CL-MAPF: Multi-Agent Path Finding for Car-Like Robots with Kinematic and Spatiotemporal Constraints (Replication Study)

Geethika Hemkumar, EID: gh22885
Avani Agarwal, EID: aa88539
Department of Computer Science
The University of Texas at Austin

1 INTRODUCTION

Multi Agent Path finding (*MAPF*) can be described as a planning problem for multiple agents. Usually, the start and goal states are defined for agents and they should try to reach the goal state without collisions. MAPF can be applied to several areas and has broad applications in industries like self-driving cars, autonomous straddle carriers, warehouse robots, unmanned surface vehicles, office robots and various other fields. It is known to be an NP hard problem [17]. Approaches to MAPF can be categorized into reduction-based [3, 14, 19], A*-based [5, 10, 13, 16], prioritized [2, 7] and dedicated search-based algorithms [1, 4, 12]. The existing solvers of MAPF assume agents as holonomic agents that move in cardinal directions and rotate in their place. They also neglect the agent size and spatiotemporal constraints. This is in contrast to real-industry robots that are usually non-holonomic, rectangular in shape and have minimum turning radii. Consequently, these solvers cannot accurately model real-world multi-agent scenarios with car-like agents.

In [11] Licheng et al. give the problem definition for Car-Like MAPF or CL-MAPF and introduce CL-CBS to solve the CL-MAPF problem. The CL-MAPF problem definition is as follows:

For a workspace $\mathbf{W} \subset \mathbf{R}^3$ and $\mathbf{O}_{ws} \subset \mathbf{W}$ where $\mathbf{O}_{ws}$ represents the set of obstacles that occupy a region [17]:

(1) The paths of N car like agents $a_1 ... a_n$ should begin at their start state and end at their goal state in a finite number of time steps.
(2) The agent remains at its goal position after reaching it.
(3) Agent should not collide with any obstacle or other agents at any time step t.
(4) Ackermann steering agents should satisfy the kinematic model. That is, the agents velocity v is bounded as $v_{bmax} \leq v \leq v_{fmax}$ where $v_{bmax} < 0$ and represent the velocity of moving backwards and $v_{fmax} > 0$ and represents the velocity of moving forwards. Also, the agents maintain a minimum turning radius r_{min} during the whole path [17].

Licheng et al. in [11] implement a novel two-level optimal solver called Car-Like Conflict-Based Search(CL-CBS) that uses a binary body conflict search tree and a spatiotemporal hybrid-state A* method for low level planning that outperformed the base models CBS-MPC [12] and the HA* solvers. The CL-CBS planner can be broken down into two steps:

(1) Body Conflict Tree
(2) Spatiotemporal HA*

The authors also give a sequential CL-CBS solver to reduce the high level search time.

As a note, we use the terms workspace and map interchangeably.

1.1 CL-CBS: Body Conflict Tree

The body conflict tree is a variant of the conflict tree used in the original CBS paper [12]. Classical MAPF solvers use edge and vertex conflicts for collisions between two agents and hence fail to account for all collision scenarios. As CL-MAPF has a continuous workspace, it uses a best first search on a binary body conflict tree where each node contains a set of inter-agent constraints and the solution satisfying these constraints. To expand the BCT, the leaf node with minimum cost is popped and a collision check is done for the solution that belongs to the node [11]. A body conflict can be denoted using the tuple $< a_i, a_j, C_t^i, C_t^j, t >$ where a_i, a_j are the two agents at timestep t and C_t^i, C_t^j is body rectangle for a_i at t. For a body conflict, two nodes are produced with the constraints $< a_i, C_t^j, [t - \delta_T, t + \delta_T] >$ and $< a_j, C_t^i, [t - \delta_T, t + \delta_T] >$. These denote that the agent should not cross the rectangle area of the other agent through the time step $t - \delta_T$ to $t + \delta_T$. Spatiotemporal HA* is performed for low-level path finding.

Figure 1 describes the pipeline for the BCT for CL-CBS.

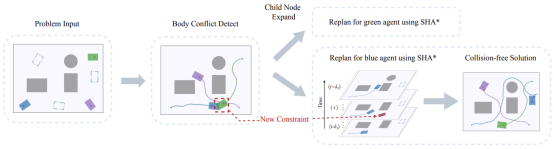

Figure 1: Pipeline of CL-CBS for multiple agents with their start and goal states represented as solid colored and dotted outline rectangles, respectively. [17]

1.2 CL-CBS: Spatiotemporal HA*

Hybrid A*, or HA^*, does not consider spatiotemporal constraints when applied to a continuous 3D state space for car like agents. Licheng et al in [17] introduce spatiotemporal $HA*$ that can plan paths that satisfy the kinematic constraints and spatiotemporal inter-agent constraints. Spatiotemporal HA^* is able to do so as it uses a 4D search space (t, x, y, θ) where $x, y, \theta \in R$ and t is discrete. The authors used seven different steering actions to expand each node using $HA*$. The heuristic also considered turning actions, driving backward and switching the moving direction.

1.3 Sequential CL-CBS

As more nodes are expanded in high level search when there are multiple agents in an area during a time step, sequential CL-CBS can be used to reduce the high level search time. In sequential CL-CBS, the agents are divided into batches, and each batch is sequentially

solved as a sub-CL-MAPF. The results are later combined for the paths calculated. The paths planned out in earlier batches act as dynamic obstacles as they are added to the constraint in the root node of BCT. This method however, comprises the completeness of CL-CBS as discussed in Section 8.

2 CHALLENGES OF CL-MAPF

In addition to computing cost-effective paths for individual agents within a given space, an algorithm that simultaneously plans for multiple agents will need to account for not only static obstacles but also dynamic obstacles imposed by the other known agents in the workspace. This entails finding intelligent ways to detect and avoid future collisions between one or more agents, which is both an interesting and challenging problem. Accounting for agents' kinematic constraints in planning adds additional complexity and improves the algorithm's applicability to real-world multi-agent path finding scenarios. Such scenarios may include interaction of multiple autonomous cars and navigating multiple agents through crowds.

3 RELATED WORK AND LIMITATIONS OF PREVIOUS APPROACHES

3.1 Categories of MAPF Solutions

As described in [15], reduction-based MAPF methods [3, 14, 19] aim to reduce MAPF to another problem. Reduction-based methods are effective for small and dense graphs are not as effective as the graph size grows. Švancara et al. [15] attempt to address this using graph pruning. Though utilizing graph pruning reduces computational time, completeness and/or optimality is sacrificed. Other reduction-based methods, including [3, 14], utilize heuristics/constraint relaxation to improve solution efficiency/quality for larger problems, but this can sacrifice optimality of solutions. Further, as variables such as the number of agents or map size grow larger, the number of parameters needed to accurately formulate and constrain the reduced problem can increase greatly.

