no optimizations were included and that the algorithms for the strategies have some limitations. The limitations and their potential optimizations are expanded upon in Chapters 4: Results and 5: Discussion.

3.3.1 Right-Way Strategy

This strategy uses the universally accepted *Right-Way traffic rule*, which here functions as a baseline deadlock resolution strategy. The rule is that if your right is open, you will go to the right. In the implementation, when the two robots are coming towards one another and register each other, both will move to the right if their own right is open. This strategy was chosen because it is a very simple rule that is easy to implement and widely used in many contexts. It is important to note that the Right-Way Strategy has a 100% success rate, meaning that it *always* resolves the deadlock.

Algorithm 1 Right-Way Algorithm

while In Deadlock **do**
    robots move to their respective right
end while
robot and *other_robot* keep moving towards end destination

Limitations of the algorithm
The most noteworthy limitation for Right-Way is also the reason it is our baseline strategy: it is impossible to optimize Right-Way. The strategy is too simple and already 100% effective, so there is no way to optimize it without changing it into something else. The biggest limitation is that there is no potential for improvement.

3.3.2 Impatience Strategy

The second strategy, which we have chosen to name the *Impatience Strategy*, is a variant of the "assertive" method presented in [8]. Similar to the robots in that study, our robots will react to a meeting by stopping and waiting for a randomized amount of time. When the first robot's waiting time is over, it will step aside. After that both robots move forward again. The waiting times are randomly selected between 500 and 1200 milliseconds. This interval is not set in stone: we could have chosen much shorter waiting times. The reason we chose this particular interval was to mimic how real life robots might be designed. Real robots would have to sense the movement using sensors, and that process is slower than it would be in a simulation, which means that if the waiting time is too short, both robots might move to the same side at once, which would lead right back to deadlock. This simulation still uses shorter waiting times than what we assume real robots would require.

To make the strategy name reflect our situation better, we have renamed it, and refer to our robots as being "impatient" rather that "assertive". The strategy in the study proved successful, and with our modification it seems naturally applicable for our study. With randomized waiting times, it will also allow us to explore the benefits of a randomized strategy.

Algorithm 2 Impatience Algorithm

```
while In Deadlock do
    if time_passed > robot_waiting_time then ▷ This robot's waiting time is up
        if other_robot not moving then   ▷ Pick side with minimal movement
            if Meeting on right side of screen  then
                Move left          ▷ Moving left requires less sideways movement
            else
                Move right      ▷ Moving right requires less sideways movement
            end if
        end if
    end if
end while
robot and other_robot keep moving towards end destination
```

Limitations of the algorithm

The raw algorithm for the Impatience Strategy has a limitation in our study, which is related to the size of the corridor. The limitation is that if both robots meet in the middle, meaning start at position 3, they will not be able to resolve the resulting deadlock at all. This is the result of the chosen corridor width, which is not wide enough for 3 robots side by side, meaning that if the robots are in the middle then none can pass without both moving. If only one of the robots is in the middle, the deadlock has a 50% chance of getting resolved depending entirely on which robot becomes impatient first. If the robot in the middle becomes impatient, the deadlock gets resolved and if not then they become stuck. This can be fixed in multiple ways: using algorithmic optimizations or even pure physical setup of the environment. These solutions are discussed in Chapter 5: Discussion.

3.3.3 Give Way Strategy

The *Give Way Strategy* is priority based, meaning that the two robots have a predetermined priority and have to take this into account when the deadlock is being resolved. This is to mimic real life situations where some robots might have more important tasks than others and therefore warrant for a higher priority. In this strategy, the robot with the lower priority must "give way" to the higher priority robot, meaning it steps aside so that both robots can continue moving forward. The lower priority robot will first attempt to step left, however if it is too close to the wall it will instead attempt to go right.

Algorithm 3 Give Way Algorithm

```
while In Deadlock do
    if this_priority > other_robot_priority then      ▷ This robot has priority
        Do not move
    else                                  ▷ Other robot has priority. This robot moves
        if robot can go left then
            Go left
        else if robot can go right then
            Go right
        else
            Do not move
        end if
    end if
end while
robot and other_robot keep moving towards end destination
```

Limitations of the algorithm
Similar to the Impatience Strategy, the algorithm of the Give Way Strategy cannot resolve the deadlock when the higher priority robot is in the middle of the corridor since the corridor is not wide enough for the lower priority robot to pass on any of its sides. However, if the lower priority robot is in the middle the deadlock is easily resolved. There are many potential solutions to this limitation, further discussed in Chapter 5: Discussion.

3.4 Evaluation methods

In the simulation, the time parameter will be measured by clocking a simulation run from start to finish. Space usage will be measured as the amount of movement taken on the x-axis during deadlock resolution. (Note that the RED robot moves up from the bottom of the screen, and the BLUE robot moves down.)

We define a simulation run as the period from the moment the robots start moving from their respective ends of the corridor, until the RED robot has reached the opposite end. This time is called *completion time* in the text. Choosing between BLUE's time and RED's time is arbitrary since the target is the difference between nominal value and completion, as that denotes time taken for actual deadlock resolution, which will be referred to as *resolution time*. There are small differences between RED's and BLUE's time values - a few milliseconds - but these were determined to be insignificant enough to be overlooked. This difference is the result of the robot's starting positions on the y-axis. The robots do not start with the exact same distance from the corridor ends, which causes the difference in time. Since we focus on approximates a minor difference between 605 and 598 ms is irrelevant.

4 Results

The study compares three deadlock resolution strategies, with regard to the time and space required to resolve a deadlock between two robots meeting in a corridor. In this section we present the results of the measured parameters time and space usage, both strategy for strategy and in side-by-side comparison.

When comparing our strategies we used nominal values for time and space. These were determined by letting the robots in the simulation move all the way through the corridor with all the same prerequisites as the tests, but in a position where no deadlock would be triggered. The closer our experiment values are to the nominal values, the better they are deemed to be. The nominal values are listed in Table 1 below.

NOMINAL VALUES	
TIME(ms)	3095
SPACE(pixels)	0

Table 1: Nominal values

The results for each strategy are presented in charts showing time and space usage in each simulated position. To compare the strategies, we present a comparison table in Section 4.5 with the minimum, maximum and average amount of time and space required amidst all positions. This table will be vital for comparing comparing the different strategies in Chapter 5: Discussion.

4.1 Important notes

The measured time variable will always depend on x-axis-movement. Since the robots never move on the y-axis and x-axis at the same time, the longer they move sideways in the deadlock resolution, the longer they are stalled in their movement forward and the more time it will take them to reach the other end of the corridor.

Another noteworthy reminder is that not all strategies have a a 100% success rate for solving the deadlock. The Impatience and Give Way strategies both have situations where they cannot resolve the deadlock.

