Multi-Agent Path Finding for Robots in Large-Scale Warehouses

Abhay Chhagan Karade*, Vaibhav Nandkumar Kadam† and Akash Ashok Thorat‡
Robotics Engineering,
Worcester Polytechnic Institute
Email: *akarade@wpi.edu, †vkadam@wpi.edu, ‡aathorat@wpi.edu

Abstract—**The Multi-Agent Pathfinding (MAPF) is a crucial problem of warehouse mobile robots to find collision free path to the target position. In this paper, we explore MAPF algorithms that produce an optimal and suboptimal path for a fixed number of robots to navigate in a warehouse environment. In addition, this paper also elaborates on the comparative analysis of the studied algorithms, with increasing environmental complexity and increasing number of agents. The compared parameters are the total cost, flowtime and makespan of each algorithm.**

I. Introduction

Logistics operations in large scale warehouses require efficiency in operation for an increased throughput. Using multiple robots for pickup & delivery of material from one point to another increases the efficiency of logistic operations. Increase in the number of robots in warehouses does bring challenges like increased congestion and problems of collision-free path planning and task allocation. When multiple robots are considered, finding a path that is collision-free and optimal is necessary. We investigate the usage of Multi-Agent Path finding (MAPF) techniques that aid to solve the problems in large-scale warehouses. We propose to implement and study various MAPF algorithms in order to solve the above-mentioned problems. We carried out experiments in simulation to test and validate results.

The project report is organized as follows: Next section discusses the related work in literature that provides the overview of algorithms for MAPF. Section III deals with methodology implemented for solving the problems of collision-free path finding. Section IV will briefly highlight the challenges encountered in the project. Section V will show the results of algorithmic implementations. Finally, Section VI will have the concluding remarks for the project

II. Related work

Multi-agent Path Finding (MAPF) has been a largely studied field that has applications to warehouse management, airport towing, and shopping centers. etc. MAPF deals with techniques to find collision-free paths for multiple robots. A. Bolu *et. al* [1] discusses a collision-free path planning algorithm for mobile robots on a warehouse grid. It introduces a modified A* algorithm which considers turning costs of the robots. G. Sharon *et. al* [2] introduces to a two level algorithm called Conflict Based Search (CBS) which does not convert the problem in single agent models. The path search is carried over conflict tree (CT) on conflicts based on individual agents. Large warehouses do face the problem of traffic flow and task allocation, Y Shi *et. al* [3] discusses a approach that handle task allocation of multiple robots in large warehouses using a decentralized auction bid scheme and the robot with lowest bid wins to which the task is allocated. Each robot's path is planned using Floyd algorithm, moreover a collision avoidance scheme is discussed to sufficiently avoid robot collisions. Silver, D *et. al* [7] highlights three different approaches to, extend the spatial A* into time domain. The algorithms presented are decoupled approaches that break down the problem into a series of single-agent searches. Cooperative A* (CA*) searches space-time for a non-colliding route. Hierarchical Cooperative A* (HCA*) uses an abstract heuristic to boost performance. Finally, Windowed Hierarchical Cooperative A* (WHCA*) limits the space-time search depth to a dynamic window, spreading computation over the duration of the route. Varambally, S *et. al* [9] discuss shortcomings of the Action Dependency Graph (ADG) framework which is provided to consider simplifying assumptions such as - robot moves in discrete time and minimal considerations about robot dynamics-affect the performance of the robots. They argue that the ADG framework provides the same robustness guarantees as the single-agent framework. To improve it authors used the Rolling-Horizon Collision-Resolution framework to solve MAPF problems with a persistent stream of online tasks. They also compared the standard MAPF model with many of its more complex variants, such as MAPF with rotation, k-robust MAPF, and continuous-time MAPF (taking robot dynamics into account). They found that Using windowing and considering rotation time during planning significantly improves throughput in most cases.

