

recursive meta-agent cbs

Group 39 – Jayden Cole



April 25, 2021

CMPT 417

# Introduction

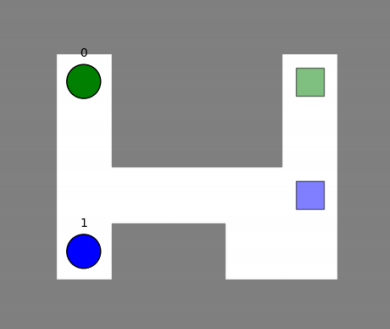
With the constant improvement and use of robotics in the modern world, the need for multi-agent pathfinding (MAPF) becomes ever more apparent. A MAPF problem consists of a map with obstacles and multiple agents, each with their own start and goal locations. The task of a MAPF solver is to move every agent through the map, to their goal state without colliding with other agents or any obstacles. Work done on MAPF solvers can try to find the minimum sum of cost for all the agents or the focus can be on finding a sufficient solution for all agents within some time bound. Below in Figure 1 is a simple example of a MAPF instance. The two agents (the green and blue circles) are in their start locations. The respectively coloured squares are the goal locations for the agents. The MAPF solver will attempt to find a collision free path for both agents along the “empty” white tiles from their start states to their goal states. These problems have been proven to be NP-Complete [1].

Figure 1: A simple example of a MAPF instance.

This project focuses on finding optimal solutions to MAPF instances while trying to minimize computation time. Particularly, it will try to improve the performance of conflict-based search (CBS) when there are agents whose paths collide many times, specifically the agents collide more than some predetermined parameter B. If there is a high rate of conflicts between agents, CBS will have to resolve many conflicts before finding a solution. A possible improvement is to calculate (separately) the solution for the highly dependant agents, then calculate the rest of the solution to the MAPF instance, treating the pre-calculated solution as one agent. This technique is known as meta-agent conflict-based search (MA-CBS).

Work has been done to try and find optimal values for the number of collisions between agents before merging them in MA-CBS. This has focused on the density and the topology of the MAPF instances while implementing a low-level solver of A\* or EPEA\*. However, Sharon et al. propose that MA-CBS may use itself (though slightly modified) as its own low-level solver. This is what this paper has accomplished. First, we will describe the implementation and changes required to make MA-CBS use itself as a low-level solver. Then, a methodology section will present what questions are asked of Recursive MA-CBS (RMA-CBS) and what instances will be used to attempt to solve them. Next, an experimental results section will be presented and finally a conclusion section will highlight the takeaways of the project and what future steps should be taken to further this research.

# Implementation

CBS and MA-CBS are both two-level algorithms using a top-level search and a low-level search. The top-level search is a constraint tree (CT) with nodes containing the time and location constraints for each agent. For every constraint tree node at the top level, a low-level search finds the optimal single agent paths based on the constraints in the top-level node. If collisions still exist after the low-level search, the high-level node is considered a non-goal node and the high-level search continues by adding more constraints to resolve the calculated conflicts.

However, the main difference between basic CBS and MA-CBS is the operation of combining individual agents into a group called a meta-agent. Only used in MA-CBS, a meta-agent is a group of one or more agents that have been merged into one agent. This happens when the number of collisions detected between two (meta-) agents is larger than or equal to some predetermined merge bound B. MA-CBS(B) will combine the agents as well as the constraints that apply to those (meta-) agents. Finally, MA-CBS(B) will call its low-level solver to find the optimal paths for all the agents within the meta-agent based on the constraints of the node.

This merge process is described in Figure 2 (next page), in lines 11-18 of the MA-CBS and CBS algorithm. For CBS, the *shouldmerge()* function always returns false and so no merging ever takes place. This is MA-CBS with a merge bound of infinity, and so CBS is really an extreme case of MA-CBS(B) with B=. MA-CBS will merge if the number of collisions between the two agents is greater than or equal to the merge bound B.

Delving into the merge process of MA-CBS, if a conflict occurs between (meta-) agents ai and aj, and the agents have B or more conflicts, the agents from (meta-) agents *a­I* and *aj* into one group of agents labeled *a­i,j*  (as shown on line 12). Then, the constraints applying to the two original (meta-) agents must be updated to apply to this new meta-agent. Constraints can be one of two types: internal constraints or external constraints. Internal constraints are cause by a conflict between agents in ai and aj and can be ignored upon merging ai and aj as the lower-level solver will find an optimal conflict free path for these agents from scratch. External constraints are constraints that apply to an agent in either ai or aj and another agent not in ai or aj. These constraints are updated to apply to the merged meta-agent and are passed to the lower-level solver.