It is also important to note that not all starting position combinations will be tested, since not all of them trigger a deadlock. For example (1,5), when RED starts in position 1 and BLUE starts in position 5, and (5,1) are not tested as these do not lead to the robots meeting.

4.2 Right-Way Strategy results

We can see that both the completion time (Fig.4) and x-axis movement (Fig.5) diagrams exhibit a similar pattern: Whenever the robots start out in positions

that put them to each others' left side, both the time and space usage drastically decrease compared to when the robots are on each others' right side.

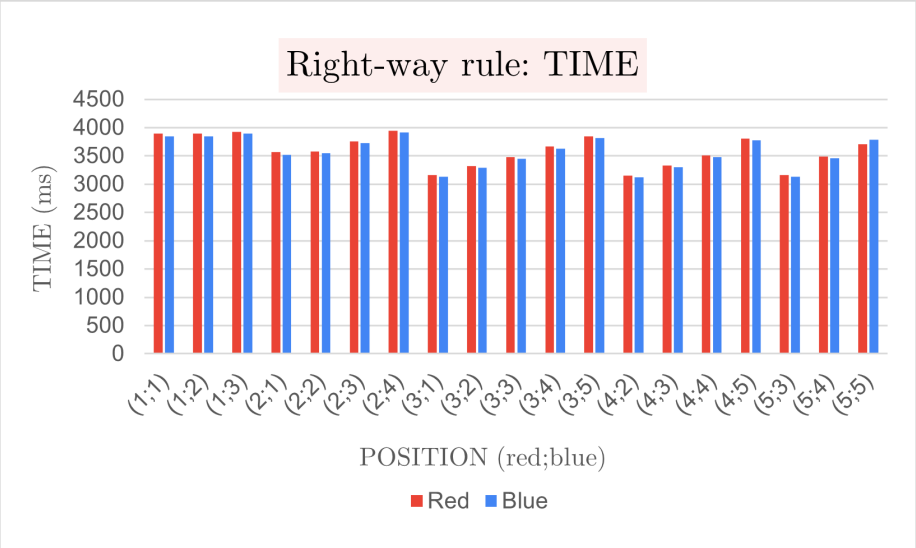


Figure 4: Time to reach end destination with Right-Way strategy. Non-meeting positions excluded.

The x-direction movement taken (see Fig.5) is directly related to positioning in relation to each other and the walls. In certain positions, when BLUE is on RED's right, the space usage is quite high. In most positions, each robot's movement is roughly the same as the other's, but observe how both have to move more when starting next their own right-side wall.

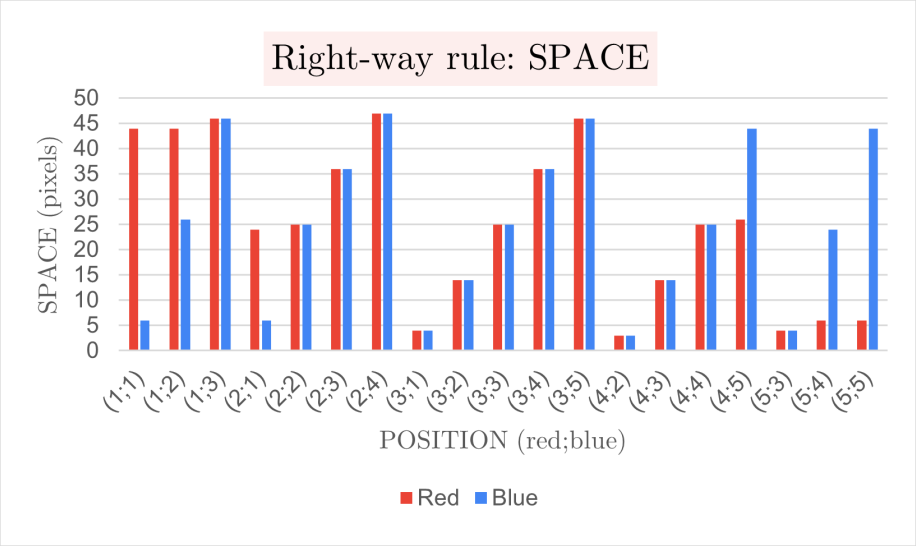


Figure 5: Movement taken in X-direction with Right-Way Strategy. Non-meeting positions excluded.

4.3 Impatience Strategy Results

The Impatience Strategy is a randomized strategy, where both robots wait for the more "impatient" one to step aside (see Section 3.3.2: Impatience Strategy). The less impatient robot never moves sideways, which means that only one robot will move on the x-axis on every run, and that every position has two possible solutions: one where RED is impatient, and one where BLUE is impatient. To illustrate this, we present two space diagrams, one for the case where RED moves (Fig.8) on the x-axis and one where BLUE moves (Fig.9).

As explained in Section 3.3.2, the Impatience Strategy does not always resolve a deadlock. If both robots start in position 3, the deadlock can never be resolved, regardless of which robot starts moving, since neither will be able to pass between the other and the wall. When only one robot starts in position 3, the robots will resolve the deadlock when the impatient robot is the one in position 3 and can move away, which occurs about half of the time. This is illustrated in Fig.7 by having the corresponding bars in a lighter colour in places with one robot in position 3. Their value is the time for deadlock resolution when the deadlock actually gets resolved (robot in position 3 becomes impatient). The situation which inevitably causes deadlock is illustrated in both Fig.7, Fig.8 and Fig.9 by orange bars marked "DEADLOCK".

As illustrated in Fig.7, the Impatience Strategy generally requires much time compared to the others, often averaging around and above 4000 milliseconds. This sacrifice of time is expected since we not only have to wait during the

deadlock resolution, but also during the robots' waiting time (between 500 and 1200 ms), while they determine which one is more impatient. How much time the deadlock resolution takes also heavily depends on which robot becomes impatient, since some positions require the impatient robot to move much more to pass than the other robot. This situation is illustrated by the following example: BLUE is in position 1 and RED is in position 2 and BLUE becomes impatient. This means that BLUE has to move to the middle of the corridor to pass RED, which requires more movement and more time than if BLUE were to pass by BLUE.