III. Methodology

Our methodology is shaped by unifying terminology defined by Stern R. *et. al* [4] for describing common MAPF assumptions and objectives. Basic assumptions in Classical MAPF are time is discretized into time steps, every action takes exactly one time step, and in every time step, each agent occupies exactly a single vertex.

To be aligned with the current research we will incorporate commonly used types of Conflicts in Classical MAPF which are Refer Fig 1.a Edge conflict, Fig 1.b Vertex conflict, Fig 1.c Following conflict, Fig 1.d Cycle conflict, Fig 1.e Swapping

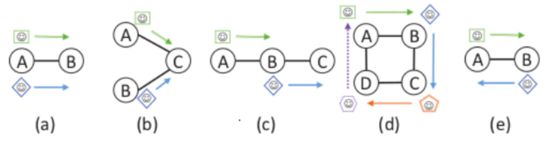

Fig. 1: Types of Conflicts

conflict. and Agents behavior at target in Classical MAPF is either stay at target or disappear at target, Our application area is in warehouse, where the stay at target behavior is more appropriate.

Evaluation measure of MAPFs are two objective functions, first is 'Makespan', it has been used extensively by compilation-based MAPF algorithms, while the 'sum of costs' has been used by most search-based MAPF algorithms.

Stern R. *et. al* [4] Introduces a new grid-based benchmark for MAPF, and demonstrates experimentally that it poses a challenge to contemporary MAPF algorithms. Which could be incorporated to test our algorithms.
Generally used different graphs for evaluating MAPF algorithms are Open N × N grids, N × N grids with random obstacles, Warehouse grids, we are planning on using N × N grids with random obstacles which will replicate the warehouse environment for prototyping. This could be created with the use of 2D simulator and to implement this work in 3D Gazebo physics engine would be a useful tool.

In our MAPF implementation We will keep the number of agents constant and test algorithms such as CBS, A* and it variants to compare the performance, however by keeping runtime constant we will evaluate the maximum no of agents each of the MAPF algorithm could handle.

Here we further discuss about various alogrithm that are studied and implemented briefly. as follows.

A. Space Time A*

A* is an optimal / sub-optimal search based algorithm ,but it cannot be directly implemented for Multi-agents as it results to be an extremely inefficient MAPF Solver. As A* search ignores the presence of other agents, or perhaps treats them as stationary obstacles. For optimal and cooperative MAPF there is no way to represent the routes of agents on a stationary map (space map). To overcome this problem, we extend the map to include a third dimension: time. Hence forming a space-time map. A* search can now be used on the space-time map. The goal of the agent is to reach the destination at any time. A* will find the route that achieves this goal with the lowest cost; this is the quickest path to the destination. Please refer Fig. 2 that illustrates planing using A* in space-time grid map (space time A*).

B. Prioritised Planning

Prioritised Planning, is a simple and straightforward approach of decoupled planning. In this approach, priorities are assigned to individual agents in a pool of multiple agents. A path search algorithm, generally space-time A* is run for

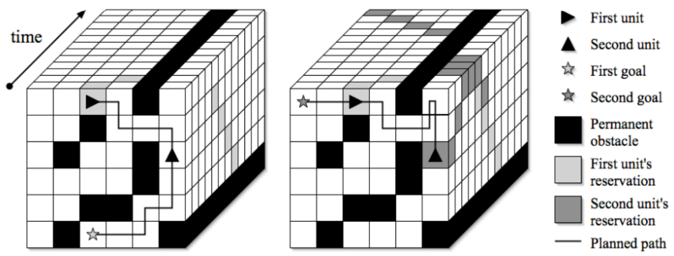

Fig. 2: Space-time grid map with 2 agents

each individual agent in the descending order of the agents priority. This ensures that the newly planned path for the current agent does not collide with the already planned paths of the other agents. This can be categorized as a sub optimal approach. Although being fast, unfortunately it can sometimes be incomplete. This means, not all agents can find a path and successfully maneuver to the goal position. This is a major limitation of prioritized planning. Also, the cost of the solution depends heavily on the priorities of the agents.