A*-based methods [5, 10, 13, 16] utilize variants of A*. When paired with an admissible heuristic, A* search will find an optimal solution. However, the branching factor of A* search can grow rapidly, especially for large graphs. Wagner and Choset [16] introduces *subdimensional expansion*, which can be used to reduce search complexity and search space size of a multi-robot path planning (MPP) problem. [13] uses *operator decomposition* (OD) to reduce the branching factor of A* when used for solving the MAPF problem. However, a tradeoff of this is that the depth of the solution will increase. The most recent algorithm within this category cited by [17], ODrM* [5], utilizes a combination of OD, a recursive version of M* (rM*) and conflict-based search (CBS) to reduce the size of the search space as much as possible, only increasing the dimensionality of the search space when needed for collision resolution. Finally, Phillips and Likhachev [10] introduce Safe-Interval Path Planning (SIPP), which reduces the state space by introducing *safe intervals*, an interval of time for which a configuration is not involved in a collision. By using safe intervals, the state space is reduced to a few states per configuration (one for each of its safe intervals) instead of several states per configuration (one for each timestep in which a configuration is collision-free).

As explained in [2], prioritized MAPF methods assign a unique priority to each agent, and planning is done sequentially from highest to lowest priority agent such that lower-priority agents must avoid higher-priority agents. This algorithm is incomplete, as it is possible that valid solutions can be invalidated by ignorance of tasks of lower priority robots (that should occur before tasks of higher priority robots in order to ensure a valid solution) when planning for higher priority robots. The incompleteness of classical prioritized planning is further discussed in [2]. A revised prioritized planning algorithm introduced in [2]. However, a notable limitation of this approach is that a few assumptions about the agent's movements that are necessary for the algorithm to succeed in finding a solution are made. Further, a path satisfying these assumptions may not exist and this algorithm is still incomplete.

Dedicated search-based MAPF methods utilize search-based planning methods (that are not necessarily based on A*) to solve the MAPF problem. One such method, ICBS [1], improves on conflict-based search (CBS). CBS itself is a popular algorithm used in MAPF, implying its effectiveness in solving this problem.

3.2 Conflict-Based Search

The high-level planner utilized in CL-CBS is based on the high-level planner presented in conflict-based search. As such, an explanation of conflict-based search is presented here.

As described in Section 3.1, CBS is search-based planner. Many MAPF algorithms are based on A*. As mentioned in Section 3.1, the branching factor of A* is quite large. Additionally, as mentioned in the CBS paper, the size of the state space is exponential in the number of agents. Thus, an A*-based solution to the MAPF problem may be too time or space inefficient. To address these inefficiencies, CBS divides MAPF into single-agent path finding problems, where the solution to each is constrained based on the paths of other agents.

CBS consists of a high and low-level planner. The high-level planner utilizes a *conflict tree* (CT). As described in [12], each node in the conflict tree consists of:

- A set of constraints, where a single constraint is defined as a tuple (a_i, v, t) implying that agent a_i cannot be at vertex v at time t
- A solution, a set of paths consisting of one path per agent, which satisfies the node's constraints
- The total cost of the node's solution (the sum of all single-agent path costs)

The high-level planner performs a best-first search on the conflict tree based on node costs. Each node found by this search is passed to the low-level planner, which finds a set of shortest paths for all agents that satisfy the constraints of this node. This solution is then validated. If validation succeeds, the search stops and the current node is declared to be the goal (conflict-free) node. Otherwise, the node is a non-goal node and conflicts must be resolved.

Formally, a conflict is a tuple (a_i, a_j, v, t), which implies that both agents a_i and a_j occupy vertex v at time t. To resolve this conflict, the current node is split into two children. To account for all possible solutions, one child node will contain constraint (a_i, v, t) and the other will contain constraint (a_j, v, t). In the case that a conflict is between more than two agents, the conflict between two

agents is resolved at the current level and the conflict between the rest of the agents is delayed to later levels.

3.3 MAPF Solutions Accounting for Kinematic Constraints

In the context of the CL-MAPF paper, an important limitation of the planning methods presented in Sections 3.1 and 3.2 is that they ignore robots' kinematic constraints. MAPF methods cited by the CL-MAPF paper that consider kinematic constraints [6, 8, 9, 18] do not address them to the extent that [17] does. However, the ideas presented in these works could possibly be improved on using ideas from CL-MAPF.

3.4 Limitations of Previous Solutions

In addition to ignoring kinematic constraints when planning, [17] emphasizes that the main limitations of many previous MAPF methods are that they assume that agents represent disks that can rotate in place and/or only consider discrete 4-connected grids as their state space. Classic MAPF solvers use vertex conflicts and edge conflicts which lead to coarser solutions that do not account for all cases of agents colliding, limiting the real-world applicability of these plans. Further, several of the real-world scenarios that MAPF is applicable to utilize nonholonomic, car-like robots. The objective of the CL-MAPF paper is to address these limitations and define the MAPF problem for car-like robots.

4 REPLICATION STUDY

Given that the authors' solver outperformed previous solvers for CL-MAPF and the vast literature and research that is ongoing and has been done on MAPF, we seek to pursue a replication study for the paper [17].

Our objective for this replication study is two-fold. Firstly, we verify the results presented in the paper by running the experiments presented in the paper ourselves. Further, we intend to compare one of the baselines used, HA*, to CL-CBS in a similar manner to the comparisons done in [17]. We implemented HA* for this purpose. Secondly, we expand on the original paper's findings by designing and running new experiments for CL-CBS. The new experiments are designed to discover limitations of CL-CBS. We will vary parameters including map size, number of agents, and obstacle radii. The full set of experiments will be described in Section 6.

5 REPLICATED RESULTS

The authors of [17] included a benchmark dataset along with their code implementing the CL-CBS algorithm. The benchmark contains examples consisting of various size workspaces (50x50, 100x100, and 300x300) with varying number of agents and (static) obstacles.

Table 1 shows the replicated results of Table 1 from the original paper [17]. The authors did not specify the exact data/examples from the benchmark they used when gathering the results displayed in the paper. Thus, averaged metrics of three examples from the benchmark per combination of map size/number of agents/number of obstacles are shown. These makespans are within 20 m of the values reported in [17].

Figures 2 through 5 show multi-agent plans for example scenarios with a variety of workspace sizes, number of agents, and number of obstacles. These figures show the state of the workspace at the end of the simulation (i.e., when all agents have reached their goal positions). The lines trailing the agents show their paths. These figures will serve as baselines for the results in Section 7.