(a) Robots in position (1,2) before BLUE becomes impatient

(b) After BLUE has become impatient and moved to the right side

Figure 6: Resolution in position (1,2) with BLUE impatient

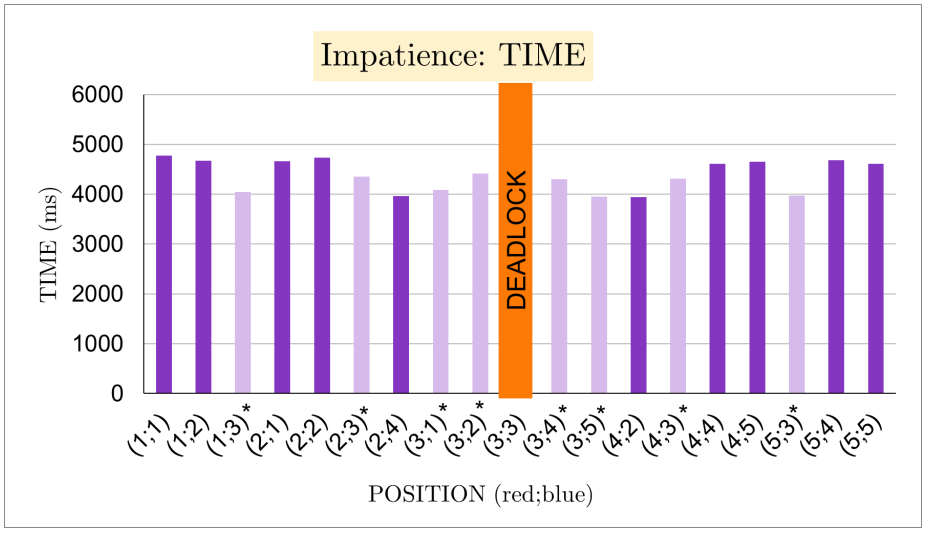


Figure 7: Time to reach end destination with Impatience Strategy.
Non-meeting positions excluded.
*When one robot is in position 3, deadlock is only resolved 50% of the time

Similar to time, it is evident that the amount of space used depends heavily on which robot becomes impatient. When comparing Fig.8 and Fig.9, we can clearly see that some of the positions resolve deadlock using much less or more space depending on which robot becomes impatient. In all places where both start in the same position (e.g. (2,2)), the space usage is identical in both diagrams, but in most places where starting positions differ, so does the space value. For example, if we look at the situation where RED starts at position 2 and BLUE at position 1, we can see that when RED becomes impatient (Fig.8) the space used is 30 pixels, but when BLUE becomes impatient (Fig.9) the space used is 70 pixels, which is a notable downgrade. It is very important to bear in mind that Fig.8 and Fig.9 do not happen at the same time. They each illustrate two different solutions to the same problem, depending on which robot became impatient.

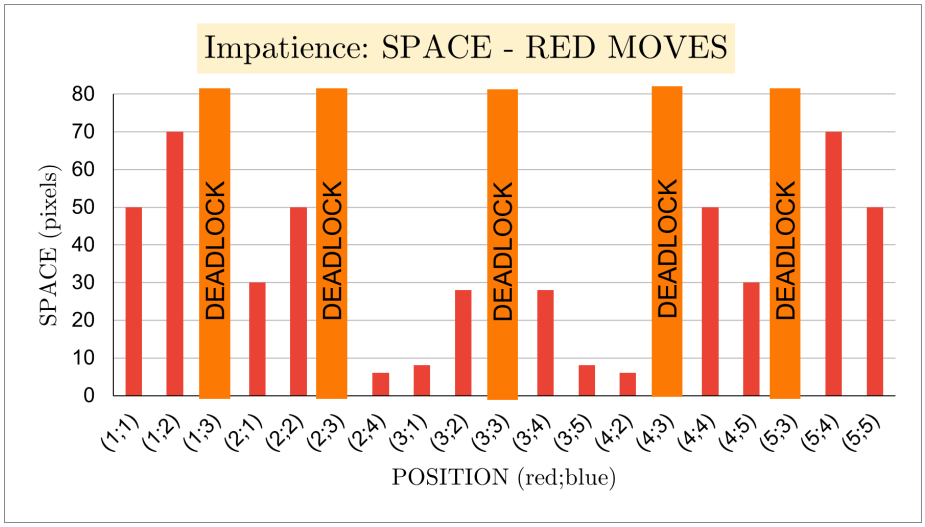


Figure 8: RED becoming impatient and moving in X-direction with Impatience strategy. Non-meeting positions excluded.

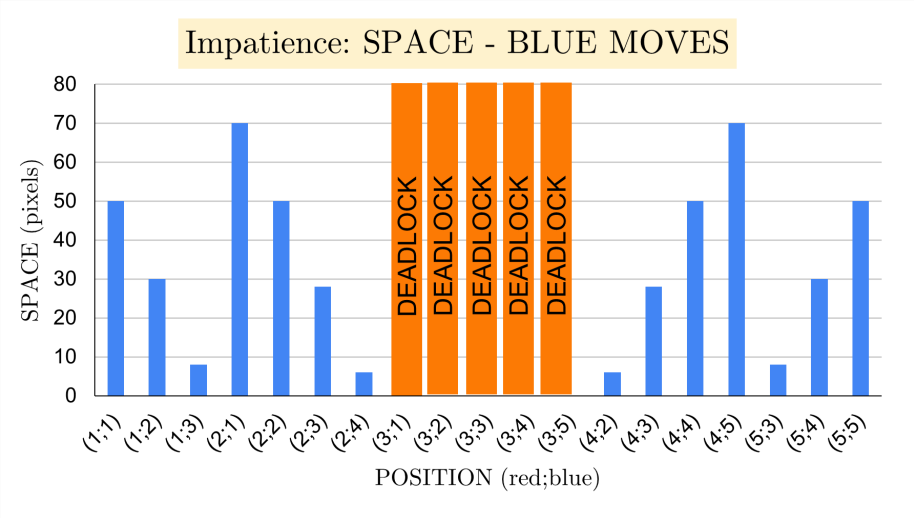


Figure 9: BLUE becoming impatient and moving in X-direction with Impatience strategy. Non-meeting positions excluded.

4.4 Give Way Strategy results

The Give Way Strategy is a priority based strategy, meaning that one of the robots is deemed "more important" than the other. In our case RED *always* has low priority and BLUE high priority. The reason we do not test both with high priority is because the chosen positions are symmetrical and not randomized. The symmetrical nature of the positions would lead to RED and BLUE having the same space usage and similar time usage, but in opposite positions. This essentially means that if you take Fig.11, which is when BLUE has the high priority, and swap the robots' positions you would get the graph for when RED has the high priority. The same principal applies to the time for Give Way, Fig.10.

The low priority robot is the one moving on around the high priority robot, meaning that in our case only RED will move on the x-axis. As we can see in Fig.10 and Fig.11 the Give Way algorithm never resolves the deadlock if the high priority robot is in position 3, since RED cannot pass around it.

Whenever the lower priority robot is closer to a wall than the higher priority robot, both time and space get higher values. Vice versa, if the higher priority robot is closer to the wall, we get lower values. We can see this visually represented in both Fig.10 and Fig.11.