There are many options for the low-level solvers of MA-CBS and CBS. But to keep MA-CBS’ optimality and completeness, the low-level solver must also be complete and optimal. Further, the lower-level solver must also be able to handle constraints. Examples of algorithms that fit these criteria are A\* + OD, EPEA\*, or M\*. Similarly, CBS and ICTS are algorithms there can be modified from their basic versions to be able to detect non solvable instances of MAPFs, and then they too can be used as a low-level solver. However, this paper presents MA-CBS as the low-level solver to itself resulting in a recursive structure of MA-CBS (RMA-CBS). When recursively calling itself, the merge bound for lower-level instances must be larger than the one above to avoid calling meta-agent CBS an infinite number of times. The way to increase this merge bound are varied, this paper tries a few basic techniques to see if there are improvements when solving MAPF instances and if some specific parameters of the MAPF instance have an influence on performance.

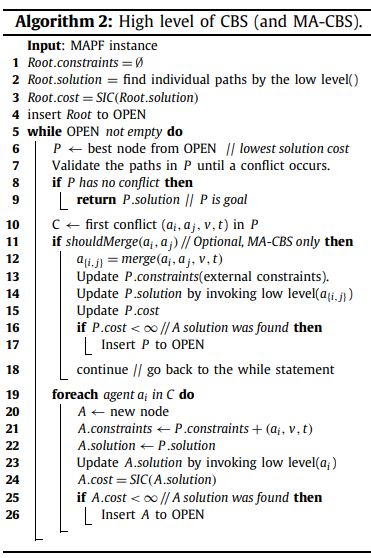
Using MA-CBS as its own low-level solver presents some overhead in generating the low-level solver instance. In this implementation, the lower-level solver did not know which specific agents from the upper levels it was solving. It always would solve a MAPF instance of N agents for agents numbered 0 to N-1. This change of values for agent numbers between the low level and higher-level MA-CBS instance means the constraints passed to the low level also had to be updated to properly apply to the new agent numbers in the low-level solver. Some additional time and memory were required to properly setup each instance of recursive MA-CBS.

Figure 2: Pseudocode for CBS and MA-CBS. Source: Shanon et al.

Otherwise, no special modifications were made to MA-CBS to optimize the algorithm to the MAPF instances tested. This was because there is no data on the performance of MA-CBS using itself as a lower-level solver and so a baseline was required before optimization can happen. However, a time limit per MAPF instance of 20s was applied to complete the full testing program in time to analyze results and make conclusions.

# Methodology

As there has been no baseline results for using MA-CBS as its own low-level solver, a general test across a spread of MAPF instances would most easily identify the strengths and weaknesses (if any) of the algorithm. By a general test, we mean a test on MAPF instances varying in size, agents, percentage of obstacles, and the merge bound increase technique. This would deliver answers to questions as to performance variations between different implementations for RMA-CBS across a variety of instances.

However, to ensure the optimality and completeness of the recursive RMA-CBS algorithm, a small test on the 50 MAPF instances from the individual project was conducted. Comparing the results to the least sum of costs file given with the project ensured the RMA-CBS was indeed complete and optimal.