C. Conflict based Planning

CBS first plans shortest paths for all agents independently (which can be done fast) using a search based algorithm, generally A* search. These paths are not allowed to collide with the environment, but are allowed to collide with the paths of other agents. If all agents find a collision free path then the algorithm simply executes their motion until the reach the goal position. Otherwise, it chooses a collision between two agents and considers two cases recursively: 1) Prohibit agent a from being in cell x at time step t. 2) Prohibit agent b from being in cell x at time step t. Based on the above constraints it replans the path. The hope is that CBS finds a collision-free solution before it has imposed all possible constraints. CBS is slower than Priority Planning, but delivers a complete solution. This means, every agent is able to find a path and successfully maneuver to its goal position, if a valid path exists. CBS is a two level search algorithm comprising of high level and low level. High level of CBS searches the binary constraint tree. Low level of CBS finds new shortest path for the agent with the newly imposed constraints. Please refer Fig. 3 that illustrates the flow chart which provides an overview of the CBS algorithm.

IV. Approach

In the given course period we investigated and studied various MAPF algorithms by surveying the literature. In order to understand the performance of each algorithm we started by prototyping fundamental algorithms like Spatial A^* and simulating them in a 2D warehouse environment. The warehouse environment was created using python and matplotlib. Four warehouse environments were created labeled as env3, env4, env5 and env6 in order of increasing complexity and an increasing number of obstacles.

We gradually developed the understanding to use more sophisticated algorithms in a MAPF environment. Through

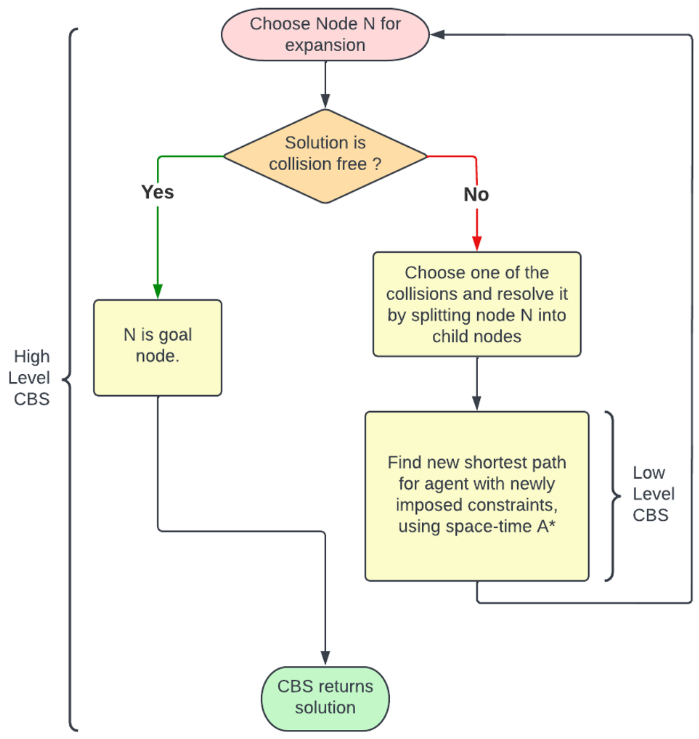
Choose Node N for expansion
Solution is collision free ?
Yes
No
N is goal node.
Choose one of the collisions and resolve it by splitting node N into child nodes
Find new shortest path for agent with newly imposed constraints, using space-time A*
High Level CBS
Low Level CBS
CBS returns solution

