WPI

Multi-agent Motion Planning for Non-Holonomic Mobile Robots via Heuristic Optimization

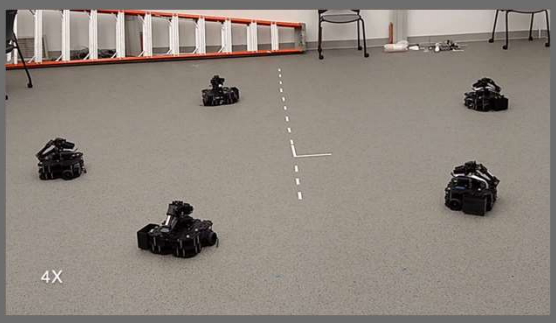
Vishrut Bohara

Aadesh Varude

Om Vinayak Gaikwad – Team Leader

Research Problem and Description

- The problem under consideration is the design of motion planning algorithm that generates safe, dynamically feasible, and near-optimal trajectories for multiple non-holonomic mobile robots.
- The state-of-the-art motion planning approach divides the path search of the agents based on priority, which may lead to infeasible subproblems.
- Hence, we propose a heuristic based approach that highlights the non-optimality of the state-of-the-art approach.



The problem under consideration is a motion planning approach that generates safe, dynamically feasible, and near-optimal trajectories for multiple non-holonomic mobile robots.
The state-of-the-art motion planning approaches divide the path search of the agents based on priority.

However, prioritized optimization may lead to infeasible subproblems for lower-priority robots due to lack of consideration by higher-priority robots.

Hence, we propose a heuristic based approach that highlights the non-optimality of the state-of-the-art approach.

Application Impacts

Warehouse:

In the warehouse, multi-agent path planning could help in various task co-ordinations efficiently.

Surveillance:

Multi-agent path planning is used in surveillance to coordinate the movements of multiple drones or other surveillance devices, ensuring that they cover the area of interest without overlap and that they collect the necessary data efficiently.

Search and Rescue Operations:

In search and rescue operations, multi-agent path planning can be used to coordinate the movements of multiple search and rescue robots, allowing them to cover more ground and find survivors faster.

3

Worcester Polytechnic Institute

As one can infer from the slides that the multi agent robots can be used in sync for execute various intricated task in warehouses, This coordinated approach allows the robots to work together efficiently and effectively, optimizing workflow and increasing productivity. Moreover, the fleet of robots can also be utilized for surveillance purposes, providing maximum area coverage of a region of interest. This capability could prove particularly useful in situations where human surveillance may be too dangerous or difficult to achieve.

In addition to these applications, multi-agent path planning can be leveraged for search and rescue operations, particularly in areas that are hard to reach or dangerous for humans. The use of multiple robots can improve the effectiveness of the search and rescue mission, allowing for wider coverage of the area and potentially leading to the faster discovery of survivors.

Hence Multi Agent Path Planning is crucial.

Literature Review

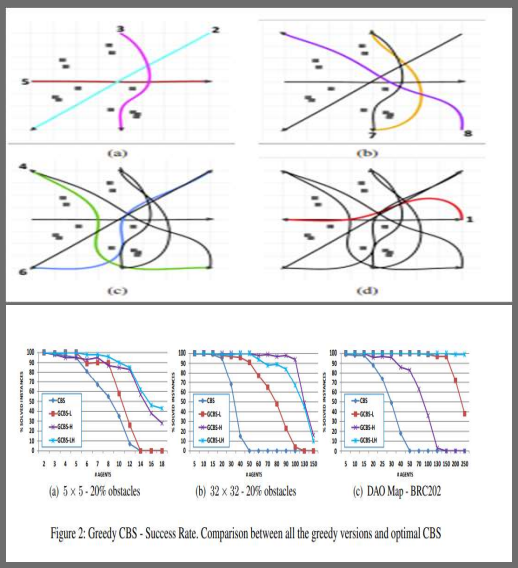


Figure 2: Greedy CBS - Success Rate. Comparison between all the greedy versions and optimal CBS

Efficient Trajectory Planning for Multiple Non-holonomic Mobile Robots via Prioritized Trajectory Optimization

- The paper presents a method for multiple robot path planning in obstacle-rich environments using trajectory optimization while taking into account non-holonomic constraints. The proposed algorithm is demonstrated through simulations and real-world experiments.

Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem

- Proposes several modifications to the Conflict-Based Search (CBS) algorithm like ECBS, GCBS,BCBS to improve its runtime performance while still maintaining near-optimal solutions for the Multi-Agent Pathfinding.

