Multi-agent Path Finding with Optimal Task Assignment

[redacted]

I. Introduction

An optimal and complete approach to solving the centralized multi-agent path finding problem (MAPF) with coupled Task Assignment (TA) is implemented in this project. Given N agents tasked with moving N identical packages from pickup to delivery locations, the aim is to compute an optimal pickup and delivery assignment for each agent, and optimal collision-free paths for each agent from start to pickup to delivery.

Applications of this problem include but are not limited to fully-autonomous fulfillment centres like the ones developed by Amazon Robotics, food delivery route+task-planning for Grubhub and Doordash and automating container loading and unloading in shipyards.

We build upon an approach from [1] and deploy conflict based search coupled with optimal task assignment (CBS-TA) to solve the MAPF problem in this project.

II. Problem Definition

Consider a graph G = (V, E, D) where $v \in V$ are vertices that correspond to locations and $e \in E$ are unit edges and $d \in D$ are diagonal edges connecting these vertices. Edges are indicative of a direct connection between two vertices. Diagonal edges have a higher penalty as compared to unit edges because a greater distance (1.4 times a unit edge) is traversed. We have N agents each represented by a_i where $i \in I$ and $I = \{1, 2, ...N\}$. They start at locations $s_i \in V, i \in I$. We are given locations for pickup $p_i \in V, i \in I$ and locations for delivery $d_i \in V, i \in I$.

The goal is to generate a path $p_i = [v_0^i, v_1^i ..., v_{T_i}^i]$ with cost C_i for each agent such that:

1) Each agents starts at its given start location
2) Each agent ends up at one of the potential goal locations and stays there until all the other paths are executed
3) Each agent moves either along an edge or stays in the same position in a unit time-step
4) The set of paths generated is free of vertex, edge and diagonal collisions

The set of paths will be considered optimal if they are consistent with the following objective functions:

$$V_1 = \min \sum_{i \in I} C_i \text{ , } V_2 = \min \sum_{i \in I} T_i \qquad (1)$$

The task of multi-agent pickup and delivery is tackled in the following two ways in this project:

A. CBS-TA for Pickup & Delivery Independently

The first method is to run CBS-TA independently for the pickup and delivery tasks which would imply that the robots have to wait at the pickup location until every robot has reached their corresponding pickup locations to plan for the delivery task. This method has the advantage that the conflict based search is run on paths of short lengths and hence more likely to find a conflict-free path within the given time. However due to the agents waiting at the pickup locations unnecessarily, it is bound to be sub-optimal in terms of time taken for delivery of individual agents. CBS-TA is used for pickup and delivery tasks separately while building a search forest for finding the optimal assignment considering conflicts.

B. CBS for Coupled Pickup & Delivery with Independent TA

Another method we implemented to solve the pickup and delivery problem is to do a conflict based search on the coupled path for pickup and delivery. In this case, the task assignments are done once initially for pickup and delivery and not changed during the search considering the conflicts. This method has the advantage that the agents need not wait for other agents after reaching their respective pickup locations. However, the path lengths are longer and conflict based search might not terminate in time. Since the assignments are done only once, it is also possible that a conflict-free path is not possible for the given assignment.

III. Proposed Approach

CBS-TA is a three tier problem: 1) Task allocation 2) Conflict based search 3) 3D search (x,y,t). The conventional approach would be to solve all three levels in a decoupled manner. But different task assignments could result in different solutions at the mid level and low level and the problem as a whole is no longer decoupled.

A. High level search: Task Assignment (TA) (Guru)

Given N agents and N tasks with C representing the cost matrix such that C_{ij} representing the cost for agent i to perform task j, the task assignment problem pertains to assigning each agent with a unique task such that the overall cost $\sum_i C_i$ is minimized and all tasks are assigned to atleast one agent. This is a combinatorial optimization problem which can be solved in $O(n^4)$ using the Hungarian method. For the CBS-TA algorithm, the unconstrained task assignment is performed initially to get an initial path for each agent. In the course of the CBS-TA algorithm, when conflicts are encountered, the algorithm checks for the next best assignment to build another CBS search tree by cleverly constraining the existing assignment. The constrained assignment is solved using the Hungarian method using a modified cost matrix where the $C_{ij} = 0$ for constraints enforcing a certain assignment and $C_{ij} = \infty$ for constraints restricting a certain assignment.

