Multi Agent Path Planning

Manideep Cherukuri, cheru050@umn.edu
Naga Hemachand Chinta, chint068@umn.edu
Joshua K. Jose, jose0159@umn.edu

December 14, 2022

Abstract

Path planning and efficient management within the limited resources is very significant. In a travelling salesman problem, a single agent moves from one city to another city and reaches back to the same city and the TSP algorithm helps to plan the optimal route to achieve the task. In multiple real-world applications, a single agent is not sufficient to execute all the task and even if the single agent is able to execute the performance could be very bad even after following an optimal path. Henceforth there is a need for the multi agent to execute the tasks in such applications. The biggest challenge with multi-agent is coordinating the agents and also trying to maintain the optimal solution. In this project our scope is only limited to two-agent co-ordination than finding the optimal path for the multi-agent.

1. Problem Description

Consider an E-commerce delivery as an application. Not every city has abundance of resources and challenges vary depending upon various factors. To name few major challenges, unavailability of delivery and pick up agents; not enough warehouses in few localities; customers staying far away from the warehouses and fulfilment centers; traffic jam and weather conditions. In all these aspects time is a major constraint and the route planned should not happen to make unnecessary double trips for the parcel delivery agent especially when there are other capable agents delivering the orders at the same time in the same path. This not only saves time but also the cost due to fuel consumption. This problem is particularly interesting because in the future when the robotic delivery agents come into the picture, these algorithms and approaches will be very much needed for the robot to co-ordinate with each other and execute the parcel delivery activities in a much better way.

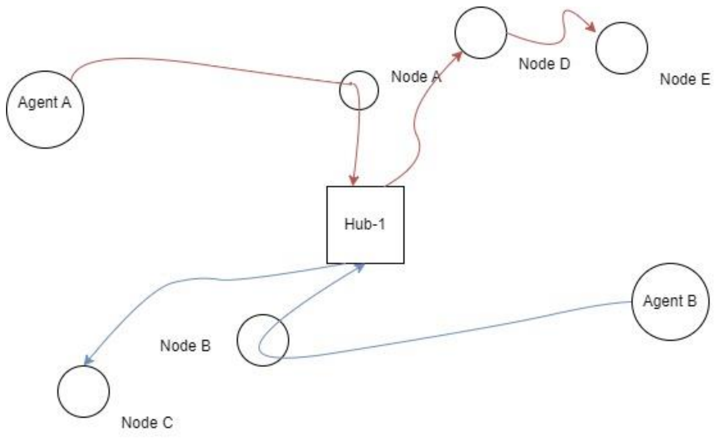

Figure 1: Basis model for transportation network.

There are only two delivery agents who need to deliver these orders at multiple points where each point could be treated an as end customer who ordered from that E-commerce company. Say the two delivery agents are delivery agent A and delivery agent B. Now delivery agent A has to deliver the parcels at node A, node D and node E and delivery agent B has to deliver the parcels at node B, node C and node E. We can see that both agent A and agent B are delivering the parcels at node E multiple times on the same day. Say there is a hub available between the path node B, node C and the path node A, node D, node E. Assuming that agent A has the capacity to carry the parcels that agent B is supposed to deliver at node E and also the agent A is nearby hub and about to reach the hub a little later than agent B reaching the hub. When agent B reaches the hub, it leaves the parcel that it is supposed to deliver at node E in the hub. Now when the agent A reaches the hub, it picks the parcel and continues its path and move towards its nearest next goals which is node D (assuming the agent A has already reached node A before reaching the hub) and deliver both the parcels which agent A was supposed to deliver and the parcel of agent B which was supposed to deliver at node E. This model is known as "Hub and spoke" model which is very popularly used in the logistics and E-commerce industry. To better understand the hub-and-spoke system, imagine an intricate bicycle wheel, with the hub as the strategic center of the network, and the spokes radiating out to connect it with remote points. The transportation and distribution industry were guided by the principles of point-to-point or direct-route operations. Transportation networks were disorganized, and shipping, aviation, and transit companies were losing money. As technology has developed, the logistics sector has found faster and more cost-effective ways of shipping freight. The hub-and-spoke model was born from industry's efforts to develop more efficient networks. In recent years, shipping companies have adopted the hub-and-spoke model to speed up deliveries and reduce costs. One more advantage of the hub-and-spoke model it improved shipment tracking. When packages are shipped directly from the source city, there are far too many routes connecting them to a vast network of other cities. Monitoring a shipment in such a complex system would be a logistical nightmare. Furthermore, since there would need to be mass coordination between cities, companies would need to spend money on staffing and properly equipping each facility. With the hub-and-spoke model, they only need to worry about one hotbed of shipping activity. In particular, private fleet companies use the hub-and-spoke model to employ a pool of non-dedicated drivers and trucks across a wider area. The non-dedicated resources can be used in other ways until clients in that region request their services. Larger-scale companies operate several hub-and-spoke systems, making the system slightly complicated to plan which coordinating multiple agents involved in the process.