Fig. 3: CBS Overview

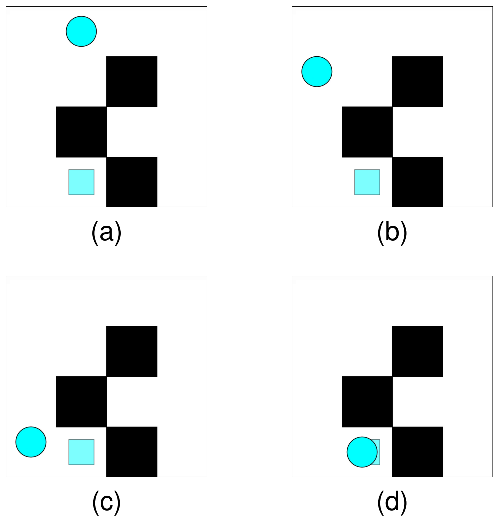

Fig. 4: Path Planning of Single agent with Space-Time Astar

this process, we learned the limitations of Spatial A* for a MAPF environment in the initial stage of the project itself. This also helped us to understand the necessity of exploring other variants of A^* such ah the Space-Time A^* which considers additional time dimension rather than just the 3D configuration space in a Spatial A^*. We found that Space-Time A^* works better in discrete time-steps. To work in continuous time-steps A^* could be substituted with SIPP which is an advanced version of A^* that allows for real-valued action execution times.

Furthermore, we implement the CBS algorithm to plan for multiple agents in the warehouse grid. We simulated this algorithm in our custom warehouse environments and analyzed its performance based on our evaluation metrics. Finally, the same process was repeated for Prioritized planning.

TABLE I: Schedule

Week	Task Divison		
	Vaibhav	Abhay	Akash
1-2	Literature review		
3-5	Simulation Env Setup	Prototyping MAPF algorithms	
6-7	Prototyping MAPF algorithms		Simulation
8-9	Integration of algorithms with Sim Env		
10-11	Testing and validation		
12	Buffer Time		
13 - 14	Code & Report documentation		

V. Results

Here we consider three colors for multiple agents namely Aqua, Blue, Yellow with agent as circle and square as respectively goal position.

We started with planning using Space-Time Astar for a single agent with less obstacles. Please refer Fig. 4 that illustrates the output frames. We see that for path finding for single agent is as same using the A^* without consideration of time steps.

In order to overcome the above mentioned problems of collision on edge as well as vertex, we studied and implemented Prioritised Planning as discussed in Section III. Here it considers priority in a descending order for all agents and plan a collision free path. Please refer Fig.(4) t = 4 it illustrates the situation where blue-agent waits for the Aqua-agent to pass such that the edge constraint is satisfied. Moreover the in Fig (6) t= 6, yellow agent takes detour to avoid collision with Aqua-agent. Similarly when yellow reaches near Blue-agent at t = 9 it avoids going via Blue-agent and reaches goal to avoid collision. Even though it avoids collision it is inefficient thus categorized as sub-optimal. Further we study Conflict Based Search as which is two level algorithm which uses Space-time A^* in the lower level and currently in progress of implementation.

We have developed a simulation setup refer Fig. 12 for testing our MAPF algorithm implementations for warehouse environment. The software setup is developed such that it can input varied obstacles or docking pods. We have depicted different configurations of gangways, throughways, alleys. We illustrate with a example showing in Fig. 6 how CBS algorithm plans path for 2 agents where cyan colored agent takes a different path in order to avoid collision with blue agent.

In order to emulate the actual scenario of warehouse we do simple 3D visualisation in RViz with implementation of Conflict based search in ROS2 C++ Fig. 7 shows 2 agents spawned in Grid environment. In Fig. 13 we show the green agent waits giving priority to the Red agent in prioritised planning in warehouse environment with 7 agents.

Please check the output video of CBS solver **[link]** and Prioritised solver **[link]** with 10 agents.