Table 1: Replicated Results

Map Size/Number of Agents/Number of Obstacles	Average Makespan Without Obstacles (m)	Average Makespan With Obstacles (m)
50x50/20/25	37.1906	42.7193
100x100/30/50	63.1697	59.8835
300x300/50/100	161.7723	162.6503

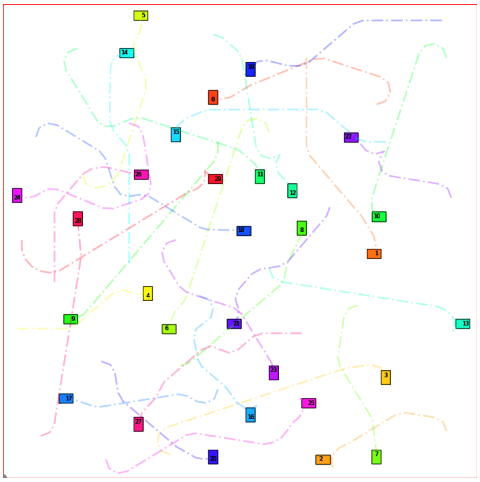

Figure 2: 100x100 map, 30 agents, no obstacles example plan

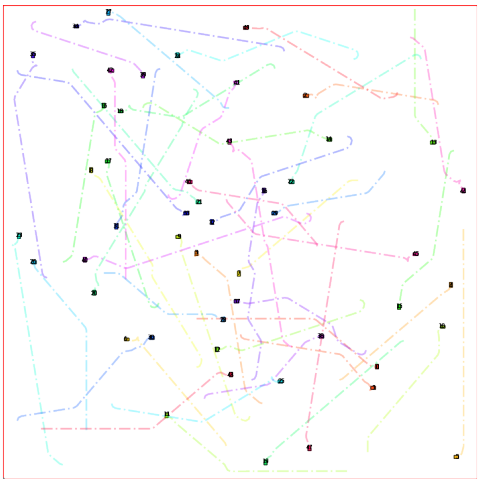

Figure 3: 300x300 map, 50 agents, no obstacles example plan

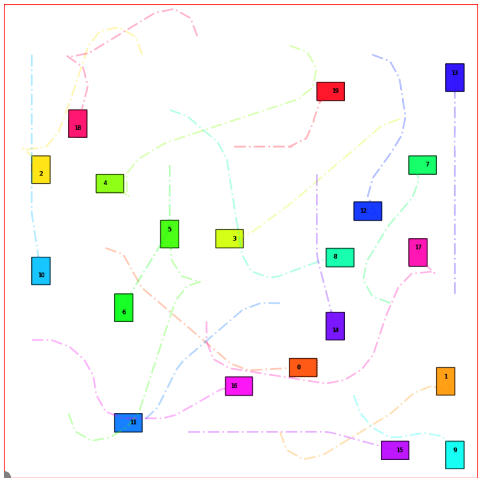

Figure 4: 50x50 map, 20 agents, no obstacles example plan

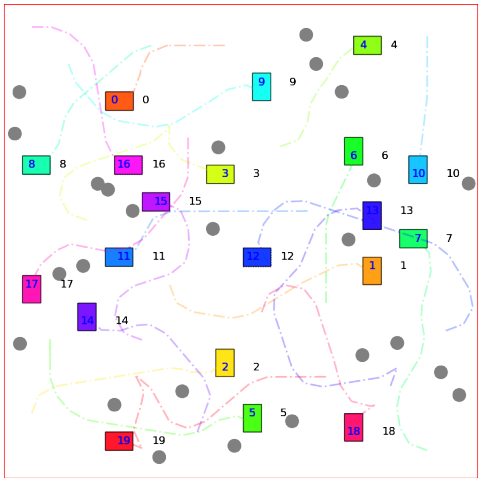

Figure 5: 50x50 map, 20 agents, 25 obstacles example plan

5.1 HA*

Licheng et al. in [17] do not discuss the heuristic implementation of $HA*$ or CBS-MPC against which they tested CL-CBS. However, the authors provided the implementation of spatiotemporal HA^*. By ignoring the temporal constraints and only considering spatial constraints, and using eucledian distance between current state and goal state, we were able to implement HA^* and run in scenarios without obstacles. The search algorithm used the heuritis estimate to expand the states. The state with lowest f-score is expanded first, where , the f-score is an estimate of the total cost of the path through the current state to the goal state. For more than 30 agents, HA^* was not able to solve more than 10 percent of the instances in 100x100 mapset.

We evaluated the runtime for agents between 5 to 30 for 100x100 mapset without obstacles. $HA*$ did not run on more than 90 percent of the instances for 100x100 mapset.

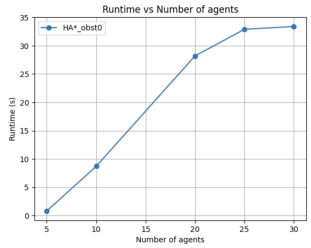

Figure 6: Runtime for scenarios with varying numbers of agents in a 100x100 workspace without obstacles

6 NEW EXPERIMENTS

In order to gain a deeper understanding of the limitations of the CL-CBS planner, we have designed a series of experiments to test its performance in various challenging scenarios. These scenarios include varying map sizes, numbers of agents, Euclidean distances between start and goal states, and obstacle configurations. By subjecting the planner to these diverse scenarios, we aim to identify the strengths and weaknesses of the planner. Our experimentation aims to provide valuable insights into the performance of the CL-CBS planner specifically for non-holonomic agents with Ackerman steering geometry as the kinetic model in scenarios that may or may not have obstacles.

6.1 Data Generation

The input and output to the CL-CBS algorithm are stored in YAML files with a well-defined format. This greatly facilitated the process of running additional experiments.

We generated the data for the new experiments by treating the workspace as a grid, with each cell, a square, holding an agent (or obstacle in some cases). We calculated the number of rows and columns in this grid is calculated as follows:

(1) $map\_area = map\_width^2$

(2) $cell\_width = \lfloor \sqrt{\frac{map\_area}{num\_agents}} \rfloor$

(3) $num\_rows = num\_cols = \lfloor \frac{map\_width}{cell\_width} \rfloor$

The number of rows and columns are sufficient to fit one agent per cell for all experiments. For the experiment placing two large obstacles in the middle of the map, the obstacles are treated as (static) agents and thus occupy a cell of their own throughout the entirety of the experiment.