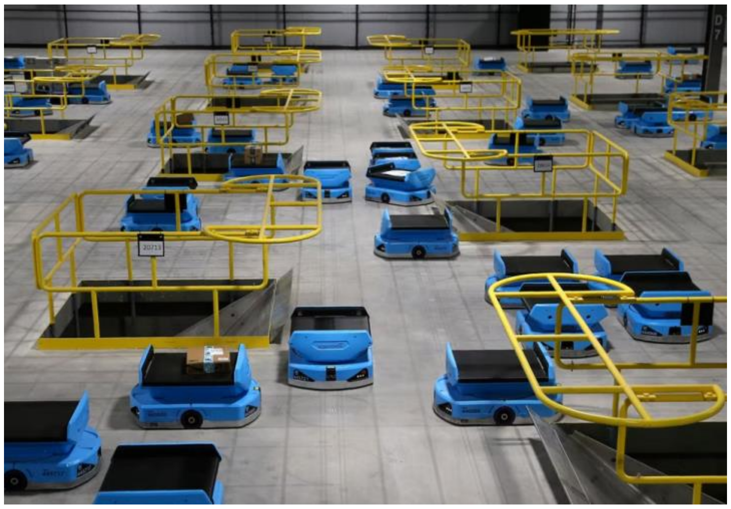

Figure 2: Co-ordination of multiple Amazon robotic transportation network in the warehouse [6].

Defining the problem in this project. The implementation in this project is an application of multi-agents in the warehouse such as Amazon robotics on the basis of figure 1. When the agents need to exchange items with each other they need to collide at a point. When they don't need to exchange the parcels, they follow their paths irrespective of the collision (path will be efficient if the collision is minimal) and when one of the agents have priority the other agent needs to wait at space while the prioritized agent is on its way.

2. Background

Multi-agent delivery problem can also be treated as a multi-agent travelling sales man problem. The research community has been working for a long time on these problems and there have been multiple developments since then but still many challenges persist. The “hub and spoke” model itself is not an ideal model because the major challenges with hub and spoke model arise exactly during the multiple irregular end points (end customer locations). This makes the hub and irregular spokes model in reality. Henceforth, there is a need to search and develop any other pattern that is much more efficient than the hub and spoke model despite the fact that practically still the hub and spoke model is the most practiced model in the industry. Let’s review the work in a sequence from various patterns practiced in the industry, research and development of technology in this field.

2.1 Transportation network models

There were few case studies which claim that “point-to-point” models have also performed well compared to the “hub and spoke” model in certain cases. A research case study [1] shows some data on Southwest airlines, the most well-known P2P (point-to-point) carrier, is used to call into question the standard theoretical models. A simple circular airline model is presented which predicts that any P2P carriers could capture up to 64 percent of the market share and have higher profits. The model implies that by pursuing P2P routes small firm entrants can successfully compete against the larger players in the airline industry.

Hub and spoke model have multiple advantages. In contrast to point-to-point, a Hub-and-spoke network allows consolidation of traffic flow at some special demand points, hubs, before reaching their destinations. This makes it economically more viable, through economies of scale mainly where travel demands are dispersed or uncertain. It also provides demand coverage within the network and facilitates transportation conditions and access to services. Hence, there are many applications of hub concept to transportation, telecommunication, logistics, etc. However, among them, public transport received least amount of attention. A research work [2] investigates the potential of hub network for public transportation planning in sparse travel and low network density conditions of Bamako. The work first diagnosed the major current issues and future challenges of public transportation planning. Then derived the projection for Bamako following its long-term development vision. Estimation of travel demand completes the later in order to confirm the existence of indices for hub application to transportation planning within Bamako. Then, a typology was proposed which is a potential hub network, star one, based on its advantage. Hub-and-spoke network had proved it efficiency in many fields such telecommunication, air transportation, computer science, logistics over decades. A deep diagnose of main factors underlying to public transport planning such as population projection and travel demand estimation, showed the deficiencies of current point-to point service and the need for more an efficient strategy to its planning. Therefore, hub structure should be an alternative. However, its success cannot be achieved without some good practices at: Institutional level: create one transit agency in the metropolitan level in order to coordinate decisions relative to fare, transit lines allocation and investments; Operational level: upgrade fleet and vehicles capacities able to meet growing demand by encouraging implementing Bus Rapid Transit or Light Rail; Organizational level: dedicate some routes or lanes for only transit in addition of regulation of unproductive competition by making operators complementary; Investments level: incentive practices being undertaken to discourage car-oriented investments such parking improvement or subsidization on fuel charges; Urbanization level: integrate land use and transportation planning.

Considering certain cons of Hub and Spoke model, a new model named “Hybrid Hub and Spoke” model is developed [3]. Hub-and-spoke transportation network can help Less-Than-Truckload carriers lower their total costs by consolidation transportation. In many cases, hybrid hub-and-spoke systems with some direct routes can be more cost-effective than pure ones. By means of analyzing a real case, this paper develops new mathematical models for a hybrid hub-and-spoke system, solving them using genetic algorithm with some strategies to determine vehicle types. At last, an average cost reduction of 2.68% is given and an optimal route planning result which forms a hybrid hub-and-spoke transportation network is described. A real case study about the design of hybrid hub-and-spoke transportation system of a carrier is studied. This company is an LTL motor carrier whose network covers all area in Shandong Province and two other cities of China. It has 15 depots and a hub in Shandong Province. The article demonstrates what cost reductions are possible by choosing suitable type of vehicles to transport the freight and additionally introducing direct transports between depots into the hub-and-spoke structure.