Li, Juncheng, Maopeng Ran, and Lihua Xie. "Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization." *IEEE Robotics and Automation Letters* 6, no. 2 (2020): 405-412.

4

Worcester Polytechnic Institute

Efficient Trajectory Planning for Multiple Non-holonomic Mobile Robots via Prioritized Trajectory Optimization

The authors present the solution in three folds:

a) First an efficient trajectory planning approach is implemented to find feasible and collision free in the environment.

a) Secondly, a safe corridor is created and prioritized trajectory planning method is implemented to solve the multi-robot planning problem and increase computational efficiency.

b) In the last step, the trajectories are smoothened, and testing is done on multiple agents.

Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem

- Proposes several modifications to the Conflict-Based Search (CBS) algorithm to improve its runtime performance while still maintaining near-optimal solutions for the Multi-Agent Pathfinding.
- Greedy CBS (GCBS) algorithm employs the same framework as CBS but offers greater flexibility in both the high-level and low-level searches Problem (MAPF).
- BCBS variant applies focal search to both levels of CBS using different heuristics.
- ECBS algorithm generates a CT node and the low level search returns two values to the high level search the cost of node and the minimum lower bound of all nodes in the OPEN list of the high-level search.

Literature Review

Fig. 1. Our method generates a Risk-DTRRT trajectory (red dashed line) based on a heuristic trajectory (blue dotted line), where the green tree represents the RRT tree, the yellow tree represents the rewired tree, and the egg-shaped gray blocks represent the moving pedestrians.

Risk-DTRRT-Based Optimal Motion Planning Algorithm for Mobile Robots

- This paper proposes a method to improve the quality of the path generated using time-based RRT in a dynamic environment. The algorithm is proven to return a dynamically feasible homotopy optimal path for the RRT generated path.
- The algorithm is divided into two major components, LoS control checking algorithm which calculated the dynamically feasible line of sight path and a rewiring algorithm which uses the LoS algorithm to calculate the homotopy optimal path.

W. Chi, C. Wang, J. Wang and M. Q. . -H. Meng, "Risk-DTRRT-Based Optimal Motion Planning Algorithm for Mobile Robots," in *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1271-1288, July 2019, doi: 10.1109/TASE.2018.2877963.

Risk-DTRRT-Based Optimal Motion Planning Algorithm for Mobile Robots paper proposes a computationally effficient optimization approach for mobile robots. The proposed method first calculates a path using RRT algorithm which is then optimized using a LoS control checking algorithm. The author also proves that this algorithm returns a homotopy optimal path. This algorithm is less expensive in terms of computation and can be used for multi agent systems.

Proposed Methods

We have implemented the following Algorithms:

1. Priority Based
   1. CBS using Line of Sight (LoS).
   2. CBS using Motion Primitives.
2. Heuristic Based
   1. Online CBS using Motion Primitives

We have implemented the following Algorithms:

1. Priority Based
   1. CBS using Line of Sight (LoS).
   2. CBS using Motion Primitives.
2. Heuristic Based
   1. Online CBS using Motion Primitives

Implementation – LoS

- This is a priority-based algorithm in which first the A* path is calculated using CBS with prior robots' paths.
- The calculated paths are then smoothened and optimized in the safety set to follow non holonomic constraints. LoS optimization was used instead of MPC optimization as done by state of the art.
- LoS is proven to return homotopy optimal path and is less computationally expensive than MPC.

Algorithm 1 Prioritized CBS based A* using LoS

Input: The *map*, *start* nodes, and *end* nodes of all the robots in priority order
Output: Non-holonomic constraint feasible trajectories

```
computed_paths = []
for i in all robots do
initial_path = CBS_A*_path(start[i],end[i],computed_paths)
possible_start = start[i]
next_end = 0
L_last = 0
while end[i] not reached do
  L = LoS_path(possible_start, initial_path[next_end])
  if L is collision free then
    next_end = initial_path[next_end + 1]
    L_last = L
  else
    actual_path.append(L_last)
    possible_start = initial_path[nextend − 1]
  end if
end while
actual_path.append(L_last)
computed_paths.append(actual_path)
end for
Return computed_paths
```


- Our first implementation is CBS using LoS approach. This is a priority-based algorithm in which first the A* path is calculated using CBS with prior robots' paths. The calculated paths are then smoothened and optimized in the safety set to follow non holonomic constraints. LoS optimization was used instead of MPC optimization as it is proven to return homotopy optimal path and is less computationally expensive than MPC.