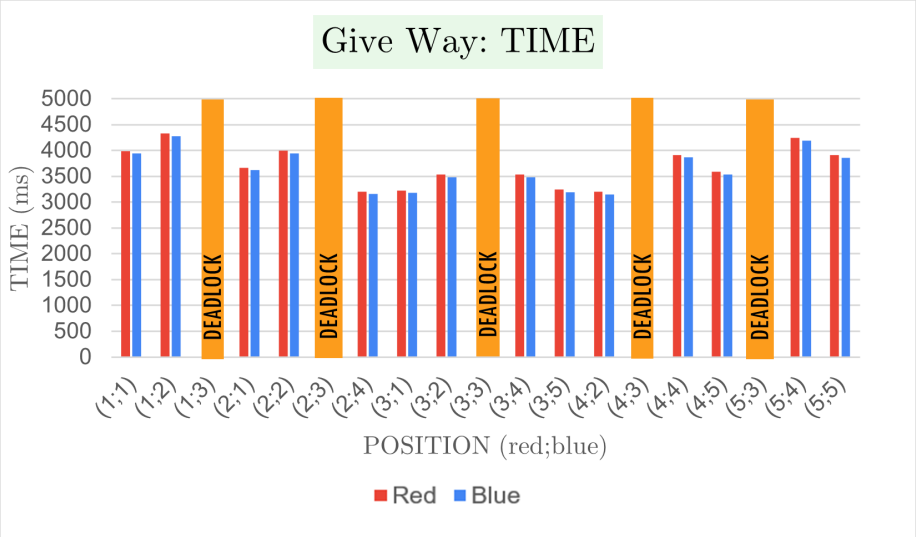


Figure 10: Time to reach end destination with Give Way Strategy. Deadlock occurs when BLUE, who has higher priority, is in position 3. Non-meeting positions excluded.

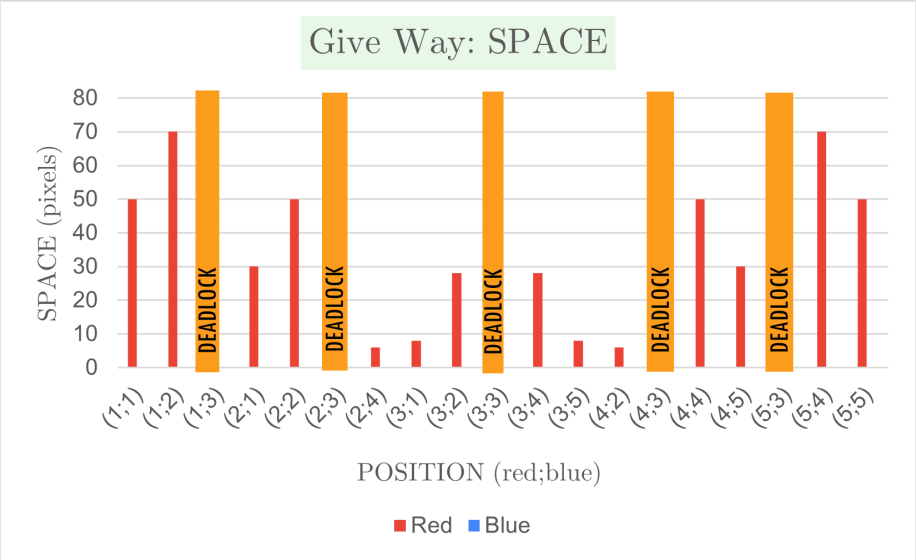


Figure 11: Movement taken in X-direction with Give Way Strategy. BLUE has higher priority. Non-meeting positions excluded.

4.5 Comparisons

Fig. 12, Fig. 13 and Table 2 below summarize all the average, minimum and maximum values for time and space from our different strategies and present the side by side for comparison.

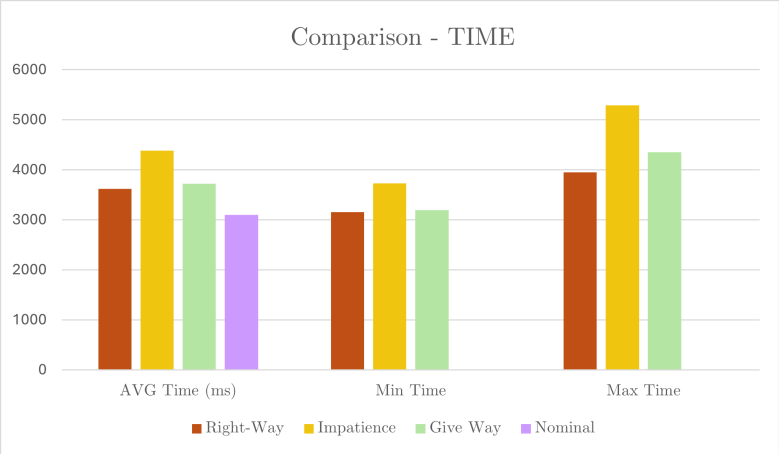


Figure 12: Average, minimum and maximum time usage.

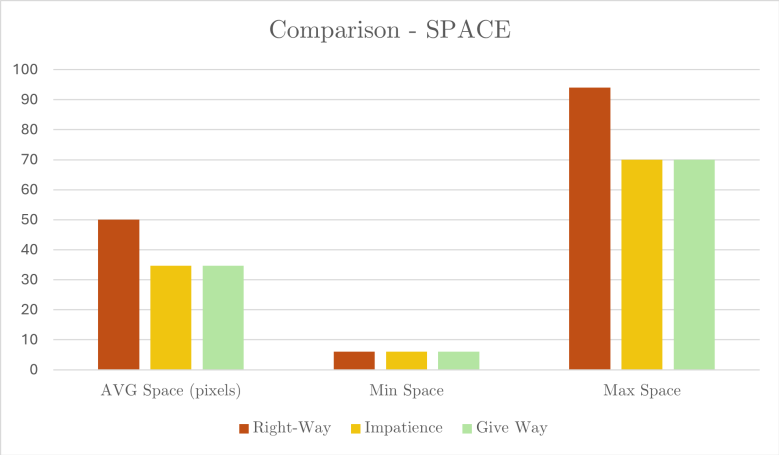


Figure 13: Average, minimum and maximum space usage.

VALUES	RIGHT-WAY	IMPATIENCE	GIVE WAY	NOMINAL
AVG Time(ms)	3621	4379	3721	3095
Min Time(ms)	3149	3731	3190	-
Max Time(ms)	3949	5290	4348	-
AVG Space(pixels)	50	34.57	34.57	0
Min Space(pixels)	6	6	6	-
Max Space(pixels)	94	70	70	-

Table 2: Average, minimum and maximum time/space usage of all the strategies.

When looking at at the results of the Right-Way Strategy, we can see that the average value of completion time is approximately 3600 ms compared to the nominal value of roughly 3100, which means that on average the deadlock is resolved in around 500 ms. The average x-movement is 50 pixels, which is same as the width of one robot (including hit-box, see Chapter 3: Method), and since both RED and BLUE move, we can safely say that on average both robots move half a robot to the side. As mentioned in Section 4.2, the amount of space and consequently time the deadlock resolution needs in the Right-Way Strategy is mostly dependant on the starting positions of the robots - both in relation to each other and to the walls. When the robots are colliding on the each other's left edge, we receive the minimum experiment values. Since the robots move to the right, it takes less time for a robot to move away when already on the other's left. The opposite is true for the maximum values: the robots collide on each other's right, and therefore require much more movement in order to pass each other, which also leads to more time needed.