2.2 Algorithms used in path planning

Till now we have seen various models used in the transportation networks. Now we'll review few algorithms used in path planning. NP problems such as Travelling Salesman Problem don't have algorithms to provide optimal solution but there are sub-optimal or better algorithms which can do the job.

A greedy algorithm is a general term for algorithms that try to add the lowest cost possible in each iteration, even if they result in sub-optimal combinations. All possible edges are sorted by distance, shortest to longest. Then the shortest edge that will neither create a vertex with more than 2 edges, nor a cycle with less than the total number of node points is added. Although all the heuristics here cannot guarantee an optimal solution, greedy algorithms are known to be especially sub-optimal for the TSP.

The nearest neighbor heuristic is another greedy algorithm, or what some may call naive. It starts at one city and connects with the closest unvisited city. It repeats until every city has been visited. It then returns to the starting city. Nearest neighbor is a sub-optimal solution. The time complexity of the nearest neighbor algorithm is O(n^2). The number of computations required will not grow faster than n^2.

Insertion algorithms add new points between existing points on a tour as it grows. One implementation of Nearest Insertion begins with two cities. It then repeatedly finds the city not already in the tour that is closest to any city in the tour, and places it between whichever two cities would cause the resulting tour to be the shortest possible. It stops when no more insertions remain. The nearest insertion algorithm is O(n^2).

Like Nearest Insertion, Cheapest Insertion also begins with two cities. It then finds the city not already in the tour that when placed between two connected cities in the subtour will result in the shortest possible tour. It inserts the city between the two connected cities, and repeats until there are no more insertions left.

Random Insertion also begins with two cities. It then randomly selects a city not already in the tour and inserts it between two cities in the tour. Rinse, wash, repeat. Time complexity: O(n^2). Unlike the other insertions, Farthest Insertion begins with a city and connects it with the city that is furthest from it. Time complexity: O(n^2).

Christofides algorithm is a heuristic with a 3/2 approximation guarantee. In the worst case the tour is no longer than 3/2 the length of the optimum tour. Due to its speed and 3/2 approximation guarantee, Christofides algorithm is often used to construct an upper bound, as an initial tour which will be further optimized using tour improvement heuristics, or as an upper bound to help limit the search space for branch and cut techniques used in search of the optimal route. The algorithm is intricate [2]. Its time complexity is O(n^4).

A problem is called k-Optimal if we cannot improve the tour by switching k edges. Each k-Opt iteration takes O(n^k) time. It originates from the idea that tours with edges that cross over aren't optimal. 2-opt will consider every possible 2-edge swap, swapping 2 edges when it results in an improved tour. 2-opt takes O(n^2) time per iteration. 3-opt is a generalization of 2-opt, where 3 edges are swapped at a time. When 3 edges are removed, there are 7 different ways of reconnecting them, so they're all considered. The time complexity of 3-opt is O(n^3) for every 3-opt iteration.

Lin-Kernighan is an optimized k-Opt tour-improvement heuristic. It takes a tour and tries to improve it. By allowing some of the intermediate tours to be more costly than the initial tour, Lin-Kernighan can go well beyond the point where a simple 2-Opt would terminate. Implementations of the Lin-Kernighan heuristic such as Keld Helsgaun's LKH may use "walk" sequences of 2-Opt, 3-Opt, 4-Opt, 5-Opt, "kicks" to escape local minima, sensitivity analysis to direct and restrict the search, as well as other methods. LKH has 2 versions; the original and LKH-2 released later. Although it's a heuristic and not an exact algorithm, it frequently produces optimal solutions. It has converged upon the optimum route of every tour with a known optimum length. At one point in time or another it has also set records for every problem with unknown optimums, such as the World TSP, which has 1,900,000 locations.

Till now we saw some of the algorithms used majorly in single agent TSP problems. Though some of these algorithms can be used in multi-agent TSP or multi-agent path planning, the most challenging part of multi-agent path planning algorithms is the co-ordination between the multiple robots.

One of the approaches in solving a multi-agent path planning problem is “Evolutionary approach” [5]. Three main stages of evolutionary algorithm are initialization, reproduction and selection. Initialization is creating a starting population for which to evolve. Reproduction carries out evolutionary operators such as crossover, mutation and improvement heuristics to produce offspring. Selection taking individuals from both the main population and from the offspring to produce the new population. This approach is slightly different than the genetic algorithm. One more type of evolutionary approach is multi-demic evolutionary algorithm [5]. The aims of this algorithm are to exploit problem structure, aligning real-world implementation demands; Decentralized solution with Communication; Use multiple populations (or demes); With well-defined agent-population interactions.