6.2 New Experiment Details

In the CL-MAPF paper [17], the authors discuss a set of constraints for which each example in their benchmark dataset follow. Of the constraints discussed, the following were most intriguing:

- the Euclidean distance between start and goal state of an agent is greater than 1/4 of the map width
- examples with obstacles contain circle obstacles with a 0.8 m radius and the entire obstacle region occupies 1% of the map area

In addition to creating experiments that varied map size and agent numbers, we created new experiments that varied these constraints to gain insight about the importance of these constraints

Table 2: New Experiments

Purpose	Details
Vary map size	Examples with a 200x200 map with and without obstacles containing 20, 25, 30, 40 and 50 agents
Vary number of agents within 50x50 map	Examples without obstacles containing 30, 40, and 50 agents
Vary number of agents within 100x100 map	Examples without obstacles containing 60, 70, and 80 agents
Vary Euclidean distance from start to goal	1. 50x50 map, no obstacles, 20 agents, start-goal distance 1/2 of map width 2. 100x100 map, no obstacles, 30 agents, start-goal distance 1/10 of map width 3. 300x300 map, no obstacles, 50 agents, start-goal distance 1/8 of map width
Vary size and number of obstacles	100x100 map with 40 agents and 2 7m radius obstacles in middle of the map
Vary size of obstacle region	50x50 map containing 20 agents with 75 0.8 m radius obstacles, obstacle region occupies 6% of map

to the success/failure of a solution being found. Table 2 shows the additional experiments created. Their results will be discussed in the following section.

7 RESULTS

7.1 CL-CBS

For each figure referenced that contains numeric labels, the starting positions are labeled with the corresponding agent number within the solid rectangles, and the goal positions are labeled with the corresponding agent numbers to the right of the dashed bordered rectangles (if no dashed rectangle is immediately visible, this is because a different agent's starting position overlaps). In these scenarios, the agents start to goal paths form a "snake" like pattern from bottom left to top right.

For each scenario, a time limit of 2 minutes per iteration for the scenarios is set. One iteration comprises of searching through the body conflict tree for the next best node, then performing low-level searches for each agent to find its individually optimal path, and finally validating all of these paths among each other to determine whether or not any of the agents' paths conflict. This is a constraint set by the authors in [17].

If a figure visualizing the output of CL-CBS does not contain paths (i.e., trails behind agents), it implies that a plan was not found for the given scenario under the time constraint.

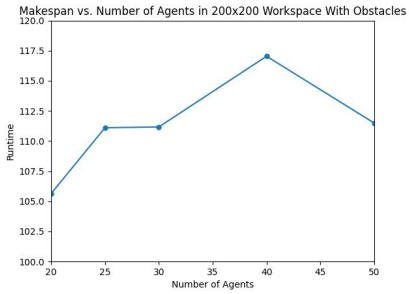

Figure 7: Makespan for scenarios with varying numbers of agents in a 200x200 workspace with obstacles

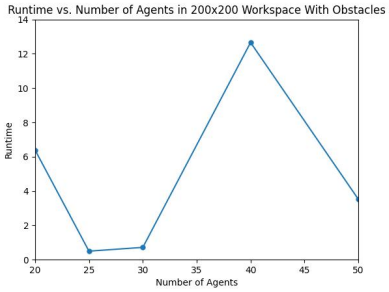

Figure 8: Runtime for scenarios with varying numbers of agents in a 200x200 workspace with obstacles

7.1.1 ***Varying Workspace Size.*** Figures 7 through 10 show the makespan and runtimes (defined in [17]) for planning scenarios with varying numbers of agents in a 200x200 workspaces without obstacles. CL-CBS is able to find a solution for all the scenarios tested. As expected, the makespan generally increases as the number of agents in the workspace increases, as an increased number of agents will require more frequent collision avoidance, increasing the makespan. There does not seem to be a discernible trend in the runtimes of these scenarios.

7.1.2 ***Varying Number of Agents.*** Figures 11 through 13 show visualizations of the paths and/or start and goal positions for each agent in a 100x100 workspace. CL-CBS successfully finds a plan for the scenario in Figure 11 but fails to find a solution within the time constraints for the scenarios in Figures 12 and 13. As the number

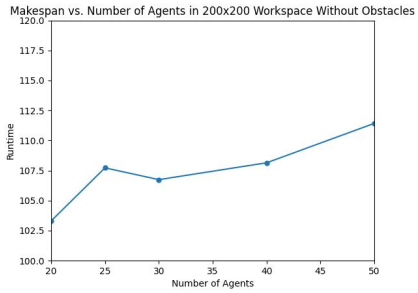

Figure 9: Makespan for scenarios with varying numbers of agents in a 200x200 workspace without obstacles

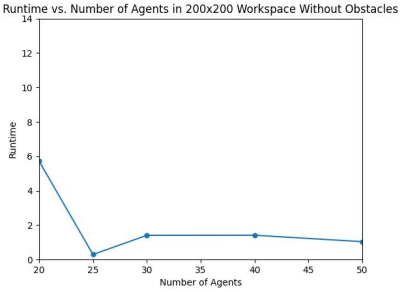

Figure 10: Runtime for scenarios with varying numbers of agents in a 200x200 workspace without obstacles

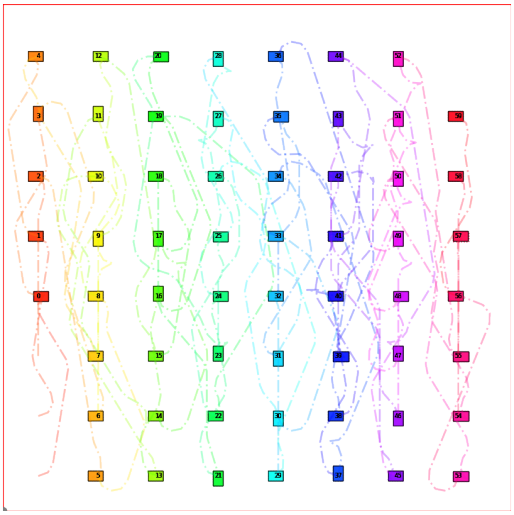

Figure 11: Visualization of paths taken for each of 60 agents in an obstacle-free 100x100 workspace

of agents in the workspace increases, the amount of empty space decreases. This implies that the agents have less room to navigate around other agents, increasing the number of conflicts. CL-CBS resolves conflicts by adding more nodes to the body conflict tree, which leads to further low-level searches to find single-agent paths

Figure 12: Visualization of start and goal positions for each of 70 agents in an obstacle-free 100x100 workspace

Figure 13: Visualization of start and goal positions for each of 80 agents in an obstacle-free 100x100 workspace

for each agent which must then be validated together to check for further conflicts. Thus, an increase in conflicts will significantly increase the runtime of the algorithm.