Right-Way is also the only strategy to have no failures when it comes to actual deadlock resolution. Deadlock was resolved in all possible positions.

For the Impatience Strategy, the average time is approximately 4400 ms, which when compared to the nominal value of 3100 ms shows that the deadlock resolution took about 1300 ms on average. We can also note a quite large difference between the minimum and maximum values, approximately 1600 ms.

When looking at the space values, we can see that the average or roughly 35 pixels and minimum of 6 pixels are both very much lower than the maximum of 70 pixels. The higher maximum comes from when on of the robots is closer to a wall and the other closer to the middle, and the one closer to a wall becomes impatient and has to move all the way to the middle to get past, which only happens about half the time. For example, if RED is in position 1 and BLUE in position 2 and RED becomes impatient, RED has to move all the way to position 3 to get past BLUE, which requires 70 pixels of sideways movement.

When comparing to the nominal completion time of around 3100 ms to Give Way's average time of 3700, we can see that the deadlock resolution happens in about 600 ms. As we can see in Table 2, the minimum and maximum values are quite far from each other and the average value. The average space value is

approximately 35 pixels.

Both space and time values are highly linked to the starting positions of the robots, since how much RED has to move around BLUE to resolve the deadlock determines both how much time and space the resolution needs.

It is also worth remembering that the Impatience Strategy and the Give Way Strategy do not always resolve the deadlock. In Impatience, whenever at least one robot is in position 3 and the other robot becomes impatient, we get an unsolvable deadlock. Give Way cannot resolve the deadlock scenario if BLUE, which has higher priority, is in position 3. These two are not effective in every position, and need optimizing to function: They need an alternative solution for these positions, or they will be stuck indefinitely. This is critical to be aware of if they are to be implemented in reality.

5 Analysis and Discussion

To answer the first part of our research question, "How do different deadlock resolution methods compare in terms of time and space efficiency, in meetings between two decentralized robots in a corridor?", we must compare our the strengths and weaknesses of our strategies.

The strategies are compared on how well they minimize the parameters of space and time, which in reality means comparing their values to our nominal values in Table 1. Minimizing the usage of space and time is very important so that the deadlock does not block the corridor for too long and the robots don't cause a full stop of all traffic in the corridor, as was described in Section 1.2.2: Time and Space parameters. It is also important to compare the effectiveness of the strategies, meaning how often they actually solve the deadlock scenario. If a strategy is efficient time- and space-wise, but very often leave the robots stuck in deadlock even after applying the algorithm, the strategy is not very useful in real life.

5.1 Comparisons

Let us start by summarizing our different strategies to fresh up the memory:

- Right-Way Strategy: both robots meet in a deadlock and move to their respective right.
- Impatience Strategy: both robots meet in a deadlock and start waiting. The more impatient robot, the one with less waiting time, moves sideways. This strategy is randomized, as the robots' waiting times are randomized.
- Give Way Strategy: both robots meet in a deadlock. The lower priority robot moves sideways. This is NOT a randomized strategy: in our case RED always has a lower priority than BLUE.

Looking at the time parameter in the comparison tables, we can see that both Right-Way and Give Way outperform the Impatience Strategy, both regarding average and minimum/maximum values. Impatience has an average of 1300 ms above the nominal value, almost double the others' time. This behaviour is expected, since Impatience sacrifices time in the waiting process. The way the algorithm works means that the robots will stand still until the shorter waiting time has passed - inevitably this means a delay of at least slightly over 500 milliseconds. Between Give Way and Right-Way, Right-Way is the most time efficient, but only marginally. The average value differs in only 100 milliseconds, compared to almost the almost 760 ms difference between Right-Way and Impatience.

In comparing space usage, all strategies have the same minimum value: 6 pixels. Give Way and Impatience have the same max (70 pixels) and average (34.57 pixels) values , while Right-Way has higher values (94 and 50 pixels). In the identical values of Give Way and Impatience, we can see signs of their similar behavior. In fact, if we compare the space diagram of Give Way (Fig. 11) with the one for RED in Impatience (Fig. 8), they are identical as well. After RED becomes impatient and therefore moves, the robots' behavior is equivalent to when BLUE has priority and RED gives way. The space usage with BLUE impatient is the same on maximum, minimum and average as for RED and Impatience in total, but the values are differently distributed between the positions (see Fig. 8 and 9). This indicates that if Give Way was to be done with RED having priority, its space diagram would match that of impatient BLUE.

Looking at both parameters, it becomes clear that Right-Way sacrifices space for time, while Impatience sacrifices time for space. In the space aspect, Give Way's results are on average and maximum much better than Right-Way but identical to Impatience. Give Way's time values outperform Impatience, and though Right-Ways beats its time, it is not by much. Thus, Give Way appears the most efficient of the three strategies.

An important aspect other than time and space usage is that Impatience and Give Way do not always solve the deadlock, which Right-Way however does. Right-Way is thus the most effective. Out of the other two, Impatience is the more effective. Due to the randomization of whether RED or BLUE moves, the strategy has a 50 % chance of succeeding in resolving the deadlock in most of the problematic positions (see the lighter colored bars in Fig.7). Only in (3,3) is the deadlock completely inevitable. Because of the priority approach, Give Way does not have this possibility of success. In Give Way, there are five positions (when BLUE is in position 3) which never resolve deadlock. It is thus the least effective.

5.2 Advantages and Disadvantages

Regarding real life application of the strategies, they each have their strengths and weaknesses that makes them useful for different situations.

5.2.1 Right-Way

As seen above, the Right-Way Strategy's weakness is space usage, an hence it is useful in situations where this parameter is negligible, and only time needs minimizing. This could for example be in a wide and/or quite empty corridor, where risk of collisions is low. Another one of the advantages is that the Right-Way Strategy does not require any explicit communication such as information sharing between the robots to resolve the deadlock, they all simply move to the right, and all the information they need (the other robot's position compared to ones own) is implicitly gained using sensors. Communication between decentralized robots comes with its own set of complications, as talked about in Chapter 2: Background. Needing to share information and communicate with other decentralized robots is very difficult and time-consuming to implement, not to mention that if the communication function breaks it might make deadlock resolution impossible. How robots share information and communicate can also vary based on model and type of robot, which can be very problematic if two different types of robots are in a deadlock and do not understand each other. It is therefore a huge advantage that robots do not need to communicate at all to use the Right-Way Strategy.