There are search based MAPF (Multi-agent path finding) algorithms which are simple and used popularly to solve the multi-agent path planning problems [7]. There are all three types optimal, suboptimal but complete and suboptimal and incomplete algorithms in MAPF algorithms. A* based search algorithm, where the state space tree of the next states of the agents is created and then over the states space tree, A* algorithm is applied. BFS and DFS could also be applied over this states space. The tree could sometimes lead to be enormous in many real-time scenarios which is a kind of one disadvantage for A* algorithm because the number of states increases exponentially as the number of agents increases. Conflict based search algorithm where the optimal path is planned for each agent independently and unlike A* search algorithm, this algorithm allows one collision to happen. This algorithm especially useful when there is are parcels to transferred from one agent to another and they need to meet minimal number of times in an optimal plan. Priority based search algorithm, the algorithm plans for one agent after another in location time space in a given order. This algorithm is a suboptimal and incomplete solution.

One of the main challenges for our problem is managing coordination between the multiple agents. Koenig et al [7] suggested a coordination system based on sequential single-item auctions. The basic premise is that a team of multiple agents must visit a number of locations in an initially partially unknown environment. Because of the nature of the environment, it is advantageous to have an auction-based system where the agents can re-allocate the previously assigned locations based on new information discovered during the environment exploration. Although we are considering a fully known environment in our problem, we could account for possible obstacles like roadblocks or traffic jams which are dynamic in nature. Briefly, the auction system works like this: All locations are initially unassigned. Every agent bid on every location and whichever agent has the least path cost to a bid location wins that location. Then, this cycle is repeated for unallocated locations until all locations are allocated. Once this is done, each agent visits all locations assigned to it by following the shortest path possible. Based on analytical evaluation, it was seen that this method will give us a path that is 1.5 times the optimal path length in a fully known environment, which is a pretty good approximation.

3. Approach

3.1 Understanding the MAPF (Multi-Agent Path Finding)

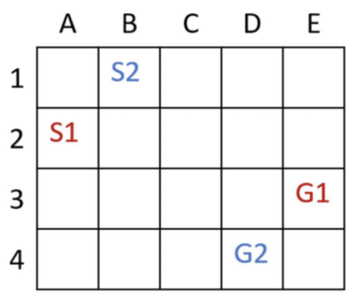

Figure 3: Example of a grid-based graph where the two different agents the red and blue respectively. The red agent starts at S1 (A2) and reaches goal at G1 (E3) while the blue agent starts at S2 (B1) and reaches goal at G2(D4).

Our Multi-Agent Path Finding (MAPF) problem has the task of calculating collision-free paths for a group of agents from their present locations to a predetermined destination [7]. Each agent can either move North, East, South, or West into any neighboring unblocked cell (provided an agent in that cell already exits that cell at the time the agent moves into it or earlier) or wait in its current cell.

3.2 Space Time A* Approach

The easiest approach is to just to think about what state it needs to be in. In other words, we need to know the current cells of each one of the agents. In the above Figure 3, we have two agents, so we need to have a pair of cells (the current cell of the red agent and the current cell of the blue agent).

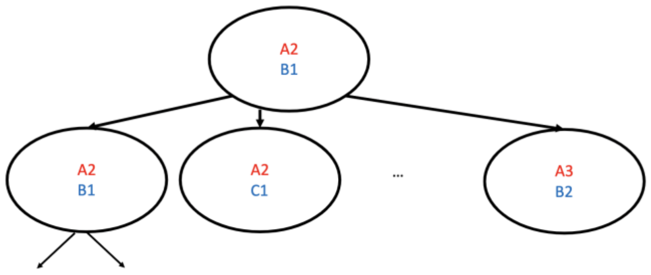

Figure 4: Example of a grid-based graph where the two different agents the red and blue respectively. The red agent starts at S1 (A2) and reaches goal at G1 (E3) while the blue agent starts at S2 (B1) and reaches goal at G2(D4).

In the initial state, the red agent is in its start cell A2 and the blue agent is in its start cell B1. For every single time step, the agents can either move North, East, South, West or Wait. For the red agent at the initial state, it can move North, East, South or Wait while the blue agent can move East, South, West or Wait. So there seems to be 15 possibilities (4 * 4 - 1) where we ignore the possibility where the red agent moves east and the blue agent moves south which leads to a collision. In a similar fashion, we compute all the possibilities at every time steps giving us a complete tree. We can use any search algorithms now like A* to traverse over the tree to find the optimal MAPF.

The drawback of this method is that every time steps, the state space is enormous. In case of hundreds or thousands of agents, the number of states increases exponentially in the number of agents. As a result, the runtime of such a A* based search is prohibitive.

3.3 Conflict Based Approach

Unlike the previous approach, we exploit the structure of the problem more to increase the efficiency of our search-based techniques leading us to the Conflict Based Search Approach. We let every agent plan independently. Initially, we let the red agent to find the best possible plan for itself disregarding what the blue agent does. Similarly, the blue agent is tasked to find the optimal path independently as observed below.