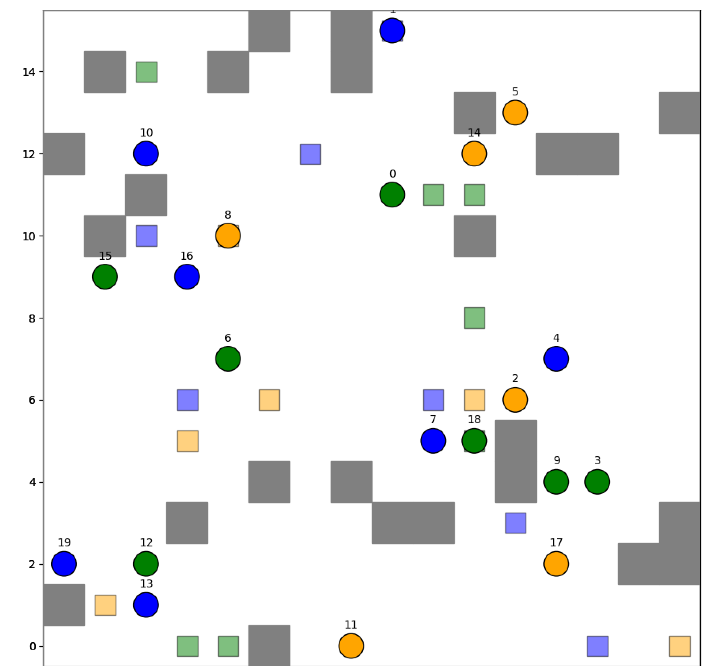
Next, a set of MAPF instances were randomly generated to test the algorithm. The number of agents per instance was either 5, 10, 15, or 20. The size of the instance ranged from 10x10 through 20x20, incrementing both the height and width by 2. The percentage of obstacles varied from 0% through 15%, in steps of 5%. Start and goal locations for agents were randomized as well as the placement of obstacles. Each combination of these parameters was then generated 20 times to create a total of 1920 unique MAPF test instances. An example of a generated instance is shown in Figure 2. This instance had 20 agents, a size of 16x16, and a percentage of obstacles of 10%.

Figure 3: A generated MAPF test instance with 20 agents, a size of 16x16, and 10% obstacle coverage.

The technique to increase the merge bound varied from incrementing the current merge bound by 1, 2, or 3 for each level down the RMA-CBS(B) chain. Finally, the starting merge bounds were chosen to reflect previous research by Sharon et al. Merge bounds of 1, 5, 10, 20, 100, and infinity were tested. Also, these would answer how CBS (as CBS is MA-CBS()) would perform on such instances and give a comparable result to the varying MA-CBS parameters.

# Experimental Setup

The experiment was run on Windows 10 using a four core i5-8250U CPU running at 1.4GHz. 8 GB of RAM was available to the program during runtime. The entire experiment was run using 32-bit Python version 3.8.1 including the test instances generation script as well as the data analysis.

# Results

Figure 4 depicts the percentage of solved instances for each RMA-CBS(B) variant with merge bound deepening value of 1, tested against the number of agents in the instance. As expected, all variants decrease in performance as the number of agents increase. The RMA-CBS algorithms that solved the most instances were those with the highest B values, suggesting that the starting merge bound parameter (B) has a heavy influence on the quantity of solves over time.

Interestingly, when increasing the merge bound deepening value, the results are slightly different as shown in Figure 5 below. RMA-CBS with initial merge bound 1 becomes the best algorithm when increasing its merge bound by 2 each iteration. Figure 6 then compares how the percentage of map that was obstacles affected the number of instances solved for each RMA-CBS(B).