However, the biggest advantage of this strategy is that it always resolves the deadlock, which leads into its biggest disadvantage: Right-Way is a completed strategy, meaning that you cannot optimize it. The lack of potential optimizations makes the Right-Way Strategy stagnant, it cannot become better than what it already is, which means that other strategies can easily surpass it. This is a result of the algorithm being too simple: it is very difficult to change "if you collide, go to the right" in a way that makes it better and keeps the actual strategy of the original. There are not many more ways to go right, than to just go right.

Right-Way does however have a very good use in its simplicity since it can be used as a default base strategy to resort to when another strategy is unable to resolve a deadlock. So even though Right-Way cannot get any better, it is still a very good choice if everything else fails.

5.2.2 Impatience

The Impatience Strategy, just like the Give Way Strategy, is much better at minimizing the space usage of the deadlock resolution than the Right-Way Strategy is, which is a very good quality when in a corridor since it prevents blockage. Just like the Right-Way Strategy, the Impatience Strategy also has the advantage of not needing any communication to resolve the deadlock. The robots again do not need to explicitly share any information, the more impatient robot simply steps aside first. This makes the strategy more applicable in real life - it can be used with different kinds of robots who cannot communicate with each other. The robots do not have to be able to communicate at all!

Another important advantage is the randomization aspect. In the Impatience

Strategy the two robots do not move at once, which means that they need to choose who will step aside. Randomization makes the strategy more adaptable and less prone to deadlock than it would be with set values. This has been discussed in Chapter 4: Results for both the Impatience and Give Way strategies.The advantage of randomization also becomes apparent if the number of robots increases. If we have multiple robots that have to step aside each other, we want them to all have different waiting times, and making sure that each set waiting time is unique requires much more effort than randomizing the waiting times.

The major downside to the Impatience Strategy is the delay caused by the waiting step. The reason why the other two strategies are more time efficient is that they do not waste time waiting for a decision, unlike Impatience. To minimize this issue, the waiting times would need to be lower. However, it might not be realistically possible to choose much lower waiting times, as we discuss in Section 5.3: Optimizations.

5.2.3 Give Way

On of the biggest advantages Give Way has over Right-Way and Impatience is that it minimizes both the space usage and completion time, which makes is the best of both worlds, making it the most efficient strategy we used. This is however, is only the case when the Give Way Strategy works. As mentioned before, the Give Way Strategy does not always resolve the deadlock. Compared to the other two strategies, Give Way is the most prone to become stuck in the deadlock, making it the least effective strategy overall.

An advantage of the Give Way strategy is in the priority aspect of its design. In real life, some robots may have reason for having a higher priority over others. In these cases, efficient deadlock solving is not as important as it is to find a solution that benefits the prioritized robot. For example, the robots could be of different sizes and it would be preferred to avoid steering the bigger one towards the middle, or one of the robots could be carrying a sensitive load that means the robot should avoid moving sideways.

However, one of the biggest downsides of Give Way is that the robots would need to explicitly communicate with each other. This does not come into play in the simulation, but in real life it would be a much more important issue. Since the robots are decentralized, they do not immediately know each other's priority and they would need some way to share this information with each other, as we established in Section 2.3. No matter how the communication is done, the deadlock resolution would end up taking much more time in real life than in simulation. Communication is also disadvantageous when more than one type of robot are stuck in a deadlock, since it requires that all robots have ways of communicating with each other, as was mentioned in Section 5.2.1: Right-Way.

The Give Way Strategy is also at a disadvantage when the number of robots increases. Unlike the Impatience Strategy, one cannot assign multiple robots

priorities randomly or only based on the importance of the task, since that might lead to multiple robots having the same priorities. Similar to the situation in Section 5.2.2 above, the robots cannot resolve the deadlock if they have the same priority, since you cannot determine which one should step aside. In the Give Way Strategy the robots are pre-assigned their priority, which means that if we want to avoid giving out the same priority, we must keep track of all of the robots, which is very time and effort consuming. It also makes the system a little more centralized, since whatever is giving the robots their priority must keep track of all of them.

5.3 Optimizations

In this study, the chosen strategies are tested in a very base form with no optimization. There are however some optimizations one could apply to make the strategies work better. One of these is how one positions the robots in the corridor. This optimizations is not algorithmic, but is very effective in its simplicity. We can see from the Section 4: Results that some starting positions give much better and worse results than other in different strategies. For example, the Right-Way Strategy performs much better when the robots collide on each other's left side than when they collide on the right. To optimize this one could simply make the robots drive along predetermined positions that guarantee that the robots will collide with their left sides. This style of optimization works for the Impatience and Give Way strategies as well. For example, by making sure that the robots don't start at position 3 (the middle), we can make sure that the deadlock scenario is always resolved when using Impatience or Give Way. Give Way can also be optimized using position by making sure that the higher priority robot is closer to the wall than the lower priority robot, which leads to less space and time being used by the low priority robot when passing by.

Another way to optimize the Impatience and Give Way strategies by letting both robots move. In the original algorithms, only one robot may move and try to pass by the other, and that sometimes causes the robots to get stuck and deadlock to remain unresolved. An example of this is both robots meeting in position 3. Let us use an example that can be applied to both Impatience and Give Way: the robot RED is trying to pass by BLUE, but BLUE is in position 3. Since BLUE is in the middle of the corridor, there is not enough space for RED to pass by, and RED will get stuck and stop. To solve this situation, we can let BLUE realise that RED is stuck since RED is still, and have BLUE move a little to the side to make room. This way the deadlock scenario is resolved and the robots continue on their merry way.

The Impatience Strategy can also be optimized further by introducing "urgency", which is a type of priority, to it. "Urgency" would tell the robot how important or "urgent" its task is, and the robot would then change the intervals within which it randomizes its waiting time to match. To illustrate with an example: the interval for randomizing waiting times in the simulation in this study is 500-1200 ms. With the "urgency" optimization, the robot would be

told how urgent its task is. If the task is "more urgent", the robot would decrease the roof of the interval, maybe to 500-800 ms, meaning it would get a lower waiting time on average, which most likely would make the robot more impatient when in the deadlock scenario. In the opposite case, if the robot is told that the task is "less/not urgent", then the robot would raise the bottom of the interval, maybe to 800-1200 ms, leading to the robot having longer waiting times and most likely less impatient in deadlock scenarios. This optimization would help the fact that Impatience generally is a very time consuming strategy. It does have the drawback of making it easier for robots to end up with the same waiting time, since the intervals become smaller, which is something to be aware of when implementing this optimization.

5.4 Future Studies

For future studies one could include static or dynamic obstacles in the corridor that the different strategies have to navigate as well as resolving the deadlock. This would be a good continuation since the situation would become more realistic to an actual workplace where there are people and objects everywhere. Especially the dynamic object would be a good model for people in corridors.

Another potential for future studies is studying more proactive methods, meaning that the robots see each other from further away and have make sure the deadlock never happens by moving in advance.