Figures 14 through 16 show visualizations of the paths and/or start and goal positions for each agent in a 50x50 workspace. CL-CBS is unable to find solutions for each of these scenarios within the time constraints. The reason for this is similar to the reasoning made for the scenarios in Figures 11 through 13. For a 50x50 workspace, the amount of empty space is even more constrained. Additionally, each agent takes up a larger proportion of the workspace's area compared to that in a 100x100 workspace.

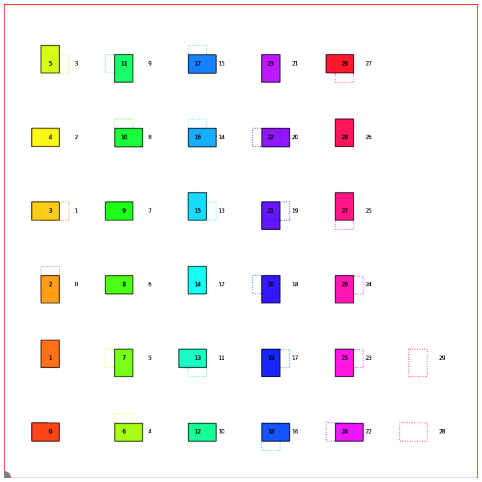

Figure 14: Visualization of start and goal positions for each of 30 agents in an obstacle-free 50x50 workspace

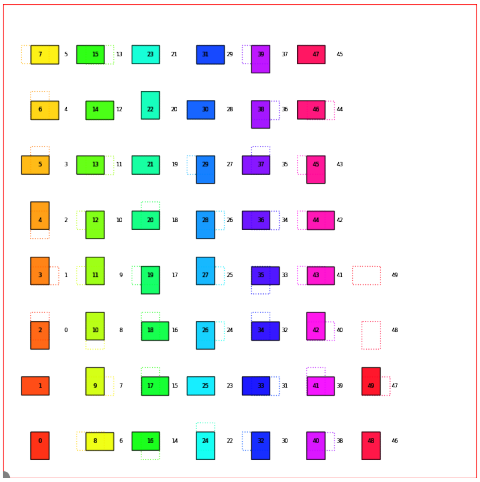

Figure 16: Visualization of start and goal positions for each of 50 agents in an obstacle-free 50x50 workspace

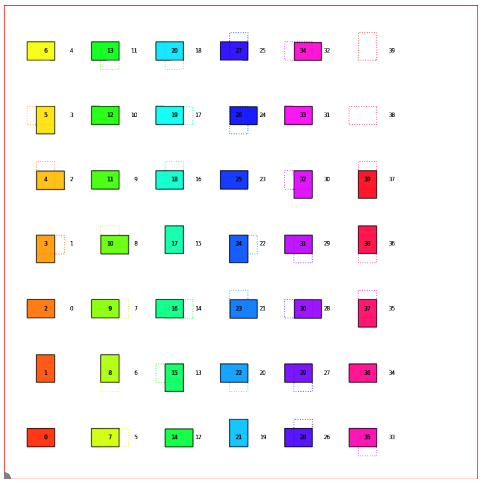

Figure 15: Visualization of start and goal positions for each of 40 agents in an obstacle-free 50x50 workspace

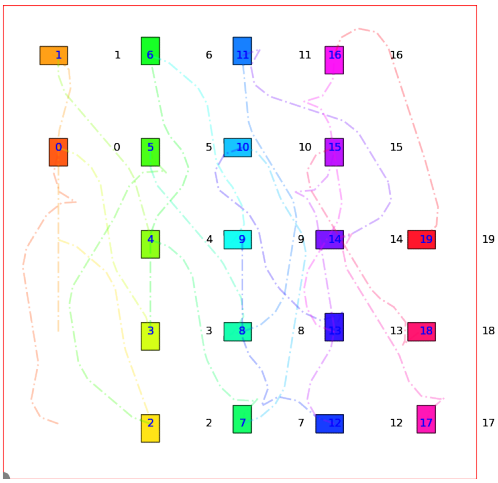

Figure 17: Visualization of paths taken for each of 20 agents in an obstacle-free 50x50 workspace, where the Euclidean distance from the start to the goal for each agent is larger than 1/2 map width

7.1.3 ***Varying Distance From Start To Goal.*** The following scenarios investigate the effect of varying the Euclidean distance from agent start to goal condition specified in [17]. The agent's goal positions are determined by determining the number of "cell widths", using the definition of cell from Section 6.1, that must separate the start and goal positions of a given agent. For example, if the separation between the start and goal positions of an agent must be 1/2 of the workspace width, then the goal position is computed by advancing the agent's position by at least the number of cells corresponding to the distance that comprises 1/2 of the workspace width. As the agent's start positions are placed in a snake-like pattern (from least to greatest agent number), the goal positions are placed in this pattern as well.

Figures 17 and 18 visualize the states of a 50x50 workspace with 20 agents. The Euclidean distance between the start and goal in Figure 17 is 1/2 of the workspace width (i.e., 25 m for a 50x50 workspace), and 1/4 of the workspace width (the baseline Euclidean distance used in the benchmark in [17]).

Visually, the difference in start to goal separation is shown by the difference number of "cells" that each agent travels between. In Figure 17, each agent's path spans three cells. For example, agent 4 travels from the top cell in the first column from the left to the third cell from the bottom in the second column from the left (this is shown by the trail behind agent 4). On the other hand, in Figure 18, each agent's path spans two cells. For example, agent 4 starts at the same position as in the scenario in Figure 17, but only travels to the second cell from the bottom in the second column from the left.

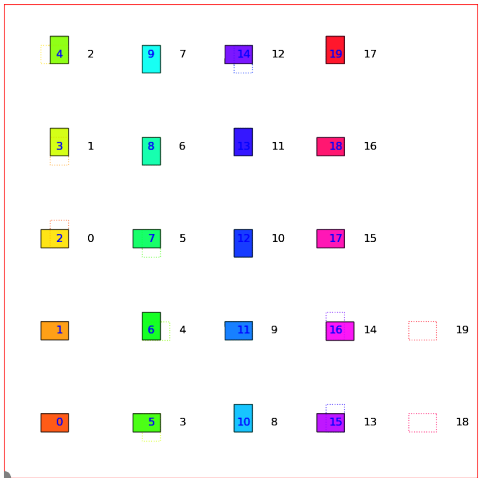

Figure 18: Visualization of start and goal positions for each of 50 agents in an obstacle-free 50x50 workspace, where the Euclidean distance from the start to the goal for each agent is larger than 1/4 map width

In this scenario, the difference in separation makes a difference in whether or not a plan can be found for the given input set of agent start and goal positions, as a plan can be found for the scenario in Figure 17 but not in the given time constraints for the scenario in Figure 18. A larger Euclidean distance from the start to the goal implies that the agents have more freedom in choosing their path, and can thus more easily avoid collisions with other agents.