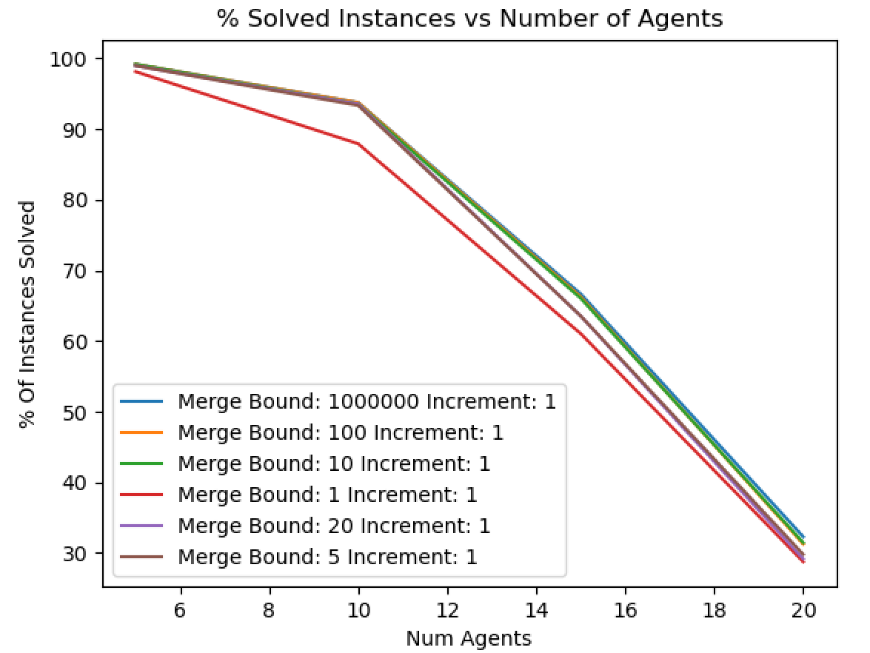
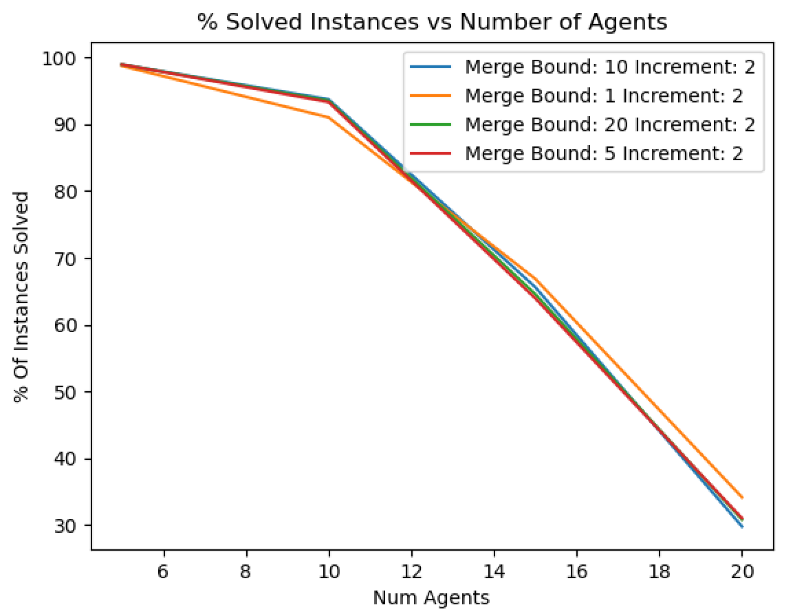


Figure 5: Percentage of solved instances vs number of agents for RMA-CBS variants with merge bound increasing 2 collisions per depth.

Figure 4: Percentage of solved instances compared to the number of agents for RMA-CBS with merge bound deepening at 1 per level.

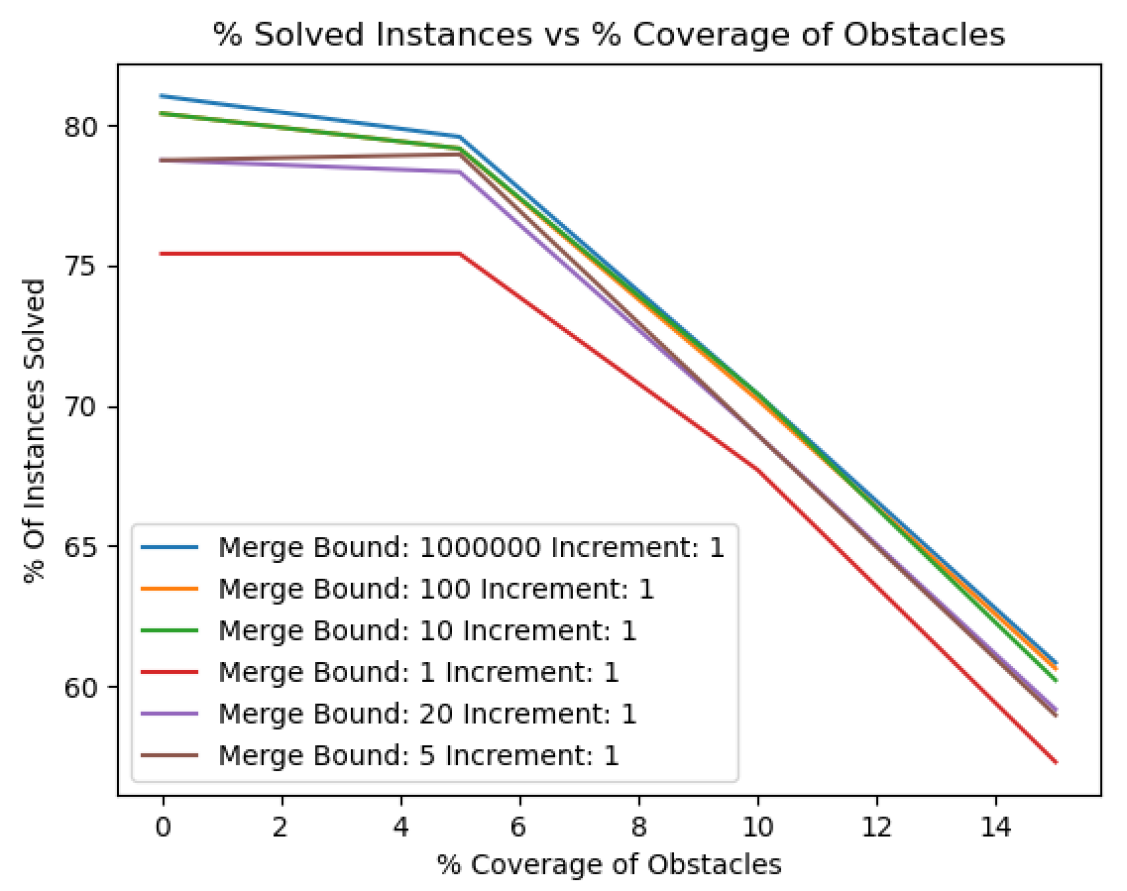
Again, Figure 6 shows the initial merge bound of the MA-CBS instance influenced the percentage of instances that were solved within the time bound. For example, RMA-CBS(100) solved over 5% more instances than RMA-CBS(1) when both had iterative deepening bounds of 1. This was consistent through obstacle percent coverage from 0-15%. When the iterative deepening bound was increased, RMA-CBS(100) was not affected, but the boost in solvability RMA-CBS(1) receives is quite large. This jump in performance is also felt by RMA-CBS(5) when its merge deepening value is increased, but for RMA-CBS(10) the solvability decreases with increased merge bound. Figures 7-10 depict the individual performance increase or decrease for each variant of RMA-CBS(B). The performance increase or decrease is consistent across all densities of obstacles.