B. *Mid level search: Conflict based search (CBS) (Aarati)*

CBS-TA at the mid level promotes the generation of forests of binary trees also known as constraint trees (CT). A constraint is of the form (a_i, v, t) where a_i is an agent, v is a vertex and t is a time. Basically, agent a_i is not allowed at vertex v at time t. Each node in the constraint tree holds the following information:

1) **Root**: It states whether or not the node is the root of a tree
2) **Task assignment**: It represents the tree it belongs to.
3) **Set of constraints**: The root of the CT contains an empty set of constraints. The child of a node in the CT inherits the constraints of the parent and adds one new constraint for one agent.
4) **Solution**: A set of paths, one for each agent consistent with the constraints for the agent returned by the low level search.
5) **Cost**: The sum of costs of all the paths.

The initial tree is created using task assignment from the initial high level search. The first node is created using the solution from an unconstrained search. A **conflict** check is conducted and it returns the first conflict in the set. A conflict between two agents can be can be either a diagonal, edge or in place collision between two agents. Conflict resolution is attempted by adding two successor nodes in this mid-level search tree. This corresponds to introducing an additional constraint for each agent participating in the conflict at the lower level. Synchronously a new tree is created. These three nodes are then pushed to an OPEN list and the node with lowest cost is chosen for expansion.

C. *Low level search: 3D search (x, y, t) (Roshan)*

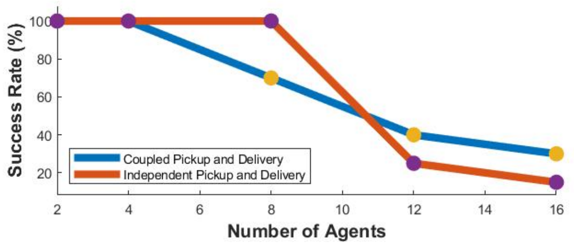
On the motion-planning level, there are primarily two types of searches that are required by the CBS - an 2D A* search when no constraints are passed, and a 3D A* (x, y, t) when time-based constraints are passed. Slightly different approaches were taken when the problem is restricted to a single delivery run from a start location as compared to the more general case when (potentially multiple) pickup and delivery maneuvers are planned together in one shot.

For the simplified case of motion-planning for N agents to N delivery points, firstly a backwards 2D-Dijkstra expansion is run from each goal to all points in the grid-map. This backwards search is used for (a) populating the cost matrix used for task assignment, (b) providing a search tree to enable back-tracking in the 2D unconstrained A* search, and (c) guiding the 3D constrained A* search as an admissible and consistent heuristic. The constrained search finds a path through time by avoiding all the collision constraints passed to it by CBS. The search is constrained to terminate only when the agent reaches its destination and the end-time of the path is greater than or equal to the time associated with each constraint.

In the general case of multiple pickup and delivery runs, the motion-planning problem is posed as a waypoint-surveyal problem for each agent. The state of the constrained 3D search problem is augmented by considering how many waypoints (pickup and delivery points) have already been visited. The heuristic calculation again utilizes the backward Dijkstra expansions - this time from all the waypoints - to estimate what the cost-to-go for a particular agent state (location + waypoints visited) would be if there were no collisions with the other agents. Eventually, the search is terminated when the agent reaches its final goal after visting all waypoints, and the end-time of the path is greater than or equal to the time associated with each constraint.

IV. Experimental Results

A series of experiments have been conducted. A set of 2D maps have been generated with randomized obstacles, starts, pick-up and delivery locations. Obstacle density is set to 10 percent of the map's area. The success rate of finding a plan within 30 seconds and time taken for finding a solution for the pickup and delivery tasks is plotted for randomized 32x32 maps.



Planning Time	2 agents	4 agents	8 agents	12 agents
8×8 *map*	0.001 sec	0.003 sec	0.315 sec	–
16×16 *map*	0.007 sec	0.020 sec	1.550 sec	4.166 sec
32×32 *map*	0.127 sec	0.080 sec	0.106 sec	0.144 sec

V. Conclusions and Future Scope

This work has successfully deployed the approach in [1] and has demonstrated efficacy in solving the MAPF PROBLEM with optimal task assignment. This approach is efficient for moderate number of agents and a dense environment as supported by a wide range of experiments. The Hungarian method adopted in this study can easily be extended to a case where the number of agents and number of goals is not equal. Conflict based search with A* is complete and optimal with respect to the sum of the costs of all agents. As mentioned in section II-B, in this implementation we don't re-plan task assignment based on conflicts that arise in the coupled delivery and pickup case. This implementation is left to the future scope of the project. Additionally, when scaling up to larger systems, an enhanced conflict based search (ECBS-TA) implementation may be more feasible. This approach is complete and boundedly sub-optimal, and thus may be implemented to speed up the search by accounting for the number of conflicts as a guiding heuristic.

References

[1] Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Conflict-based search with optimal task assignment. In *AAMAS*, 2018.