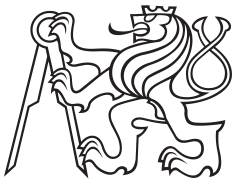
Bachelor Project



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Multi-Agent Path Finding with Human Presence

Sofie Konvalinová

Supervisor: Ing. David Zahrádka
Field of study: Cybernetics and Robotics
November 2025

Acknowledgements

We thank the CTU in Prague for being a very good *alma mater*.

Declaration

I declare that this work is all my own work and I have cited all sources I have used in the bibliography.

Prague, November , 2025

Prohlašuji, že jsem předloženou práci vypracovala samostatně, a že jsem uvedla veškerou použitou literaturu.

V Praze, . listopadu 2025

Abstract

In process to the end of semester 2025/2026 ...

Keywords: Multi-Agent Path Finding

Supervisor: Ing. David Zahrádka

Abstrakt

V procesu do konce semestru 2025/2026 ...

Klíčová slova: Multiagentní plánování cest

Překlad názvu: Multiagentní plánování cest za přítomnosti člověka

Contents

Figures

Tables

Chapter 1

Introduction

Multi-Agent Path Finding (MAPF) is a fundamental problem in robotics that focuses on computing paths for a group of agents moving from their origins to goal destinations without collisions. However, simply planning a path for each agent individually is often insufficient because they might block each other's way. As the number of agents grows, coordinating their movements becomes increasingly difficult. Consequently, efficient software is essential for running autonomous robot fleets in places like factories, airports, or computer games.

While current studies offers a wide array of solutions focusing on efficiency, optimality, and robustness of robot-to-robot coordination, less attention has been paid to the interaction between autonomous fleets and human workers sharing the same workspace. Based on the review of existing studies, most algorithms represent dynamic obstacles primarily as other predictable agents or static barriers. However, human unpredictability presents a significant challenge and a mandatory safety boundary that current solvers are simply not designed to guarantee.

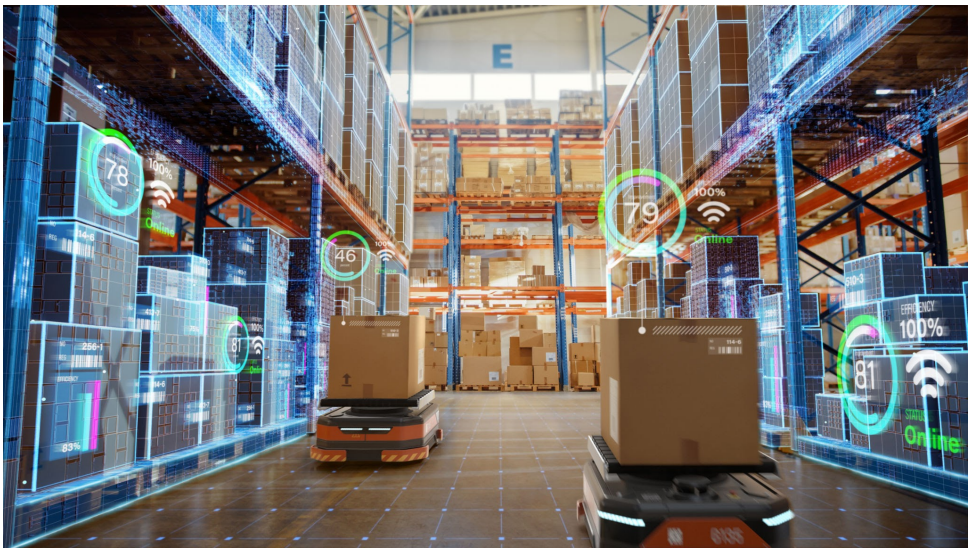


Figure 1.1: A visualization of a modern warehouse with fleet of autonomous robots.

This thesis focuses on the problem of integrating human maintenance workers or service technicians into a warehouse full of autonomous robots managed by optimal MAPF solvers. The primary goal is to extend current planners to strictly guarantee that the human worker always has a clear trajectory to the safety zone. While it is understood that this mechanism reduces optimality, guaranteeing human survival is the highest priority. The work considers both the optimality of different solutions and the practical implementations.

1.1 Contributions

The main contributions of this thesis are as follows:

- **Design of an Evacuation Assurance Mechanism:** The thesis introduces a mechanism designed to strictly guarantee an escape trajectory for the human worker, overriding robot optimality when necessary.
- **Implementation:** I developed a simulation scenario representing a mixed warehouse environment where autonomous agents interact with a human maintenance worker, modeling the specific challenges of dynamic obstacle avoidance.
- **Analysis of the safety efficiency of algorithm:** I performed an experimental evaluation to quantify the cost of safety. The thesis analyzes how the human escape route impacts the overall system compared to standard, non-safe MAPF solvers.