Figure 6: Comparing the percent coverage of obstacles to the number of instances solved.

Finally, comparing the number of solved instances to the size of the instances generates results perhaps contrary to what would be thought. These results are shown in Figure 11 with a deepening value of 1, then again in Figure 12 with deepening value of 2.

Finally, comparing the number of solved instances to the size of the instances generates results perhaps contrary to what would be thought. These results are shown in Figure 11 with a deepening value of 1, then again in Figure 12 with deepening value of 2.

Finally, comparing the number of solved instances to the size of the instances generates results perhaps contrary to what would be thought. These results are shown in Figure 11 with a deepening value of 1, then again in Figure 12 with deepening value of 2.

Finally, comparing the number of solved instances to the size of the instances generates results perhaps contrary to what would be thought. These results are shown in Figure 11 with a deepening value of 1, then in Figure 12 the difference in increased performance between B=1 and B=10 when increasing the deepening merge value is shown. The % of instances solved increased as the size of the maps increased. This could be because agents were less likely to collide with the bigger map, and since the recursive structure of RMA-CBS seems to slow down performance when being called recursively, the less likely the agents collided would increase performance.

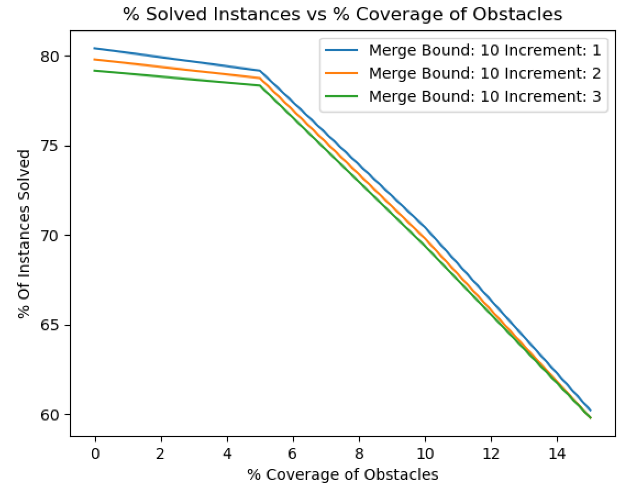
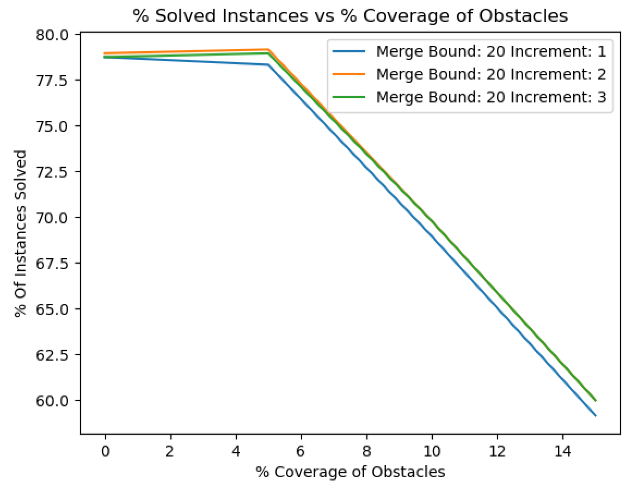
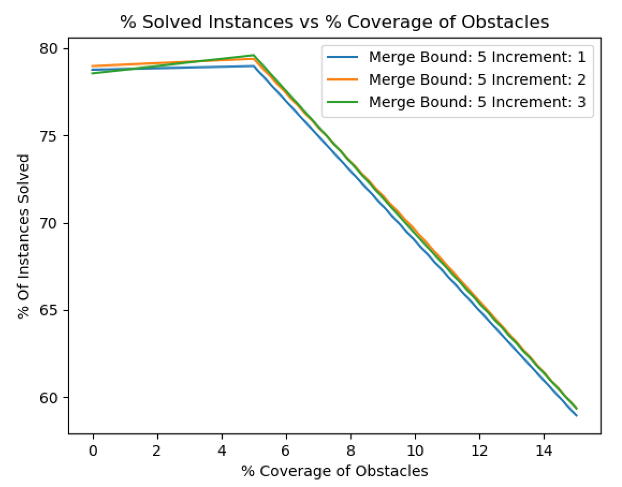
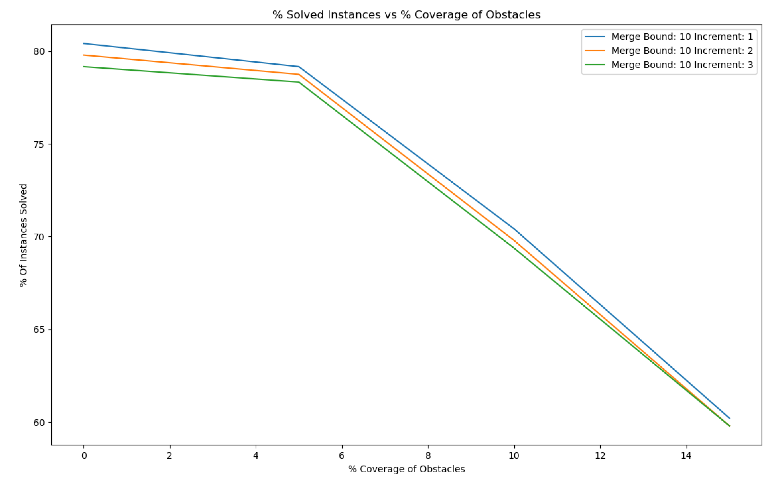


Figure 7: An increase in the number of instances solved is found when the deepening merge bound for RMA-CBS(1) is increased.

Figure 8: RMA-CBA(5) also solves more instances when its deepening merge bound is increased. Although this is not consistent for all densities of obstacles.

Figure 9: The number of instances solved by decreases for RMA-CBS(10) when increasing the deepening merge value.

Figure 10: RMA-CBS(20) loses performance when its merge deepening value is increased for all densities.

Again however, increasing the deepening merge bound increased performance on low initial merge bound algorithms. For example, Figure 12 shows that RMA-CBS(1) increased performance by approximately 5% simply by increasing the deepening merge bound to 2 despite RMA-CBS(10) not receiving the same benefits. Secondly, it shows that as the size of the instance gets larger, RMA-CBS(10) only improves solving rates to about 78%, whereas RMA-CBS improves rates up over 80%. This begs the question if the instance gets larger, where would the lower starting merge bound algorithm taper off its performance rate. This leaves room to experiment in terms of size of map as well as the deepening merge bound increase.