Implementation - A* Non holonomic

- This is also a priority-based approach which infuses A* algorithm with motion primitives.
- For each node the algorithms chooses an action from the sets of motion primitives and validates those action for collision avoidances and boundary conditions
- This path search is computed in a priority order and the computed paths of high priority robots are considered as collision points in CBS to ensure safe collision free trajectories.

Algorithm 2 Prioritized CBS based A* with motion primitives

Input: The *map*, *start* nodes, *end* nodes of all the robots in priority order, and motion *primitives*

Output: Non-holonomic constraint feasible trajectories

computed_paths = []

for i in all robots **do**

path = *get_path*(*start*[*i*],*end*[*i*],*computed_paths*,*primitives*)

computed_paths.append(*path*)

end for

Return *computed_paths*

For our next algorithm we integrate the conflict based A* search with motion primitives that are predefined considering non holonomic constraints. For each node the algorithm chooses an action from the sets of motion primitives and validates those action for collision avoidances and boundary conditions. This path search is computed in a priority order and the computed paths of high priority robots are considered as collision points in CBS to ensure safe collision free trajectories.

Implementation with the Heuristic

Novel heuristic-based approach:

- This algorithm first decomposes the configuration space into discrete set.
- Then for each agent, it calculates the path cost of all discrete points in the set to the goal using BFS and defined motion primitives.
- Finally, all agents greedily follow the path which minimizes the sum of BFS cost and distance heuristic with all the robots.

Algorithm 3 Heuristic CBS with motion primitives

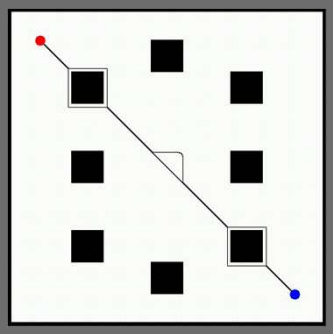
Input: The *map*, *start* nodes, *end* nodes of all the robots, and motion *primitives*
Output: Non-holonomic constraint feasible trajectories
$goal\_distance = []$
for i in all robots **do**
$distance\_map = BFS\_costs(end[i], map, primitives)$
$goal\_distance.append(distance\_map)$
end for
$heuristic\_function = (\frac{8}{distance})^3$
$reached = False$
$computed\_paths$ = []
$current\_pose = start$
while *not reached* **do**
  $priority = goal\_distance[current\_pose]$
  $robot\_order = argsort(priority)$
  for i in $robot\_order$ **do**
  $next\_pose = Greedy\_NH(i, current\_pose, Heuristic)$
  $current\_pose[i] = next\_pose$
  end for
  $reached = check\_reached(end, current\_pose)$
  $computed\_paths$.append($actual\_path$)
end while
Return $computed\_paths$

9

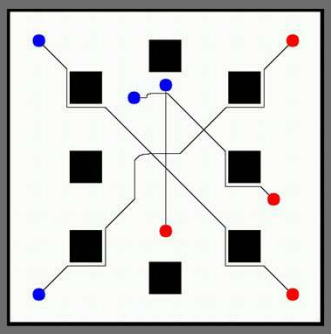
Worcester Polytechnic Institute

For the heuristic based implementation we created an online CBS algorithm which also uses motion primitives. This algorithm first decomposes the configuration space into discrete set. Then for each agent, it calculates the path cost of all discrete points in the set to the goal using BFS and defined motion primitives as it can be seen in the first for loop from the psuedo code on right. After that all agents greedily follow the path which minimizes the sum of BFS cost and distance heuristic with all the robots.

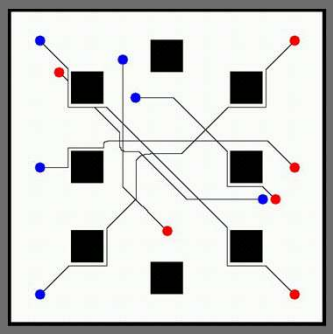
Results in Visualizer (Without Non holonomic constraints)



2 Agents



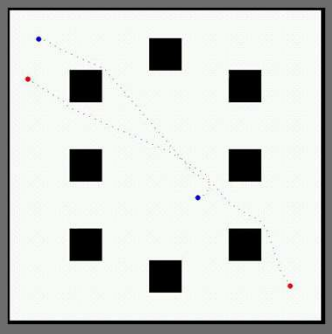
4 Agents



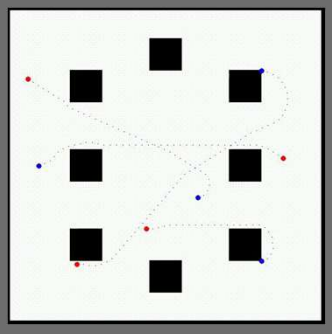
6 Agents

Here are the prelimnary results for the implementation without non-holonomic constraints for 2,4, and 6 agents. As observed from the visualizer the agents manuveur in the environment from a given start point to a goal point without colliding with obstacles and other agents