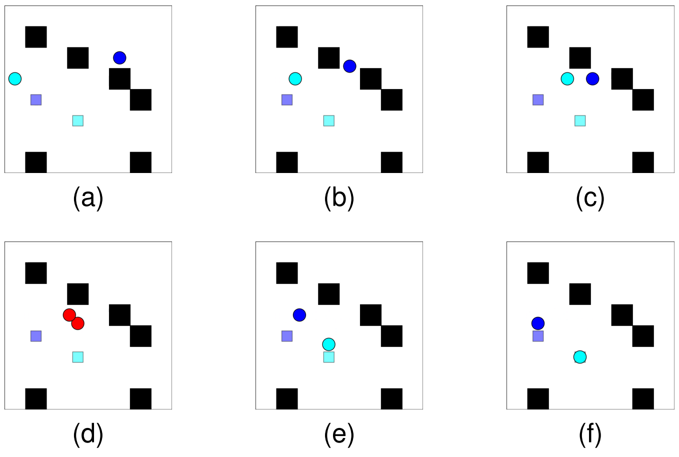

Fig. 5: Independent planning using A^* for respective agents.

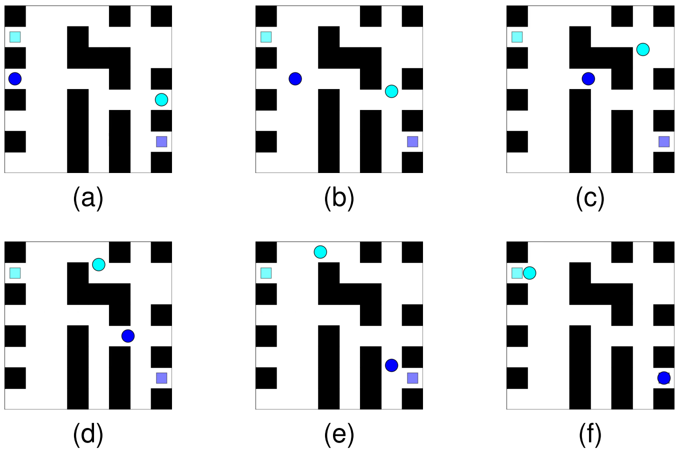

Fig. 6: CBS MAPF solver

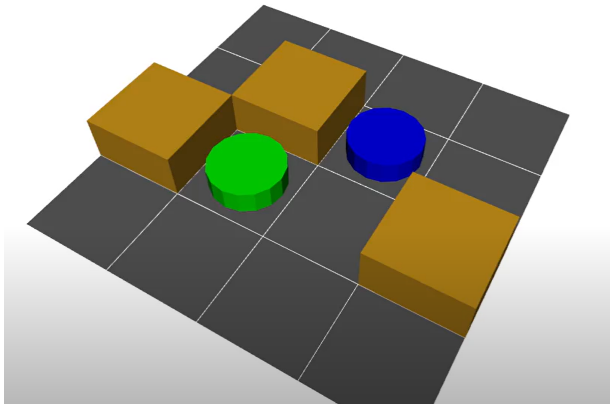

Fig. 7: CBS Algorithm implementation in 3D

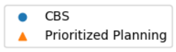

Fig. 8: CBS and PP

VI. Challenges

In this project we conducted experiments with multi-agent robots to find their path in a simulated warehouse environment. While implementing Conflict Based Search in early prototyping we faced issues inorder resolve conflicts for n agents as increased. We were able to overcome those to implement CBS successfully and perform our experiments. Further we developed a gazebo world of Warehouse in order simulate them by using Nav2 Stack for each agents where our MAPF solver shall provide the required path for the robot to follow in a warehouse map. We faced issues while integrating Navigation stack for N agents so we had to visualise our algorithms in rviz without physics engine. As we increased N number of agents doing experiments in constrained resources takes alot of time to successfully investigate performance metrics so limit the agents to 10.

VII. Discussion

In this section we investigate the performance of Multi-agent Path Finding Algorithms. We started with implementation of the independent planning using building block algorithm A*.

Our performance analysis is based on the three criteria, total cost, flow-time and makespan on environments with increasing complexity levels. In the attached plots we considered four environments where env 3 is a simple environment with straight paths between the warehouse shelf's , then we added static obstacles and made it more complex in subsequent environments. In order to have a fair analysis we keep number of agents as ten along with the same start and goal locations for all the environments while performing experiments for the various mapf algorithms. We did this performance analysis for two MAPF algorithms, CBS and Prioritized Planning, The next subsection will discuss about the individual criteria and observations.