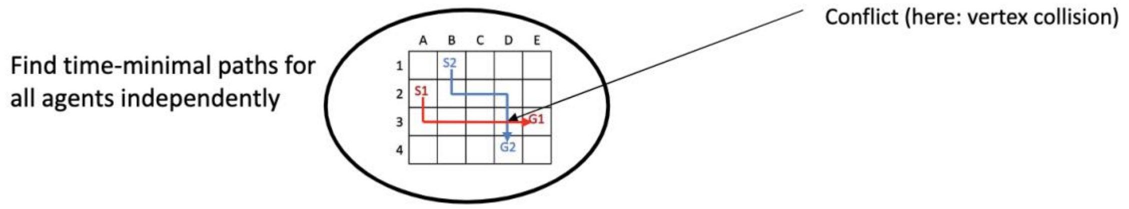
Find time-minimal paths for all agents independently
Conflict (here: vertex collision)
A
B
C
D
E
1
2
3
4
S1
S2
G1
G2

Figure 5: Optimal path plans for all agents independently where vertex collision is observed at D3 at time step 4.

We can observe that there is a collision at cell D3 between both the agents before they reach their respective goals. A simple way of going about this is to group all the colliding agents together into a team and plan for them jointly. This could work well but very often in difficult MAPF instances, we very quickly need to group all of the agents together and we are back to the A* approach as discussed earlier which doesn't work so well.

Given that there is a solution for both the agents to reach their targets, we know that there would be at least one collision at one point of time when both the agents meet. In the above Figure 5, there is one vertex collision observed where both of the agents are at the same time (time step 4) at same place (D3). Conflict based search here avoids this by making sure that at least one of the agents which are colliding at not meeting at the same location at the same point in time.

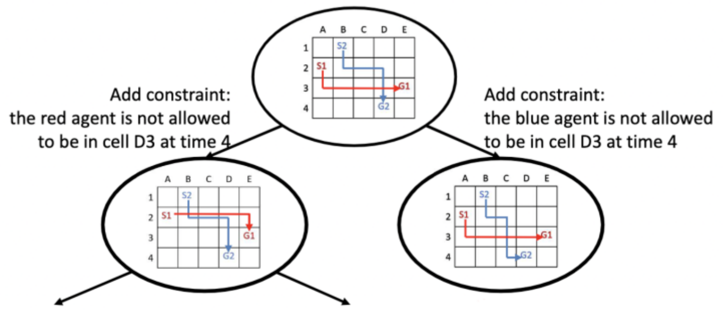
Add constraint: the red agent is not allowed to be in cell D3 at time 4
Add constraint: the blue agent is not allowed to be in cell D3 at time 4
S1
S2
G1
G2

Figure 6: Adding Constraints to one agent at a time to avoid collision.

Assuming the above scenario, we arrive at 2 different situations. The first one where we don't allow the red agent to be in cell D3 at time step 4 and the other where we don't allow the blue agent to be in D3 at time step 4. Looking closer at the left bottom situation of Figure 6 where both agents will be planning independently while red agent will be taking constraints also into account, we can observe that the blue agent will still be taking the best possible path since no constraints have been imposed on it. On the other hand, we can observe that there would be still be another vertex collision at B2 at time step 1. In a similar fashion, we apply the constraints again on one of the agents to not be at B2 at time step 1. Doing this recursively, we arrive at a similar tree where we can apply any search algorithm to find the best path possible.

The drawback of this approach is that we can only pick one collision at a time to resolve it. When we have multiple collisions, it becomes complicated to determine which collision to pick as it impacts the runtime of the algorithm.

3.4 Priority Based Approach

Though conflict-based search algorithm is much faster than a simple A* algorithm, it still has its limitations. Priority based planning approach overcomes this by using a sub optimal MAPF planner instead of an optimal MAPF planner. Not only is this very fast, it plans for one agent after the other in location-time space in a given order.

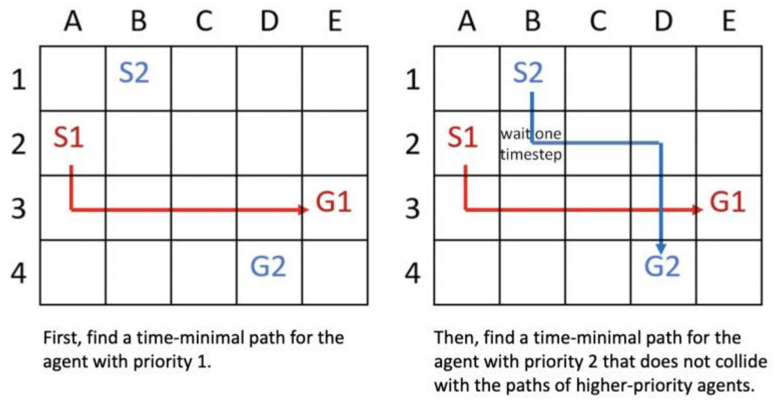
A B C D E
1 2 3 4
S2
S1
G1
G2
First, find a time-minimal path for the agent with priority 1.
wait one timestep
Then, find a time-minimal path for the agent with priority 2 that does not collide with the paths of higher-priority agents.

Figure 7: Adding Constraints to one agent at a time to avoid collision.

The underlying idea here is when we find a plan for an agent, it is not allowed to collide with the plans that we found already previously for other agents. Consider the above example in Figure 7 where we plan for the red agent first assuming it has a higher priority than the blue agent. The red agent is now tasked to find the best possible for itself but it can't collide with any of the previous plans which are zero is this case. Similarly, we move to the blue agent now where it finds the best possible path for itself making sure it is not colliding with the plans of the previous agents. In other words, it cannot collide with those of the reds plan. To overcome this collision, the blue agents wait for a time step at location B2 and then moves in its path.