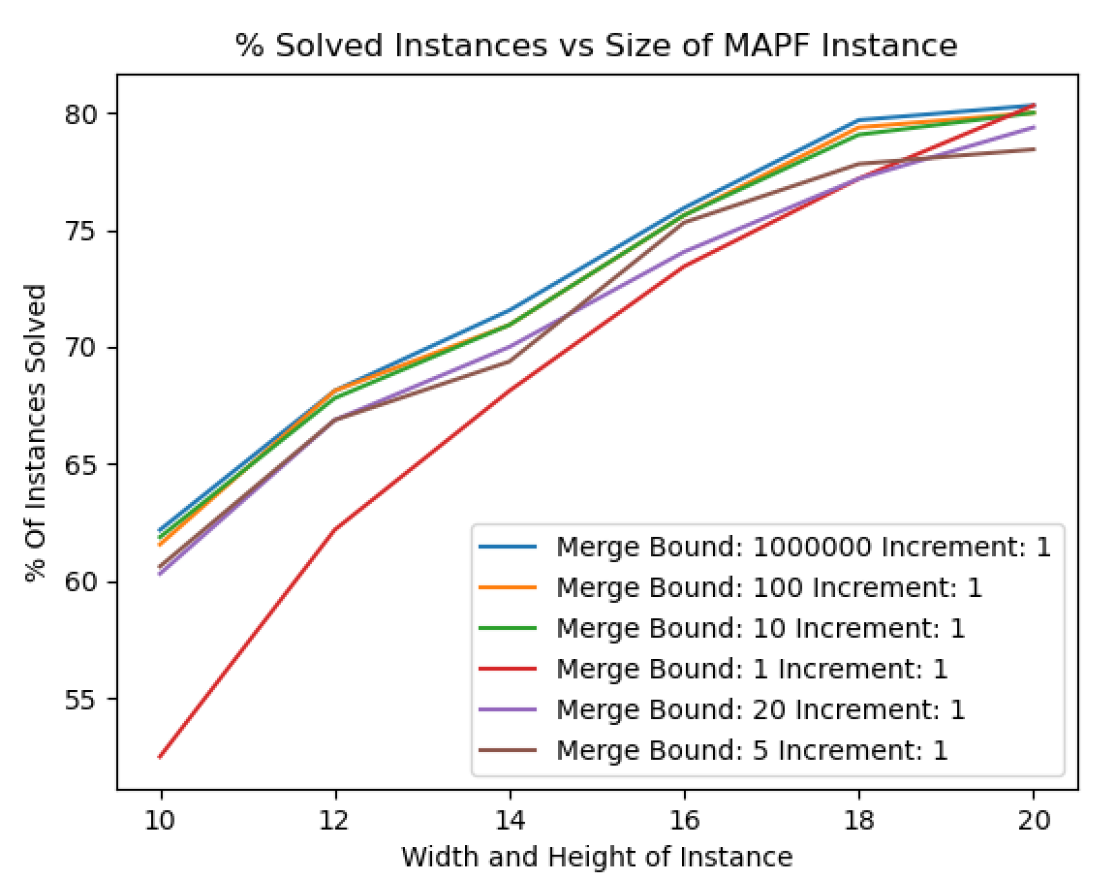


Figure 11: An increase in the size of the instance vs the size of the instance.