A. *Total Cost*

The Total cost is the total path length of all the agents. In ideal situation the total cost would be the sum of optimal path lengths of all the agents, however to avoid conflicts the algorithms make agents to traverse farther distances, which eventually increases the cost. In plot. 9 we can observe that up-to env no 5 the performance of both algorithms were comparable with CBS performing slightly better than PP. However there is significant change in performance for env 6 which is relatively complex environment with multiple bottlenecks, CBS found overall shortest path for agents and solution from PP is sub-optimal.

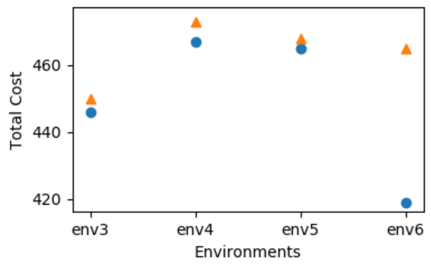

Fig. 9: Total Cost Analysis

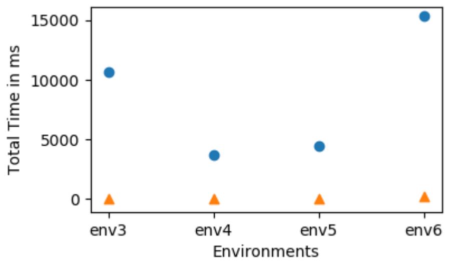

Fig. 10: Flow-time Analysis

B. *Flow-time*

Flow-time is the time required for the planning of all the agents. Flow time is an important factor when planning time is priority and the application could tolerate the sub-optimal expenses. The CBS algorithms first plan the optimal path for all the agents and then conflicts are resolved one by one so the flow time is directly proportional to the number of conflicts that occur. the flow time plot 10 shows the performance for env 5 and env 6 is better than env 3, it could be because it encountered more conflicts for agents in env 3. In the case of Prioritised Planning it plans path of agents as per assigned priorities, so it considers the conflicts with higher priority agent as a obstacle and plans the path around it, so planning time is more dependent on exploring thought obstacles and which is nearly same for all the environments.

C. *Makespan*

Makespan is the completion time of agent which has the maximum execution time. In our case we are not considering non-holonomic constriants, all agents are point robots and travel with constant speed and instant acceleration. considering these assumptions for our case the longest path length agent amoung all would be Makespan. Makespan is more relevant when considered all the dynamics of the robots, in that case even if the path length is shorter but it includes multiple turns and eventually the Makespan will increase. considering our assumptions we can observe that CBS has better performance in finding shorter path lengths for the agents.

VIII. Conclusion

Multi-Agent Path Finding for Robots in Large-Scale Warehouses is a NP hard problem statement however crucial for the boost in the warehouse automation development. In this project we implemented two MAPF algorithms -CBS and