3.5 Conflict Based Approach with Disjoint Splitting

If CBS (Conflict-Based Search) chooses to resolve a vertex collision where agents red and blue are both in cell location x at time step t, then it generates two negative vertex constraints, namely the negative vertex constraint 〈red, x, t〉 (that prohibits the red agent from being in cell x at time step t) and the negative vertex constraint 〈b, x, t〉 (that prohibits the blue agent from being in cell x at time step t). Similarly, if CBS chooses to resolve an edge collision where the red agent moves from cell x to cell y and the blue agent moves from cell y to cell x at time step t, then it generates two negative edge constraints, namely the negative edge constraint 〈red, x, y, t〉 (that prohibits the red agent from moving from cell x to cell y at time step t) and the negative edge constraint 〈blue, y, x, t 〉 (that prohibits the blue agent from moving from cell y to cell x at time step t).

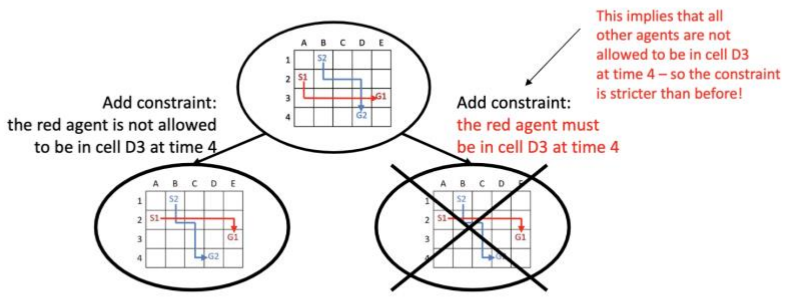
This implies that all other agents are not allowed to be in cell D3 at time 4 – so the constraint is stricter than before!
Add constraint: the red agent is not allowed to be in cell D3 at time 4
Add constraint: the red agent must be in cell D3 at time 4
S1
S2
G1
G2

Figure 8: CBS with disjoint splitting changing the second vertex collision.

CBS with disjoint splitting changes the second constraint in both cases. If CBS with disjoint splitting chooses to resolve a vertex collision, it changes the second negative vertex constraint that prohibits the blue agent from being in cell x at time step t to a positive vertex constraint that requires the red agent to be in cell x at time step t. On the other hand, if CBS with disjoint splitting chooses to resolve an edge collision, it changes the second negative edge constraint that prohibits the blue agent from moving from cell y to cell x at time step t to a positive edge constraint that requires the red agent to move from cell x to cell y at time step t.

In other words, CBS with disjoint splitting changes the second negative constraint (that prohibits the second agent from executing the colliding action) to a positive constraint for the first agent (that requires the first agent to execute the colliding action) in both cases as observed in Figure 8. It could also change the first negative constraint to a positive constraint for the second agent and thus can choose one of the colliding agents freely, for example, randomly.

The reason for this change is that the second constraints are now stronger (meaning more constraining). For example, if the red agent is required to be in cell x at time step t, then all other agents (including the blue agent) are automatically prohibited from being in cell x at time step t since they would otherwise collide with the red agent. Thus, the second positive vertex constraint of CBS with disjoint splitting implicitly includes the second negative vertex constraint of CBS. Thus, CBS with disjoint splitting can be expected to run faster than CBS but remains complete and optimal.

4. Experiments, Results and Analysis

Our objective in this experiment to execute the three conditions proposed in the problem definition above. We constructed a custom grid map of size 8 * 7 as the environment to test our algorithms. There will be two agents, one is agent 0 in green color and the another one is agent 1 in blue color. The initial states and the final goal states are shown in Figure 9. The agents are represented as circles and the destination locations are represented as squares.

4.1 Independent planning using A*

First, we compute the independent MAPF solver plans for all agents independently using A* search as the baseline model. Their paths do not collide with the environment but are allowed to collide with the paths of the other agents. Thus, there is a collision when the blue agent 1 moves towards its goal cell while the green agent 0 moves on top of it. In the Figure 9b, both agents turn red when this happens, and a warning is printed on the terminal notifying you about the details of the collision.

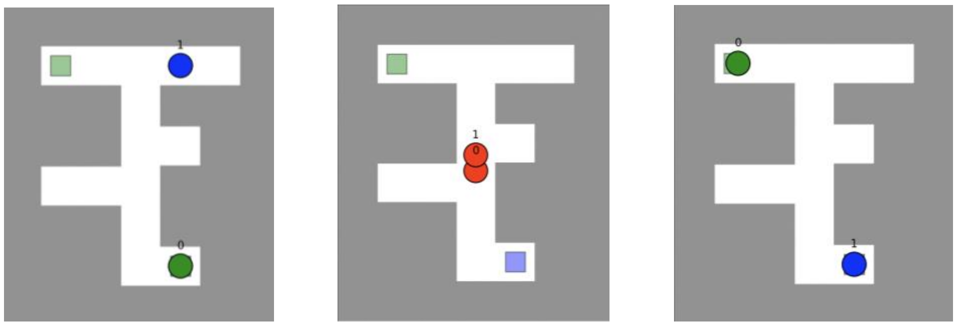

Figure 9: a) Initial state of the agents. b) Collision indicated by the agents turning red at that time step. c) Goal states of the agents.