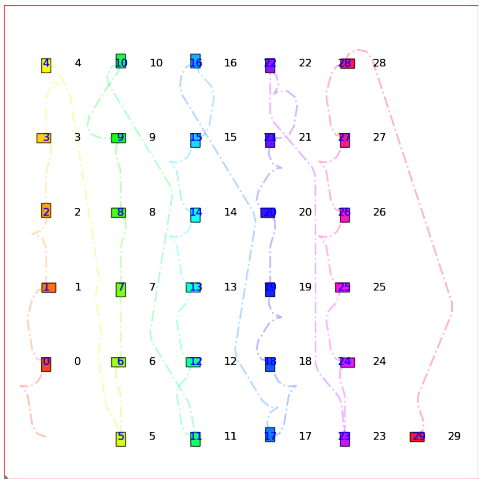

Figure 19: Visualization of paths taken for each of 30 agents in an obstacle-free 100x100 workspace, where the Euclidean distance from the start to the goal for each agent is larger than 1/10 map width

Figures 19 and 20 visualize the states scenarios in a 100x100 workspace with 30 agents.

The Euclidean distance between the start and goal in Figure 19 is 1/10 of the workspace width (i.e., 10 m for a 100x100 workspace),

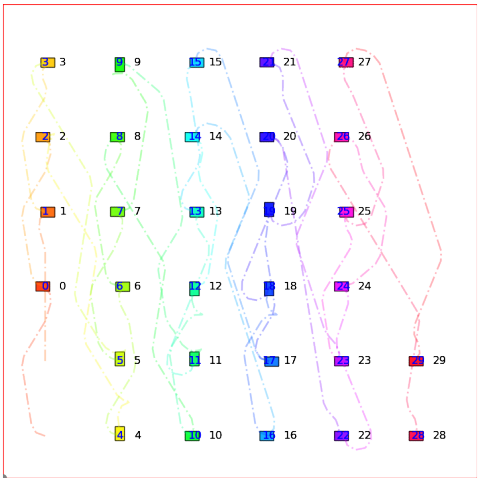

Figure 20: Visualization of paths taken for each of 30 agents in an obstacle-free 100x100 workspace, where the Euclidean distance from the start to the goal for each agent is larger than 1/4 map width

and 1/4 of the workspace width in Figure 20 (the baseline Euclidean distance used in the benchmark in [17]).

As discussed in the similar previous set of scenarios with the 50x50 workspace, the difference in start to goal separation is shown by the difference number of "cells" that each agent travels between. In Figure 19, each agent's path spans one cell. On the other hand, in Figure 20, as in the scenario with the same distance separation between start and goal positions with the 50x50 workspace, each agent's path spans two cells.

CL-CBS is able to find a plan for both of these scenarios. The spacing between agents in the 100x100 workspace, and the distance between the start and goal for all agents seems to be sufficient compared to the 50x50 scenarios.

Figures 21 and 22 visualize the states scenarios in a 300x300 workspace with 50 agents.

The Euclidean distance between the start and goal in Figure 21 is 1/8 of the workspace width (i.e., 37.5 m for a 300x300 workspace), and 1/4 of the workspace width in Figure 22 (the baseline Euclidean distance used in the benchmark in [17]).

As discussed in the similar previous two sets of scenarios with the 50x50 and 100x100 workspaces, the difference in start to goal separation is shown by the difference number of "cells" that each agent travels between. In Figure 21, each agent's path spans one cell. On the other hand, in Figure 22, as in the scenario with the same distance separation between start and goal positions with the 50x50 workspace, each agent's path spans two cells.

In this scenario, as with the 50x50 workspace scenarios, difference in separation makes a difference in whether or not a plan can be found for the given input set of agent start and goal positions, as a plan can be found for the scenario in Figure 22 but not in the given time constraints for the scenario in Figure 21. Similar reasoning to previous scenarios in this section can be applied.

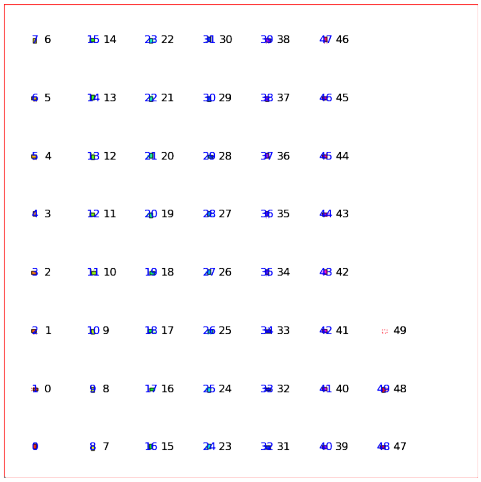

Figure 21: Visualization of start and goal positions for each of 50 agents in an obstacle-free 300x300 workspace, where the Euclidean distance from the start to the goal for each agent is larger than 1/8 map width

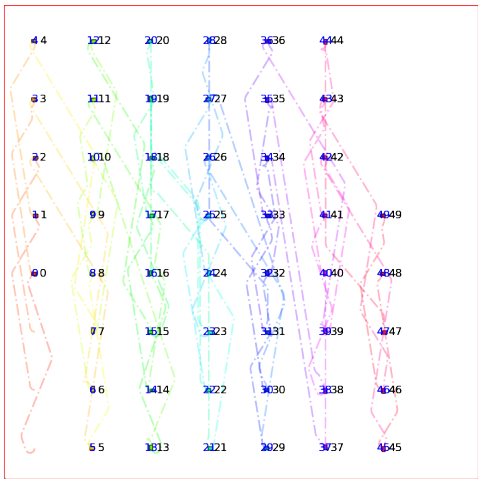

Figure 22: Visualization of paths taken for each of 50 agents in an obstacle-free 300x300 workspace, where the Euclidean distance from the start to the goal for each agent is larger than 1/4 map width

7.1.4 ***Varying Obstacle Size and Number.*** The following scenarios investigate the effect of varying (static) obstacle size and number. As mentioned in [17], each obstacle in the benchmark dataset has a default radius of 0.8 m and the obstacle area comprises 1% of the total workspace area.

Figures 23 through 25 visualize the results of applying CL-CBS to scenarios in a 100x100 workspace with 40 agents and varying numbers and size of the obstacle. In Figure 23, two large (7 m radius) obstacles are placed in the middle of the workspace. A plan is unable to be found within the given time constraints. In Figure 24, 36 obstacles are placed evenly throughout the workspace. A plan is found for this scenario. Finally, as a baseline comparison, 50 obstacles are placed in a seemingly random manner in Figure 25. A plan is unable to be found in the given time constraints for this scenario as well.