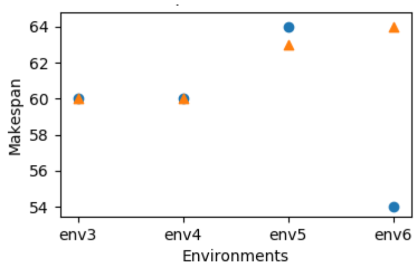

Fig. 11: Makespan Analysis

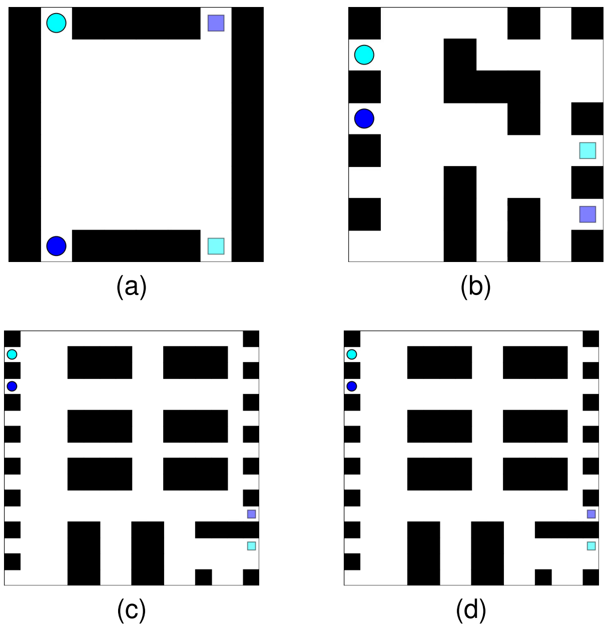

Fig. 12: Warehouse Simulation Setup with varied obstacles and Start Goal Position.

Prioritised Planning- in vanilla form, and visualized using a 2D warehouse environments. We did performance analysis using three metrics Total cost, flow-time and Makespan. we observed that the CBS finds a shorter path with less total cost however lacks in flow-time metric. Prioritised Planning(PP) has a better performance in flow-time however it generates sub-optimal solution. The performance of PP depends on priorities assigned to the agents, in certain cases where the goal locations are in bottleneck the PP might not able to find solutions because a high priority agent already occupied the bottleneck. overall the choice of the algorithm depends on application area and the space constrains. The extended goals of implementation in 3D environment is implemented for a simple environment.

Appendix A
Individual contributions

A. ***Abhay Karade***

- Literature review
- 2D Test Environment Development
- Algorithmic implementation:
    - Spatial A*.

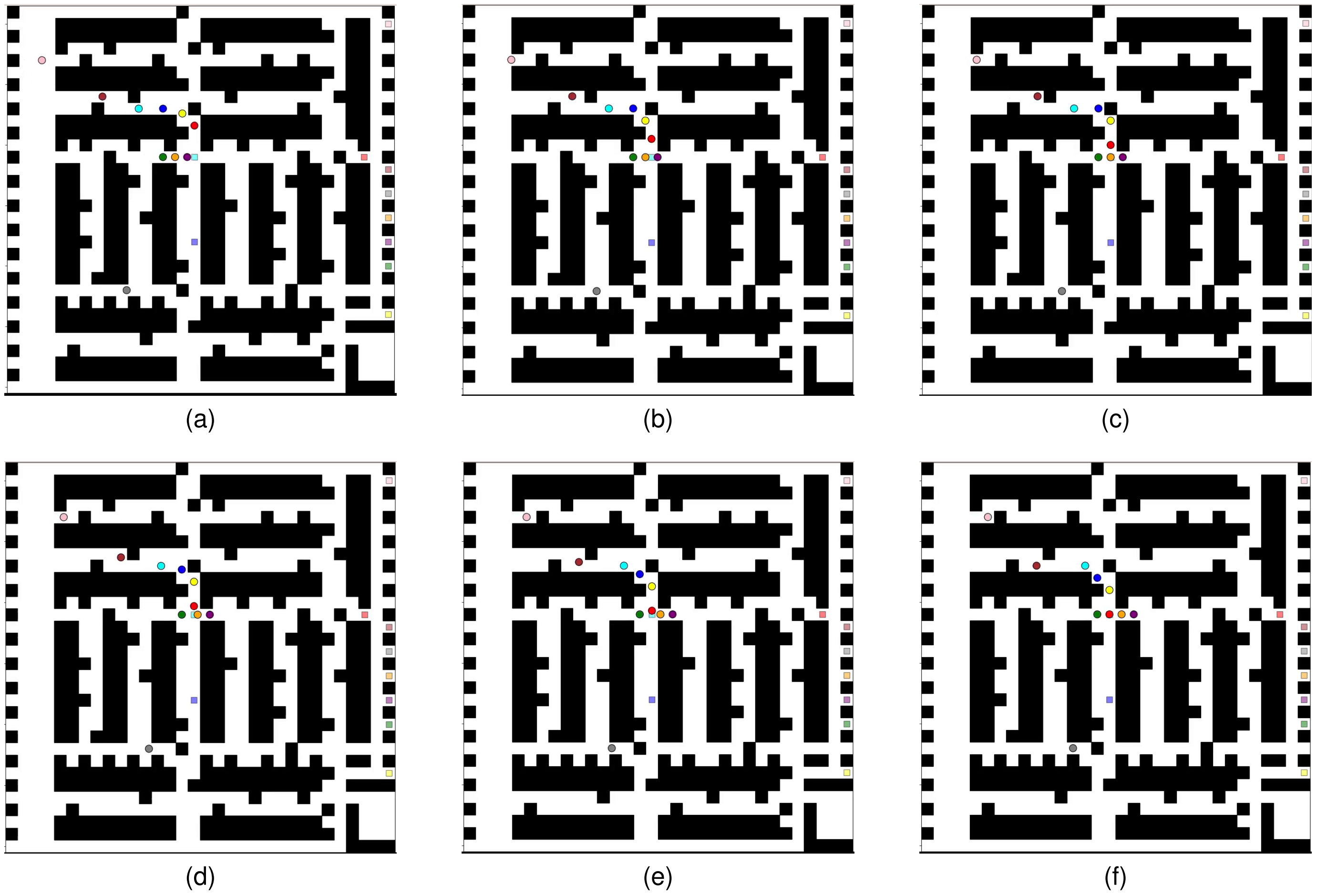
Fig. 13: Green agent waits for High Priority Red agent.