4.2 Space Time A* Planning

Here, we change the single agent solver to perform a space-time A* search that searches in cell-time space and returns a shortest path that satisfies a given set of constraints. Such constraints are essential for further MAPF solvers such as prioritized planning and CBS. We observe an identical behavioral as above.

4.3 Prioritized Search Planning

The prioritized MAPF solver finds paths for all agents, one after the other, that do not collide with the environment or the already planned paths of the other agents as seen in Figure 10. To ensure that the path of an agent does not collide with the already planned paths of the other agents, the function defined for A* receives as input a list of (negative) constraints compiled from their paths.

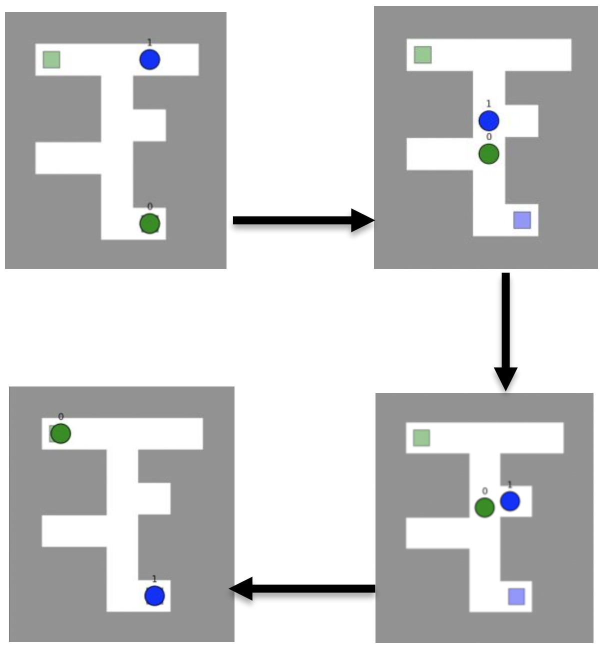

Figure 10: Action sequence of the Prioritized MAPF where the blue agent 1 makes way for the green agent 0 due to its high priority.

4.4 Conflict Based Search Planning

We detect collisions among agents, namely vertex collisions where two agents are in the same cell at the same time step and edge collisions where two agents move to the cell of the other agent at the same time step as observed in Figure 11.

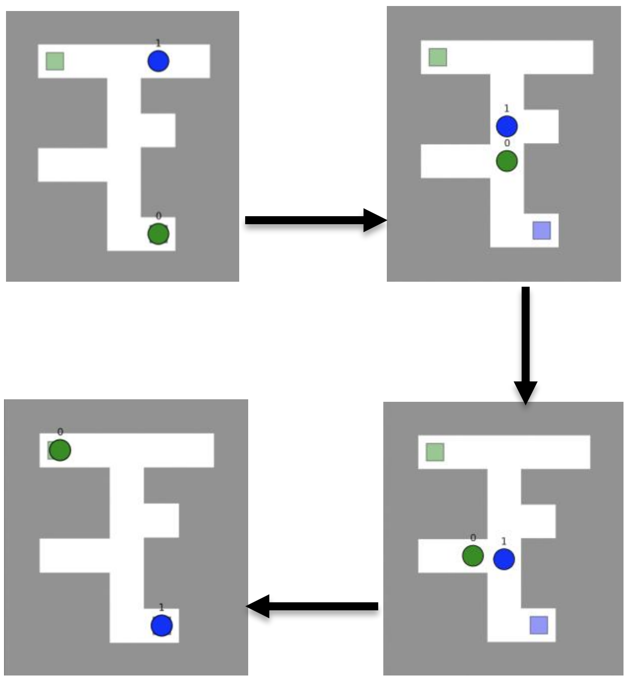

Figure 11: Green agent 0 moves left making way for the blue agent 0 due to one of the negative constraints prohibiting it from colliding.

4.5 Analysis of the performance benchmarks

In these benchmark [8], the map is a 20x20 matrix with obstacles distributed in the 5% of the map. The idea is to increase the number of agents (from 4 to 26 with step 2) and see if the algorithm can solve the problem in less than 5 minutes. For each number of agents, the same map is used for 25 times, but the start and goal positions are randomly distributed.