Chapter 2

Related Work

This chapter outlines existing research in optimal MAPF solvers, dynamic environments, and human-robot safety standards. The analysis highlights the lack of guaranteed escape mechanisms and defines the research gap which this project intends to fill.

2.1 Optimal MAPF Solvers

Multi-Agent Path Finding is the problem of finding collision-free paths for a set of agents from their start locations to their goal locations. The problem is widely applicable in automated warehousing, traffic management, and computer games.

2.1.1 Conflict-Based Search (CBS)

Older methods tried to plan paths for all robots at the exact same time using the standard A* algorithm. The problem is that with many robots, the math becomes too difficult for computers to handle quickly.

To fix this, Sharon et al. [?] created Conflict-Based Search (CBS). The idea is simple: instead of coordinating everyone perfectly from the beginning, let each robot find its own best path first, as if it were alone. Then, the system checks if any robots crash. It only fixes the specific spots where paths cross. This saves a lot of computing power because the system doesn't have to calculate every move for every robot at once.

CBS operates on two levels:

- **High-Level:** Acts like a manager. It looks for collisions between robots. If two robots crash, it creates a new rule (constraint) to stop that specific crash from happening again.
- **Low-Level:** Acts like a worker. It finds a path for a single robot that follows the rules given by the High-Level manager.

This thesis uses this method because it makes it easy to add strict safety rules, which is exactly what we need to keep humans safe.

2.1.2 Prioritized Planning

A common alternative is Prioritized Planning (PP). This method works like a priority list. Each robot is given a number based on importance. The first robot plans its path, then the second robot plans its path around the first one, and so on.

This method is much faster than CBS. However, it has a major flaw: it is not guaranteed to find a solution. Sometimes, a lower-priority robot gets blocked by the robots before it and cannot find a path, even if a solution actually exists. In safety-critical situations, this is risky because the system might fail to calculate a path when it is needed most.

2.2 Dynamic Environments and Replanning

Standard MAPF usually assumes a static world where we know where every obstacle is before we start. But real warehouses are dynamic—machines move around, and people walk through the aisles.

2.2.1 Replanning Approaches

The most common way to handle surprise obstacles is Replanning. Algorithms like Lifelong MAPF calculate new paths every few seconds or whenever a robot gets stuck.

This works well for efficiency, but it has a problem: it is reactive. The robot only changes its plan after it sees an obstacle. For human safety, this is not good enough. By the time the robot realizes it needs to move, the robots might have already surrounded the human, blocking their way out.

2.2.2 Robustness and k-Delay

Another method is to make plans that can handle delays. This is called k-robustness. It ensures robots don't crash even if they are a few seconds late.

While this helps with small mistakes or delays, it doesn't guarantee that a path to the exit will stay open for a person who is moving unpredictably. It makes the plan stronger against delays, not against a moving human.

2.3 Human-Robot Safety in Shared Workspaces

Keeping people safe in warehouses full of robots is extremely important for modern industry.

2.3.1 Local Collision Avoidance

Most safety systems today use sensors like cameras or lasers (LiDAR). Robots use these to see nearby objects. If a human gets too close, the robot stops or

moves aside.

This logic prevents a physical crash, but it doesn't solve the problem of getting trapped. If many robots stop around a human to avoid hitting them, they can accidentally create a wall that blocks the human from leaving.

2.3.2 Predictive Human Modeling

Advanced research tries to guess where a human will walk using Artificial Intelligence (Machine Learning). If robots know where the human is going, they can stay out of the way.

However, in an emergency, people act unpredictably and might run in random directions. These computer guesses are not 100% certain, so they cannot guarantee the absolute safety needed to save a life.

2.4 Gap in Research

Based on the literature review, we identified a significant gap. Current research is excellent at coordinating robots to work efficiently and avoiding immediate crashes. However, there is no global strategy that treats **human evacuation** as a strict, mandatory rule.

Existing solvers focus on avoiding collisions, not on keeping a path open to an exit. Because of this, standard algorithms might create efficient plans that accidentally trap the human agent in a **deadlock**. This thesis fixes this problem by building a guaranteed escape route directly into the planning logic, ensuring that safety is just as important as efficiency.

Chapter 3

Problem Formulation

In this chapter, we define exactly what problem we are solving. We set the "rules of the game": the map, the inputs, the players (robots and humans), how time works, and what counts as a valid solution.

3.1 The Environment

Imagine the warehouse as a checkerboard. We formally represent this grid as a graph.

[Graph Representation] The workspace is a graph $G = (V, E)$, where:

- V (Vertices) are the squares of the grid.
- E (Edges) are the connections between neighbors. If two squares are next to each other, a robot can move between them.

[Obstacles and Free Space] Some squares are blocked by walls. We call these static obstacles X. Robots cannot enter these squares. All other squares are "Free Space" (V_{free}).

[Exit] The Safety Zone E is a specific set of squares that represent the emergency exits. The goal for the human is to reach any square in this zone.

[Human] Human H is a specific set of squares that represent the human standing position. The goal for the human is to reach exit.