  - Prioritized Planning.
- Team Coordination and Project Management

B. *Vaibhav Kadam*

- Literature review
- Project Documentation
- Algorithmic implementation:
  - Space time A*.
  - CBS.
- Simulation and 3D environment visualisation.

C. *Akash Thorat*

- Literature review
- Algorithmic implementation:
  - Spatial A*.
  - Space time A*.
  - CBS.

References

[1] A. Bolu and Ö. Korçak, "Path Planning for Multiple Mobile Robots in Smart Warehouse," 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA), 2019, pp. 144-150.

[2] Sharon, G., Stern, R., Felner, A., Sturtevant, N. (2012). Conflict-Based Search for Optimal Multi-Agent Path Finding. Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 563–569. Toronto, Ontario, Canada: AAAI Press.

[3] Y. Shi, B. Hu and R. Huang, "Task Allocation and Path Planning of Many Robots with Motion Uncertainty in a Warehouse Environment," 2021 IEEE International Conference on Real-time Computing and Robotics (RCAR), 2021, pp. 776-781.

[4] Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., … Bartak, R. (2019). Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. doi:10.48550/ARXIV.1906.08291

[5] mapf.info : webmaster: Sven Koenig — Main / Welcome! — browse. Retrieved 11 September 2022, from http://mapf.info/

[6] Boyarski, E., Felner, A., Stern, R., Sharon, G., Betzalel, O., Tolpin, D., Shimony, S.E. (2015). ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding. SOCS.

[7] Silver, D. (2005). Cooperative Pathfinding. AIIDE.

[8] Phillips, M.; and Likhachev, M. 2011. SIPP: Safe Interval PathPlanning for Dynamic Environments. InICRA, 5628–5635.

[9] Varambally, S., Li, J., Koenig, S. (2022). Which MAPF Model Works Best for Automated Warehousing? Proceedings of the Symposium on Combinatorial Search (SoCS), 190–198.

[10] Andreychuk, A.; Yakovlev, K.; Boyarski, E.; and Stern, R. 2021.Improving Continuous-Time Conflict Based Search.InAAAI,11220–11227.

[11] Enginbaglayici (no date) Enginbaglayici/ConflictBasedSearch: Conflict-based search for multi-agent path finding (MAPF), GitHub. Available at: https://github.com/enginbaglayici/ConflictBasedSearch (Accessed: December 4, 2022).