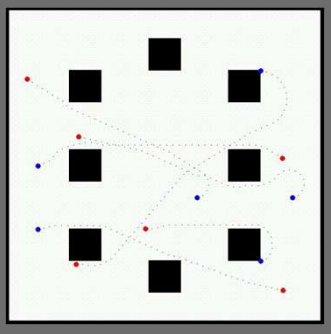
Results in Visualizer (With Non holonomic constraints)



2 Agents



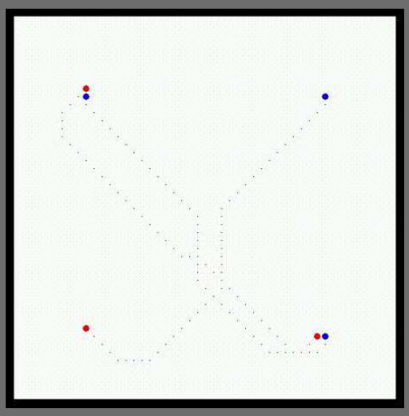
4 Agents



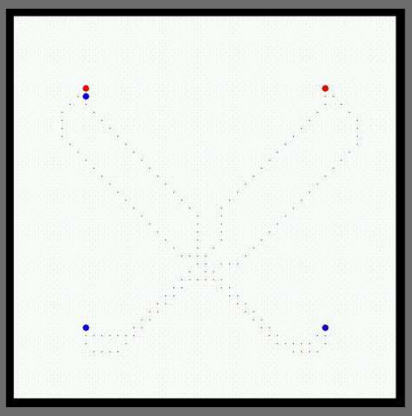
6 Agents

Here are our results for the implementation with non-holonomic constraints for 2,4, and 6 agents. As observed here the agents follow a non-holonomic trajectory from start point to a goal point without colliding with obstacles and other agents

Results for novel Heuristic based approach



3 Agents



4 Agents

Next we have the results for the trajectory generated by novel Heuristic based algorithm for 3 non-holonomic robot in an empty grid environment as evident from the figures all the agents take equivalent amount of time as opposed to the priortity based implementation seen in the previous slides.

Simulation Platform - ARGoS

ARGoS is a *multi-physics* robot simulator. It can simulate large-scale swarms of robots of any kind, efficiently.

Why ARGoS?

- **Speed:** ARGoS is generally faster than Gazebo because it is designed specifically for swarm robotics simulations
- **Modularity:** ARGoS has a modular architecture, which means that researchers can easily add new modules to the simulator to customize it for their specific needs
- **Scalability:** ARGoS is highly scalable, which means that it can handle simulations with a large number of robots without sacrificing performance.
- **Behavior-based approach:** ARGoS is based on a behavior-based approach to robot control, which means that each robot is controlled by a set of behaviors that can be programmed by the user.

13

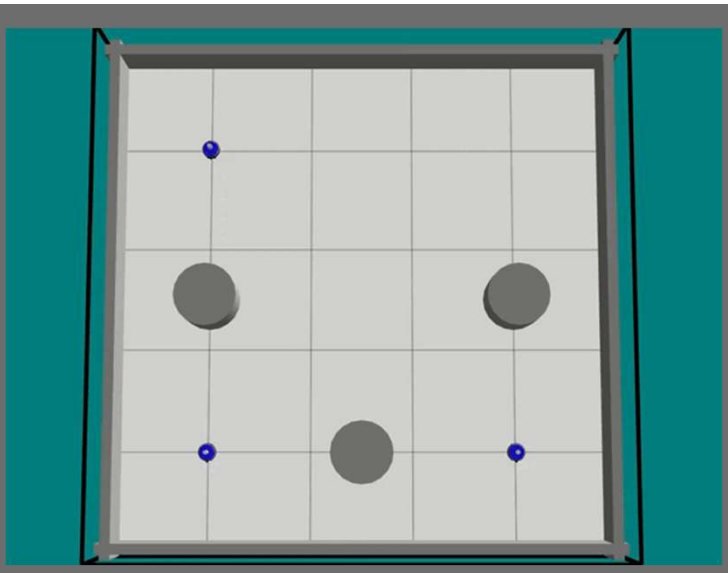
Worcester Polytechnic Institute

ARGoS and Gazebo are both powerful simulators used for robotics research and development. Each of them has its own advantages and disadvantages, depending on the specific use case. Here are some advantages of ARGoS simulator over Gazebo:

- Speed: ARGoS is generally faster than Gazebo because it is designed specifically for swarm robotics simulations. This means that ARGoS is optimized for simulations that involve large numbers of robots, while Gazebo is more general-purpose and therefore less efficient in this regard.
- Modularity: ARGoS has a modular architecture, which means that researchers can easily add new modules to the simulator to customize it for their specific needs. Gazebo is also modular, but ARGoS is designed specifically for swarm robotics, so it may be easier to work with for certain types of research.
- Scalability: ARGoS is highly scalable, which means that it can handle simulations with a large number of robots without sacrificing performance. Gazebo is also scalable, but may not be as efficient as ARGoS for very large-scale simulations.
- Behavior-based approach: ARGoS is based on a behavior-based approach to robot control, which means that each robot is controlled by a set of behaviors that can be programmed by the user. This allows researchers to experiment with different behaviors and see how they interact with each other in a swarm setting. Gazebo is more general-purpose and does not have this specific focus on behavior-based control.

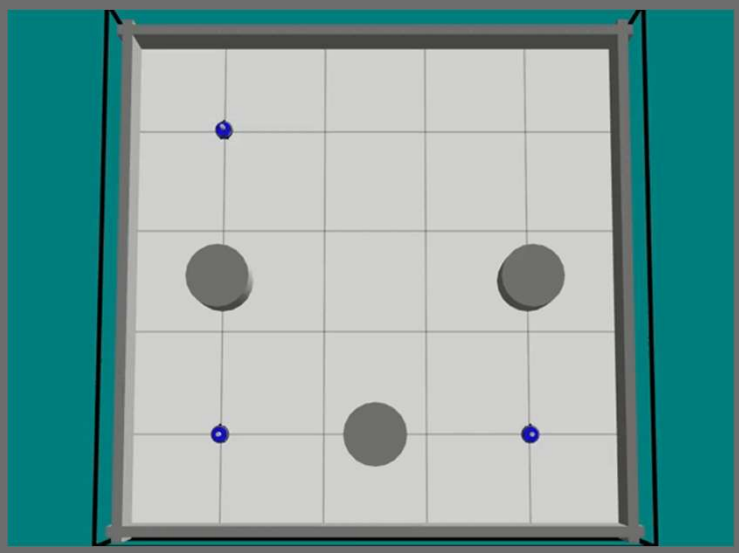
Overall, ARGoS is a great choice for swarm robotics research, while Gazebo is better suited for general-purpose robotics research. However, the choice of simulator ultimately depends on the specific needs of the project.

Results – Prioritized



Here we present the implementation of a priority based algorithm for 3 agents in the ARGoS environment.

Results – Heuristic



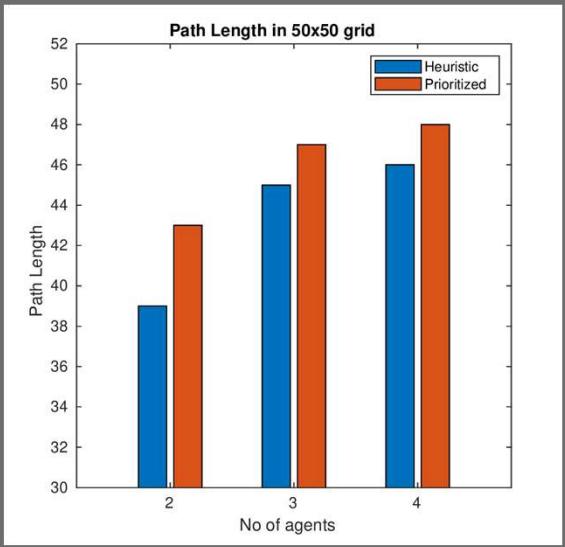
Next we have the simulation results for the heuristic based approach in the same setup.