3.2 Problem Input

Before the algorithm starts, it receives the following information:

[Inputs] The input consists of:

- The map (Graph G, Obstacles X, Exit E, Human H).
- A team of k robots, where each robot has a specific Start (s) and Goal (g).
- The starting position of the Human agent (p_H).

3.3 Time and Actions

Time in our model is not continuous like in the real world. Instead, it works in steps, like a turn-based strategy game.

[Discrete Time] Time moves in logical ticks $t = 0, 1, 2, \ldots$. In every tick, everyone moves at once.

[Actions] In each turn, a robot can do only two things:

- **Move:** Go to an adjacent square (up, down, left, right).
- **Wait:** Stay on the same square.

3.4 Agents

We have two teams in this environment: the robots we control and the human we do not control.

3.4.1 Robot Agents

Let A be a group of autonomous robots. Each robot has a start s_i and a goal g_i. Our job is to find a path (a list of moves) that gets the robot from start to goal.

3.4.2 The Human Agent

We also have one Human Agent, denoted as H.

- The human is located at position p_H.
- Unlike robots, the human **does not follow our orders**. Their movement is "non-deterministic," meaning they can decide to move anywhere or stop unexpectedly.

3.5 Conflicts and Safety Constraints

A valid plan must follow strict rules to avoid accidents.

[Robot-Robot Conflict] Robots crash if:

- **Vertex Conflict:** Two robots try to enter the same square at the same time.
- **Edge Conflict:** Two robots try to swap places (head-on collision).

The most important rule in this thesis is about the human:

[Guaranteed Evacuation Constraint] A situation is considered **safe** only if the human has an open path to the Safety Zone. Even if the robots are not touching the human, they must never form a wall that blocks the human's escape route.

3.6 Problem Objective

The final goal is to find paths for all robots so that:

1. Every robot reaches its goal.
2. Robots do not crash into each other.
3. The **Guaranteed Evacuation Constraint** is always true (the human is never trapped).
4. The solution is efficient (we want to minimize the total time/cost).

Chapter 4

Method

jak resim ten problme, popsany algoritmus, jak to funguuje, nakonci sekce to kazdy umi naiplementovat sam, vcetne popis LNS

This chapter describes the algorithmic framework used to solve the Safety-Aware MAPF problem. We propose a solution that adds a strict safety check to the standard pathfinding process. The core idea is simple: at every single time step, we check if the human can escape. Then, we use a method called Large Neighborhood Search (LNS) to iteratively fix and improve the plan until it is both safe and efficient.

4.1 Algorithmic Framework

The system works as an iterative solver. Unlike traditional methods that only care about robots crashing into each other, our framework adds a second layer that focuses purely on human safety. The process follows three logical steps:

1. **Initialization:** First, we generate a basic plan for all robots. At this stage, the plan might not be perfect, and it might even be unsafe for the human.
2. **Safety Validation:** Next, the system looks at every time step of the plan. It checks if the human still has an open path to the Safety Zone, given the current positions of the robots.
3. **Iterative Optimization (LNS):** Finally, the solver tries to improve the plan. It uses local search operations to find and fix the specific parts of the plan where robots are blocking the human or moving inefficiently.

4.2 Safety Verification Mechanism

To guarantee the safety of the human worker, we use a method called Reachability Analysis.

At every time step t, we treat all robots as if they were static obstacles (frozen in place). Then, we run a standard graph search algorithm (Breadth-First Search) starting from the human's position.

- The algorithm explores all accessible squares in the grid.
- If the search finds a path to any square in the Safety Zone, the situation is marked as Safe.
- If the algorithm cannot find a path (meaning the robots have formed a wall around the human), the situation is marked as Unsafe, and this plan is rejected.

4.3 Optimization via Large Neighborhood Search (LNS)

Finding the perfect solution in this environment is very hard for a computer. Therefore, we use Large Neighborhood Search (LNS). This is a smart heuristic method that can handle complex rules efficiently.

The LNS algorithm works on a "Destroy and Repair" principle:

- **Destroy Operator:** The system selects a group of robots and deletes their current paths. It specifically targets the robots that are causing safety violations (the ones blocking the human).
- **Repair Operator:** The system calculates new paths for these specific robots using A*. During this calculation, the robots are strictly forbidden from blocking the human's escape route.

This process repeats in a loop until the system finds a solution that is valid, collision-free, and safe for the human. This approach allows the algorithm to adaptively "clear the way" for the human by moving the problematic robots aside.

Chapter 5

Experiment results

map yco pouziviam, na jakym procesoru, jaka skala exprerimentu ,pak jdu k jednotlivym experimentum, runtime evalluation -1000 evaluaci, runtime-value - 100s, tables, graphs

Chapter 6
Conclusion

Lorep ipsum [1]

Bibliography

[1] J. Doe. *Book on foobar.* Publisher X, 2300.