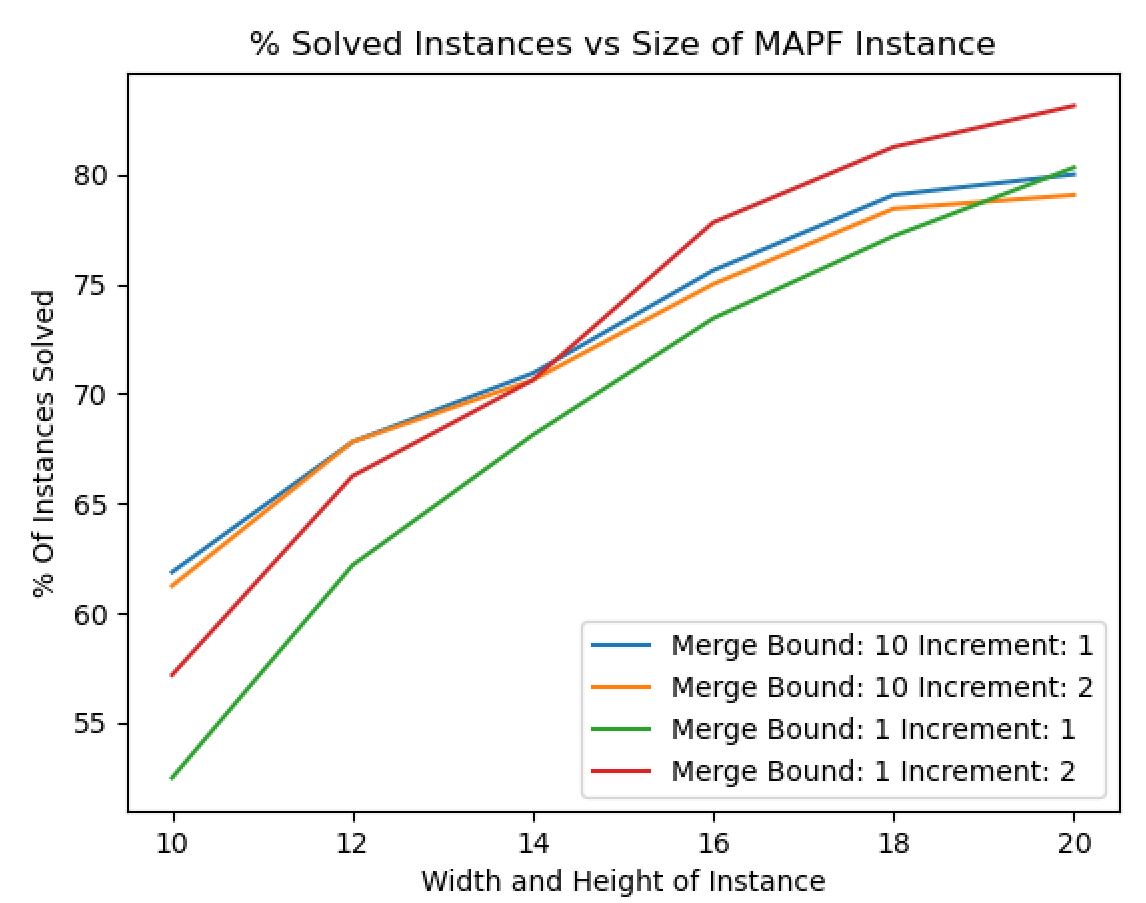
.

Figure 12: Again, RMA-CBS(1) gets a large increase in performance when increasing the merge bound, although it performs much worse relative to RMA-CBS(10) for instances that are smaller in size.

# Conclusions

Overall, recursive meta-agent conflict based search with a finite merge bound (B) and a merge bound deepening increase of 1, 2, or 3 did not solve as many instances as CBS, or RMS-CBS() within the time limit. Simply, the overhead of merging and generating the RMA-CBS lower-level solver was too slow to compensate for the speed in solving the simplified instance. Comparing this to the conclusions drawn from [2], RMA-CBS does not enjoy the benefits of a A\* type lower-level search. In a sense, RMA-CBS merges when it may be beneficial to perform some A\* type search, but instead instantiates its own RMA-CBS solver that has the same strengths and weaknesses as it high level solver. RMA-CBS does not benefit from the variety and complimentary behaviour of CBS and A\* search together.

Despite this, when increasing the merge bound deepening value, RMA-CBS solvers with initially low values for merge bounds gained significant improvements in performance. This was particularly consistent across changes in the density and size of the map. Future research on RMA-CBS could attempt to solve some of the following questions:

1. How does a more complicated deepening merge bound affect the performance of the solver?
2. What if the map size and density are changed (and made larger)?
3. Are there specific instances where RMA-CBS significantly outperforms pure CBS?
4. If given more time to solve instances, would a greater difference between algorithms emerge?

Nonetheless, RMA-CBS may not be the most effective way to solve MAPF instances, but it can be done if the need arises in the future.

# References

[1] Q. Sajid, R. Luna and K. Bekris, *Multi-Agent Pathfinding with Simultaneous Execution of Single-Agent Primitives*. Association for the Advancement of Artificial Intelligence, 2021, pp. 1-4.

[32] G. Sharon, R. Stern, A. Felner and N. Sturtevant, *Artificial Intelligence 219 (2015)*. ELSEVIER, 2015, pp. 40-66.

[3] J. Yu and S. Lavalle, *Planning optimal paths for multiple robots on graphs*. Karlsruhe, Germany: IEEE, 2013, pp. 3612-3617.