5.5 Social Impact

The robots that inspired our research (See Appendix A and Chapter 1: Introduction.) are a very representative example of a situation in which efficient and effective deadlock resolution is absolutely necessary. In such hospital environments, robots are often used for carrying supplies or instruments which could be vital for the functionality of the hospital services. It may then be crucial to not waste time, be stuck in deadlock or clog up a corridor with a traffic jam. We see a development in society where this is going to be a necessary problem to consider both in environments like these, and in everything from industries to homes. Therefore, our subject is vital and from a social sustainability perspective, because of its consequences for a stable functionality of these systems.

Our developed solutions could be a foundation for minimizing the issues we describe here. While Impatience and Give Way cannot be implemented in their raw form without causing occasional deadlock, we suggest ways to use them as a base for more effective strategies. We hope that our findings can inspire future research on the subject and real life testing of strategies to produce solid deadlock resolution strategies for a reliable and safe use of robots in society.

5.6 Summary

To be able to answer the second part of our research question, in which we design a new algorithm, we will summarize our analysis and see what we can use to design our improved strategy.

After analyzing the three chosen deadlock resolution strategies, we have seen a very important point: having only one robot move is bad for effectiveness, as Right-Way, the only strategy where both robots move, is the only strategy that always resolves the deadlock. We can therefore incur that the improved strategy should have both robots move, to maximize effectiveness. Another important point is that time efficiency is dependant on space efficiency, meaning the less space we use for the deadlock resolution, the less time it will take. This is especially obvious when looking at the Give Way and Right-Way Strategies. The more space the robots need to use to resolve the deadlock the longer it will take.

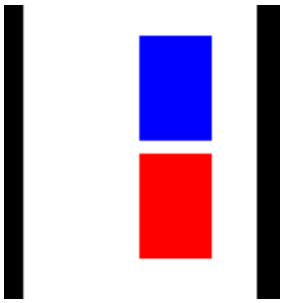
We can therefore conclude that our improved strategy should fulfill these two criteria: both robots should move and we should aim to minimize space usage. The new strategy is presented in its own Chapter.

6 Designing an improved strategy: Minimal Space

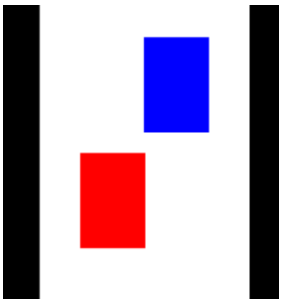
Named "the Minimal Space Strategy", this improved strategy focuses on minimizing the space usage further, and is subsequently the answer to the second part of our research question: "How could we design a more optimized strategy based on our analysis of the results from part 1.".

6.1 Strategy and Algorithm Description

The Minimal Space Strategy works as follows:
Two robots meet from opposite directions in a corridor, entering deadlock. We assume that the robots have cameras or sensors so that they can detect which areas of themselves would be close to the other robot, meaning which parts of the other robot is blocking them from moving forward. To resolve the deadlock, the robots move in the direction which minimizes the additional blocking area on the robots. For example, if the robots are blocking each other in their right corners, then both of them would move to the left, as illustrated in Fig.14b. If they instead moved to the left then the blocking area would increase: the robots would block not only on the left corner but also the middle, which is more than before. In the case of a head-on meeting the robots simply both go to the right. It does not make a difference which side the robots go to in a head-on meeting since both sides lead to the same amount of area being unblocked, see Fig.14a.



(a) Robots meeting head-on



(b) Robots meeting at right corner - minimal space needed to go left

Algorithm 4 Minimal Space Algorithm

while In Deadlock **do**
  if *other_robot* is on *robot*'s Right **then**
    Go left
  else if *other_robot* is on *robot*'s Left **then**
    Go right
  else if *other_robot* and *robot* meet Head On **then**
    Go right
  end if
end while
robot and *other_robot* keep moving towards end destination

6.2 Space and Time Results

In both the time and space graphs, Fig15 and Fig16 respectively, we can see a similar pattern. The highest values in both graphs are found in positions where the robots meet head-on, meaning that both RED and BLUE start in the same positions. Note that the total values of space in these positions is always the same, being the total of 50. The graph (Fig.**??**) shows both the robots movements, and their total movement in every position is the sum of both robots' space values.

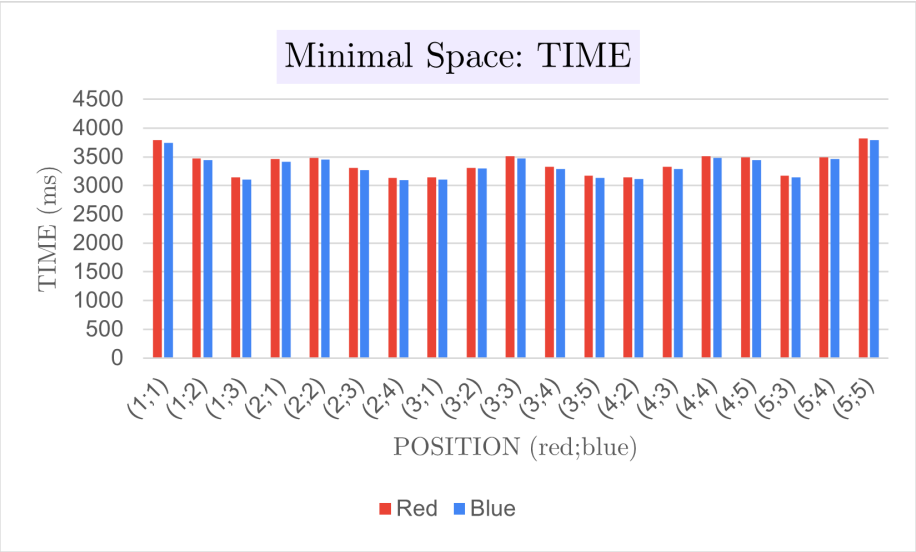


Figure 15: Movement taken in X-direction with Minimal Space Strategy. Non-meeting positions excluded.

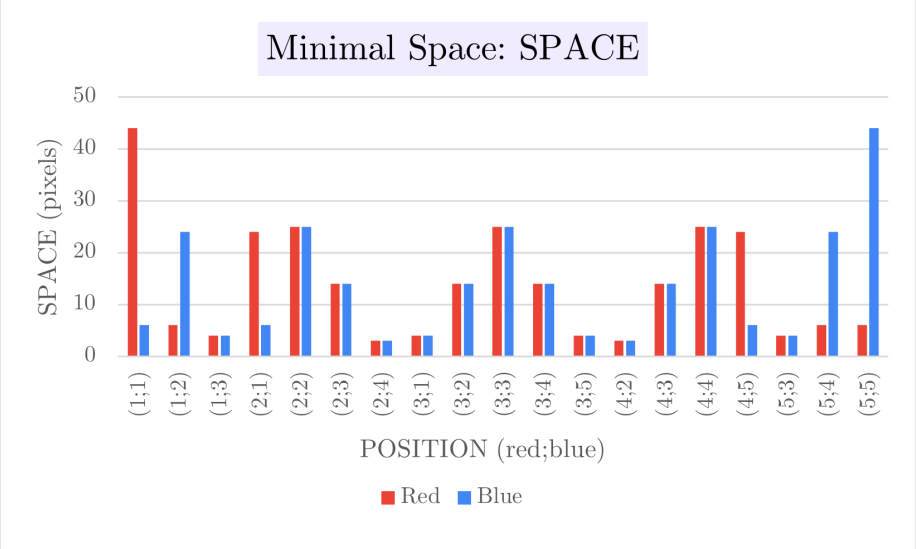


Figure 16: Time to reach end destination with Minimal Space Strategy. Non-meeting positions excluded.