These results indicate that the size and location of obstacles affects the planning process. A dense obstacle region with two large obstacles (radius 7 m) in the middle of the workspace hinders planning to a greater extent than a workspace with fifty of smaller obstacles (radius 0.8 m) scattered around. The former scenario times out on the second iteration, whereas the latter scenario times out on the fourth. The obstacle area in the former scenario is 98π and the obstacle area in the latter scenario is 32π. Even though the obstacle region in the former scenario is larger compared to the obstacle region in the latter scenario, the results seem surprising. A densely packed region of obstacles in the middle provides greater free space towards the edges of the workspace compared to a scattered region. Further, the agents may have been able to maneuver in a circular pattern around the center obstacles (much like a traffic circle), but cannot necessarily do so with the scenario with scattered obstacles.

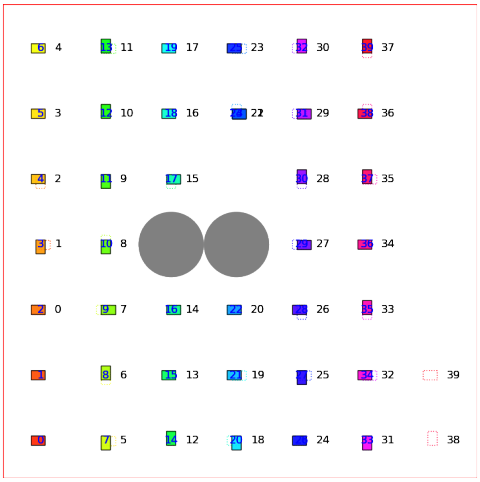

Figure 23: Visualization of start and goal positions for each of 40 agents in a 100x100 workspace with two large (7 m radius) obstacles in center of workspace

Figures 26 through 28 visualize the results of applying CL-CBS to scenarios in a 50x50 workspace with 20 agents and varying numbers of obstacles. In Figure 27, obstacles take up 6% of the workspace. A plan is able to be found for this scenario. In Figure 28, 16 obstacles are placed evenly throughout the 50x50 workspace. A plan is unable to be found for this scenario within the time constraints. Finally, Figure 5 is used as a baseline comparison, which visualizes a 50x50 workspace with 25 obstacles placed in a random pattern. A plan is able to be found for this scenario.

Interestingly, a solution is unable to be found for the scenario with the smallest number of obstacles among the three recently discussed. The structure of the obstacle region may explain this behavior. In Figure 5, there is no noticeable structure to the obstacles and thus the agents are not restrained in a particular way. On the other hand, the obstacles form a lattice-like pattern in Figures 26

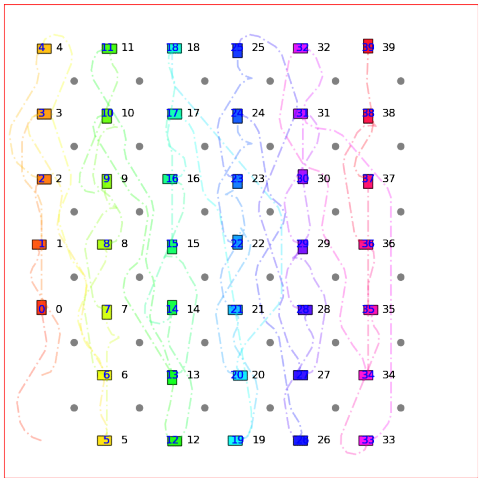

Figure 24: Visualization of paths taken for each of 40 agents in a 100x100 workspace with 36 obstacles of radius 0.8 m

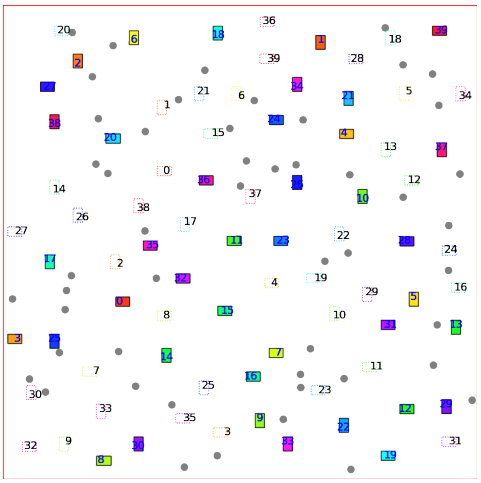

Figure 25: Visualization of start and goal positions for each of 40 agents in a 100x100 workspace with 50 obstacles of radius 0.8 m (baseline from original benchmark set created by Wen et al. [17]

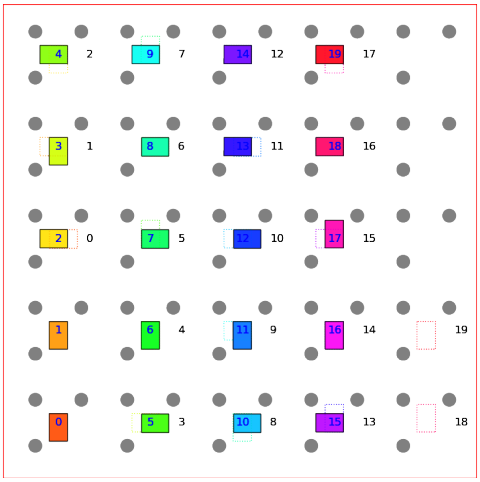

Figure 26: Visualization of start and goal positions for each of 20 agents in a 50x50 workspace with 75 obstacles of radius 0.8 m (6% of workspace area)

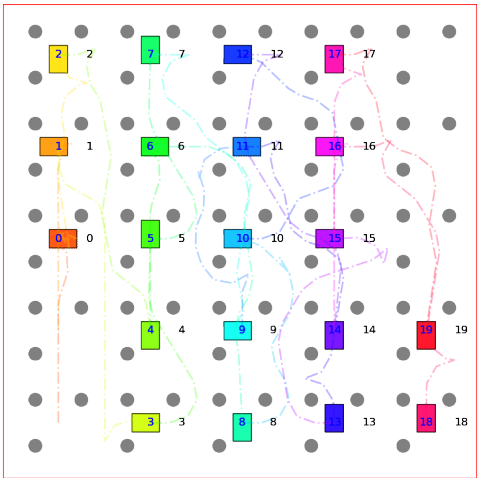

Figure 27: Visualization of paths taken for each of 40 agents in a 100x100 workspace with 75 obstacles of radius 0.8 m (6% of workspace area)

and 28. Further, the agents in the latter scenario are more "boxed in" compared to the agents in the former scenario.