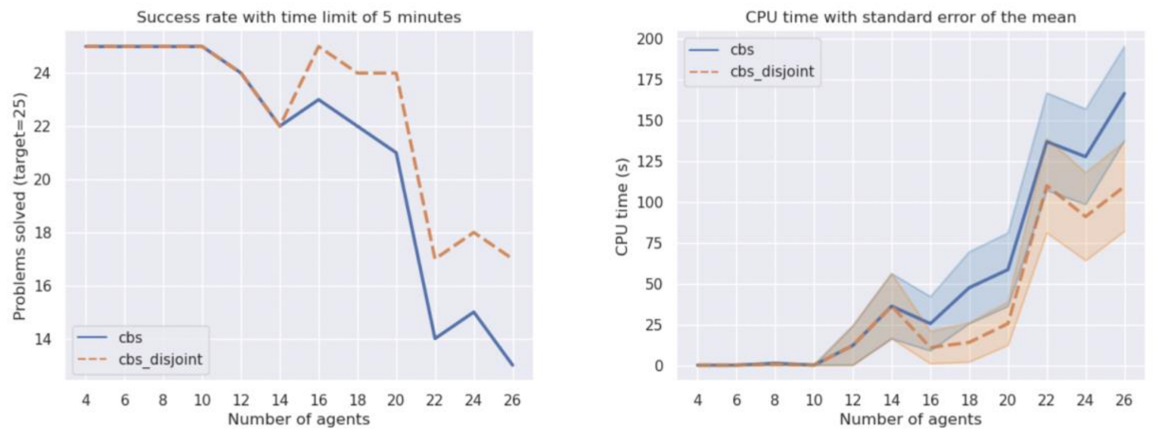

Figure 12: Performance benchmarks between the CBS and CBS-DS for multiple agents with 2 step increment over a common map.

As plotted in the above Figure 12, the algorithm CBS-Disjoint Split runs much faster than the CBS algorithm and the number of nodes expanded is smaller. So, the benchmarks verified that the CBS-DS algorithm seems to be more efficient than the CBS.

5. Conclusions

In this project, we have seen various research in the field of multi-robot path planning and various applications. E-commerce industry taking a highly significant leverage of this technology especially in the domain of transportation and logistics. We implemented MAPF based search algorithms namely A* based search algorithm, Conflict based search algorithms and priority-based search algorithm to generate various cases manifesting the behavior of the agents under various conditions proposed in the problem statement.

The configuration space for path planning is always huge, traditional path planning methods search path in figuration space point by point such as A* searching algorithm and Dijkstra algorithm, it is time consuming. Although modified searching method such as sparse A* (SAS) improved the planning speed to some extent, it is still difficult to meet the demands for real-time path planning. Genetic Algorithm (GA) might meet the requirement. GA is a kind of evolutionary algorithm that obtains optimal solution by iterative evolution of population [4]. In the iteration process of getting the optimal solution, plenty of suboptimal solutions are generated, some of which can be preserved as a choice of substitute if the suboptimal solutions were feasible paths. Henceforth, genetic algorithms can be a future scope of the current work in giving fast performance and optimal path.

.

6. References

[1] Ball, Christopher. (2022). Rethinking Hub versus Point-to-Point Competition: A Simple Circular Airline Model. Available:

https://www.researchgate.net/publication/242464525_Rethinking_Hub_versus_Point-to-Point_Competition_A_Simple_Circular_Airline_Model

[2] M. Telly and N. Zhang, "Implications and benefits of Hub-and-spoke network to public transport planning in large African cities with sparse travel demand case of Bamako," Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), 2011, pp. 126-131, doi: 10.1109/TMEE.2011.6199163.

[3] J. Zhang, Y. Wu and P. Liu, "Routing Problem for Hybrid Hub-and-spoke Transportation Network: a Case Study of a LTL Carrier," 2007 IEEE International Conference on Automation and Logistics, 2007, pp. 170-175, doi: 10.1109/ICAL.2007.4338551.

[4] S. Li, M. Ding, C. Cai and L. Jiang, "Efficient Path Planning Method Based on Genetic Algorithm Combining Path Network," 2010 Fourth International Conference on Genetic and Evolutionary Computing, 2010, pp. 194-197, doi: 10.1109/ICGEC.2010.55.

[5] Kent, T., & Richards, A. arXiv preprint arXiv:1906.05616\, keywords = Allocation and Routing,Decision Making,Distributed problem solving,Evolutionary Algorithms,Multi Agent Travelling Salesman, title = Decentralised Multi-Demic Evolutionary Approach to the Dynamic Multi-Agent Travelling Salesman Problem, url = http://arxiv.org/abs/1906.05616, year = 2019.

[6] Matt O'Brien. As robots slowly take over Amazon's warehouses, are they causing more harm than good? Available: https://www.independent.co.uk/news/business/robots-amazon-delivery-artificial-intelligence-technology-a9264036.html

[7] http://idm-lab.org/project-p/project.html

[8] http://mapf.info/index.php/Main/Benchmarks

Member Contributions

Manideep Cherukuri

Manideep made significant contributions to methodology, designing the algorithms for the problem-solving strategy, and implementing a variety of algorithms, including Space Time A*, conflict-based algorithms, conflict-based algorithms with and without disjoint splits, and developing benchmarks for analysis.

Naga Hemachand Chinta

Hemachand's contributions included working on the background research on various models, frameworks and algorithms used in single agent and multi-agent path planning problems and various patterns in the transportation network. Worked on the priority-based search algorithm in the problem approach.

Joshua K. Jose

Joshua contributed in drafting the first iteration of the report and researched various algorithms needed for the multi-agent co-ordination. He contributed towards the background research and worked on the simple A* based search algorithm in the problem approach.