We can see in Table 3 that the Minimal Space Strategy has the overall lowest completion time and space values. We can see that the average value is

approximately 3360 ms, which is only about 260 ms more than the rough nominal of 3100 ms. This is the lowest average of any strategy, and the trend continues with the maximum value and minimum values. The space values show a similar trend. The Minimal Space Strategy really lives up to its name, as both the average space value of about 28 pixels and the maximum of 50 are much lower than the corresponding values of 35 and 70 respectively in the Impatience and Give-Way Strategies. The lowest space value of 6 stays the same in every single strategy.

VALUES	MIN	MAX	AVG	NOMINAL
Time(ms)	3132.2	3819.2	3357.8	3095
Space(ms)	6	50	27.68	-

Table 3: Average, minimum and maximum time/space usage for the Minimal Space Strategy.

6.3 Advantages and Disadvantages

The Minimal Space Strategy is specifically desinged to minimize the time and space required to solve deadlocks when two robots meet in a corridor. This means that its biggest advantage is that it is the best at both time and space of all the strategies that have been tested in this study. It also has the great advantage that it always resolves the deadlock and is therefore very effective, similar to the Right-Way Strategy.

The biggest disadvantage of Minimal Space is that it is only this effective when we do not have more than two robots. If there are more than two robots that meet there might be trouble deciding which side to move to and the robots might end up not resolving the deadlock. This would lead to much bigger blockage problems than the other three strategies would have.

7 Conclusions

The increasing use of robots in indoor workplaces inspired us to investigate the problem of deadlock resolution for decentralized robots in an enclosed space. To do this, three different deadlock resolution strategies were studied in a simulation, and compared in terms of time- and space efficiency when resolving deadlocks in a corridor. The Give Way Strategy was the efficient across the board, but it was the least effective at consistently resolving the deadlock. The Right-Way Strategy was time efficient and always resolved the deadlock, but was not space efficient. The Impatience Strategy was space efficient but not very time efficient. It also failed to resolve the deadlock sometimes, but not as often as Give Way.

The three strategies could be applied in different contexts to utilize their individual strengths and weaknesses, and they could all be useful in the right situation. We hypothesize that the real life advantages and disadvantages of

the strategies will differ from in our simulation, since real robots have more complex components. We also see much optimization potential which could increase the usefulness of the strategies, and would be of interest for further studies.

After studying the first three strategies, we designed the Minimal Space Strategy, which generated the best results of all four. The Minimal Space Strategy has the fastest times, the lowest space usage and resolves the deadlock scenario in all cases, making it the most efficient and the most effective strategy to employ when two robots meet in a corridor.

References

[1] D. Langert, "Här är roboten som gör jobbet på nks," *SVT Nyheter*, 2016.

[2] M. Wilson, "Chapter 3 - automation system components," in *Implementation of Robot Systems*, M. Wilson, Ed., Oxford: Butterworth-Heinemann, 2015, pp. 39–73, ISBN: 978-0-12-404733-4. DOI: https://doi.org/10.1016/B978-0-12-404733-4.00003-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780124047334000035.

[3] L. Poudel, S. Elagandula, W. Zhou, and Z. Sha, "Decentralized and Centralized Planning for Multi-Robot Additive Manufacturing," *Journal of Mechanical Design*, vol. 145, no. 1, p. 012 003, Oct. 2022, ISSN: 1050-0472. DOI: 10.1115/1.4055735. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/145/1/012003/6926701/md_145_1_012003.pdf. [Online]. Available: https://doi.org/10.1115/1.4055735.

[4] M.-K. Kvalsund, K. Glette, and F. Veenstra, "Centralized and decentralized control in modular robots and their effect on morphology," eng, 2022.

[5] Z. X. et al, "A novel motion coordination method for variable sized multi-mobile robots," *Frontiers of Information Technology and Electronic Engineering*, 2023.

[6] S. Szomiński, W. Turek, and A. Byrski, "Socially-inspired fully decentralized robot coordination," *Simulation Modelling Practice and Theory*, vol. 119, p. 102 528, 2022, ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2022.102528. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X22000314.

[7] M. ˇCáp, "Centralized and decentralized algorithms for multi-robot trajectory coordination," Ph.D. dissertation, Czech Technical University, 2016.

[8] J. Thomas and R. Vaughan, "After you: Doorway negotiation for human-robot and robot-robot interaction," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3387–3394. DOI: 10.1109/IROS.2018.8594034.

[9] C. M. J. Changyun Wei Koen V. Hindriks, "Altruistic coordination for multi-robot cooperative pathfinding," *Applied Intelligence*, 2015.

[10] S. Khan and B. Athar, "De-centralized multi robot co-ordination and communication," *Mehran University Research Journal of Engineering and Technology*, vol. 38, no. 1, pp. 95–102, 2019, ISSN: 2413-7219. DOI: 10.22581/muet1982.1901.08. [Online]. Available: https://publications.muet.edu.pk/index.php/muetrj/article/view/743.

[11] X. An, C. Wu, Y. Lin, M. Lin, T. Yoshinaga, and Y. Ji, "Multi-robot systems and cooperative object transport: Communications, platforms, and challenges," *IEEE Open Journal of the Computer Society*, vol. 4, pp. 23–36, 2023. DOI: 10.1109/OJCS.2023.3238324.

[12] R. Marcotte, X. Wang, and D. e. a. Mehta, "Optimizing multi-robot communication under bandwidth constraints," *Auton Robot*, vol. 44, pp. 43–45, 2020. DOI: 10.1007/s10514-019-09849-0.

[13] K. Khateri, M. Pourgholi, M. Montazeri, and L. Sabattini, "A comparison between decentralized local and global methods for connectivity maintenance of multi-robot networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 633–640, 2019. DOI: 10.1109/LRA.2019.2892552.

[14] *Pygame v2.6.0 documentation*. [Online]. Available: https://www.pygame.org/docs/.

Appendix A The video that inspired this study

https://www.expressen.se/nyheter/har-drabbar-robotarna-samman-pa-sjukhuset/

TRITA-EECS-EX-2024:353
Stockholm, Sverige 2024