8 LIMITATIONS OF CL-CBS

(1) Sequential CL-CBS compromises the completeness of the algorithm. As shown in Figure 29. The blue agent cannot reach its goal because the gray agent that was planned previously is parked between the obstacles.
(2) As authors expanded from discrete to a continuous workspace Spatiotemporal Hybrid A^* has scalability issues for increase in obstacles or increase in workspace area.
(3) Licheng et al. in [17] do not discuss the heuristic or implementation of $HA*$ or CBS-MPC against which they tested CL-CBS.
(4) Through our experimentation, we found that the success ratio of the CL-CBS planner decreases for map sizes of 100x100 with more than 70 agents. This indicates a limitation of the planner for larger multi-agent scenarios and highlights the need for further improvements to enhance its scalability.
(5) During our experimentation, we observed that when the size of the obstacles was increased in a 100x100 map with 40 agents, as shown in Figure 23, the CL-CBS planner was unable to generate collision-free paths for the agents. This

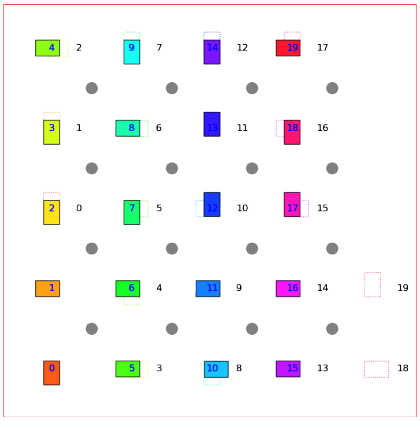

Figure 28: Visualization of start and goal positions for each of 40 agents in a 50x50 workspace with 16 obstacles of radius 0.8 m

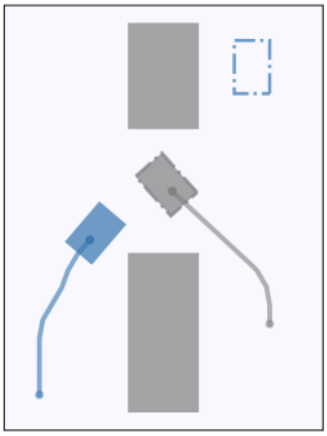

Figure 29: Sequential CL-CBS: A fail case [17]

highlights the limitations of the planner in dealing with complex or big obstacle configurations and emphasizes the need for developing new techniques to overcome these challenges.

Another limitation of CL-CBS is its behavior when planning for agents moving in a non-random pattern, as explored in Section 7. Though CL-CBS is able to plan for some such scenarios, the algorithm fails in others given the time constraints due to a seeming lack of maneuverability. Based on the example scenarios we tested from the benchmark, the agents seem to be placed in a random manner.

Additionally, the size, placement, and density of obstacles seems to noticeably impact planning, which is expected. The exact relationship between these factors and the ability for a solution to be found is not clear.

9 FUTURE WORK AND CONCLUSION

(1) Test CL-CBS with scenarios designed to follow real-world traffic patterns, such as automobile traffic, traffic in warehouses, or traffic in offices (which likely will involve interaction between robots and humans)
(2) Better understand the impact of the size, placement, and density of obstacles on planning.
(3) Vary time constraints placed on the runtime of the algorithm on scenarios based on perceived planning difficulty
(4) As we can see from sequential CL-CBS fail case, one direction of future work can be allowing agents to move after they reach the goal to allow other agents to reach their goal
(5) Extend CL-CBS to non-holonomic and holonomic agents.
(6) Evaluate CL-CBS using makespan and sum of costs under the scenario where each agent must maintain a speed.
(7) Try to extend CL-CBS to Target Assignment and Path Finding, wherein we must have optimal assignment of multiple agents to a set of targets, and the agents should follow the best path to reach their respective target.

REFERENCES

[1] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, Oded Betzalel, David Tolpin, and Eyal Shimony. Icbs: The improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 6, pages 223–225, 2015.

[2] Michal Čáp, Peter Novák, Alexander Kleiner, and Martin Selecký. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE transactions on automation science and engineering*, 12(3):835–849, 2015.

[3] Esra Erdem, Doga G. Kisa, Umut Oztok, and Peter Schüller. A general formal framework for pathfinding problems with multiple agents. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, page 290–296. AAAI Press, 2013.

[4] Ariel Felner, Roni Stern, Solomon Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Proceedings of the International Symposium on Combinatorial Search*, volume 8, pages 29–37, 2017.

[5] Cornelia Ferner, Glenn Wagner, and Howie Choset. Odrm* optimal multirobot path planning in low dimensional search spaces. In *2013 IEEE International Conference on Robotics and Automation*, pages 3854–3859. IEEE, 2013.

[6] Wolfgang Hönig, TK Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 26, pages 477–485, 2016.

[7] Sang-Hoon Ji, Jeong-Sik Choi, and Beom-Hee Lee. A computational interactive approach to multi-agent motion planning. *International Journal of Control, Automation, and Systems*, 5(3):295–306, 2007.

[8] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, TK Satish Kumar, and Sven Koenig. Multi-agent path finding for large agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7627–7634, 2019.

[9] Hang Ma, Wolfgang Hönig, TK Satish Kumar, Nora Ayanian, and Sven Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7651–7658, 2019.

[10] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE international conference on robotics and automation*, pages 5628–5635. IEEE, 2011.

[11] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.

[12] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[13] Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 173–178, 2010.

[14] Pavel Surynek. Uniagent: Reduced time-expansion graphs and goal decomposition in sub-optimal cooperative path finding. In *Proceedings of the International*

Symposium on Combinatorial Search, volume 6, pages 236–237, 2015.

[15] Jiří Švancara, Philipp Obermeier, Matej Husár, Roman Barták, and Torsten Schaub. Multi-agent pathfinding on large maps using graph pruning: This way or that way? In *Proceedings of the International Symposium on Combinatorial Search*, volume 15, pages 320–322, 2022.

[16] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ international conference on intelligent robots and systems*, pages 3260–3267. IEEE, 2011.

[17] Licheng Wen, Yong Liu, and Hongliang Li. Cl-mapf: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints. *Robotics and Autonomous Systems*, 150:103997, 2022.

[18] Konstantin Yakovlev, Anton Andreychuk, and Vitaly Vorobyev. Prioritized multi-agent path finding for differential drive robots. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2019.

[19] Jingjin Yu and Steven M LaValle. Planning optimal paths for multiple robots on graphs. In *2013 IEEE International Conference on Robotics and Automation*, pages 3612–3617. IEEE, 2013.