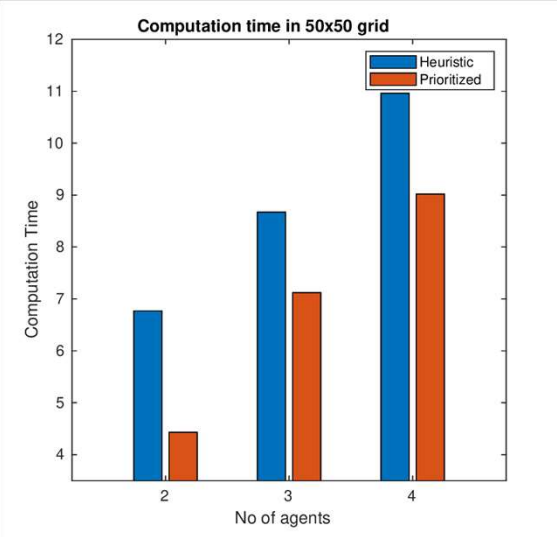
Comparison

Metric	Priority	Heuristic
Computation	Centralized	Decentralized
Mode of Execution	Offline	Online
Completeness	Not Complete	Not Complete
Optimality	Approximately Optimal	Approximately Optimal
Computation Cost	Less than Heuristic Based	Depends on Map Size
Parameter Tuning	Orientation Cost Parameter	• Distance Heuristic Choice and Parameters • Orientation Cost Parameter
Dynamic Obstacle	Can not handle	Can Handle

Here we present a comparison between Priority and Heuristic based algorithm. The Heuristic based algorithm design is independent of other agents, therefor decentralized whereas Priority based algorithm requires path of high priority agents, and must be a centralized implementation. For this reason, Heuristic based paths can be calculated online while execution whereas Priority based path needs to be calculated offline before execution. In terms of completeness and optimality both the algorithms are not complete and approximately optimal. Computation cost of Heuristic based algorithm is dependent on map size since, it requires a distance computation of all grid cells whereas Priority based algorithms are fast and computationally efficient. Heuristic algorithm performance is dependent on heuristic parameter tuning which is not required in priority based algorithm. On the other hand, the design of heuristic algorithm allows it to handle dynamic obstacle, which is not configured in priority based algorithm.

Evaluation





This is what we have observed through our experiments. As expected Heuristic algorithm takes more computation time than the prioritized algorithms. The difference was expected to grow with the number of agents but as discussed in previous slide it is dependent on the map size as well. Also for the path lengths of two categories, Heuristic paths are heavily dependent on the heuristic choice and tuning, and can outperform the priority algorithm as presented in the graph.

Limitation and Conclusion

- The Heuristic based algorithm is a proof of concept to the present non-optimality of the priority-based (SOTA) algorithm.
- Although this algorithm is scalable it does not guarantee completeness and is approximately optimal.
- There is still research going on to develop a scalable, optimal, and complete algorithm.

- The Heuristic-based algorithm serves as evidence of the limitations of the current state-of-the-art (SOTA) priority-based algorithm. While the priority-based algorithm is scalable, it falls short in terms of optimality. On the other hand, the Heuristic-based algorithm, while scalable, cannot guarantee completeness, and is only approximately optimal. These shortcomings have highlighted the need for continued research and development to create an algorithm that is both scalable and optimal, while also guaranteeing completeness.

Timeline and Work Distribution

TABLE I

Timeline	Task	Task allocation
Week 1	Literature Review and concept understanding.	Aadesh, Om, Vishrut
Week 2	Literature Review and State of the art simulation setup and code study.	Aadesh, Om, Vishrut
Week 3	State of the art setup error issues and new gazebo simulation setup.	Om
Week 3	Code implementation and visualizer setup	Vishrut, Aadesh
Week 4	Proposal. slides, Video creation	Aadesh, Om, Vishrut
Week 5	Spring break	
Week 6	Exploration of the optimization and smoothening methods.	Aadesh, Om, Vishrut
Week 6	Understanding motion primitives.	Aadesh
Week 7	Debugging the ROS simulation errors	Om,Aadesh
Week 7	Exploration of the optimization and smoothening method	Aadesh, Vishrut
Week 6/7	Setting up environment in ARGoS	Om, Vishrut
Week 8/9	A* implementation with motion primitives for MAPF.	Aadesh
Week 8/9	ARGoS simulation robot implementation	Om
Week 8/9	Implementation of optimization technique.	Vishrut
Week 9	Working and designing the heuristic function.	Om, Vishrut
Week 10/11	ARGoS code implementation with the new smoothen curve for 3 Robots	Aadesh, Vishrut, Om
Week 12/13	Final testing and simulation implementation	Aadesh, Vishrut, Om
Week 14	Final report and Documentation	Aadesh, Vishrut, Om

This is our planned task allocation and distribution of the project.

WPI

Thank You