**Conclusion**

This thesis presented an overview and definition of the multi-agent problem (Chapter ?), two solutions to the MAPF and MAPD problems (Chapter ??), a generalizable approach to adapting solutions to the problems for implementation (Chapter ???), and a workflow consisting of two computer applications which ensures a level testing field for multiple algorithms (Chapter ????). The stated purpose of the work was to simplify the testing process, minimize knowledge requirements when comparing algorithms for performance, provide an improvement in visualization over existing solutions, and expose additional data to the user to enable deeper analysis.

Using the implementation strategy of Chapter ???, FleetBench was developed. Its basic function as a state machine and user interface for creating test cases was extended to include execution of two families of algorithms. FleetBench and GraphRendering were used to generate several arbitrary test cases in a fast and intuitive process. Algorithms belonging to each of the two families presented in Chapter ?? were then tested against these test cases. This process demonstrated known and hypothesized behavior from research in the field within the implementation, affirming the accuracy of the implementation. Features of FleetBench proved useful in visually confirming the behavior of the algorithms, enabling more intuitive analysis. Statistical data collected during the simulation was used to fuel analysis and springboard the development of explanations for the behavior of the system in simulation. Taken together, these represent success—the intended goals for the research and implementation work were achieved.

**Future Work**

The algorithms implemented in this paper are of a certain type: they operate with similar methods of storing or processing data and generally rely on the optimality of the A\* algorithm for pathfinding. Other approaches exist in research which may not integrate as smoothly with the framework presented in Chapter ??? as WHCA\* or TPTS. For example, BIBOX [1] and Push and Swap [2] solve multi-agent problems through topological analysis of the system state. Centralized approaches such as CBS [3] which seek optimal moves at each step were said to be difficult to scale, and have not been implemented in FleetBench. New directions in the research make use of Machine Learning to apply pattern-based solutions, as in PRIMAL [4]. While the framework presented in this work can be used as a target for adapting the function of each of these solutions, it may become simpler for the user to adjust the presented state machine to better suit these approaches.

In terms of performance, FleetBench is far from a perfect application. While it performs the stated objective with sufficient performance in relatively simple and small cases, it suffers from performance issues as the scale increases. As discussed in the background research, real-world warehousing problems span enormous spaces and field thousands of robots to aid in fulfillment of tasks. Analogizing such problems within FleetBench would prove extremely difficult—even for powerful computers—due to a lack of performance optimization in the current implementation.

FleetBench could also be made smarter, though this is largely up to the user extending its function. If conclusions can be drawn about the excellence of an algorithm in solving a particular class of multi-agent problem, analysis could be performed before testing the algorithm to indicate an algorithm may be of particular interest to the user.

Additional statistical tracking could be implemented to deepen analysis. For example, a user may be interested in the rate of task completion throughout the simulation. A significant drop could indicate a bottleneck, allowing a user to easily identify the simulation timesteps leading up to the performance throttle.

The extensibility of FleetBench is ultimately a subjective experience but changes could be made which make the process smoother. Of particular interest would be dynamic option generation, as currently a user extending the program is required to implement their own UI code, executed at runtime. Currently, the component scripts are broken out for modularity. It is possible that a user would prefer to instead supply a library in any format they prefer to develop, which is an approach currently not supported by FleetBench.

Most of all, every additional algorithm which is correctly added to the baseline functionality of FleetBench increases its value as a testing tool, saving effort and time for the end user.

[1] P. Surynek, “A novel approach to path planning for multiple robots in bi-connected graphs,” in *2009 IEEE International Conference on Robotics and Automation*, Kobe: IEEE, May 2009, pp. 3613–3619. doi: 10.1109/ROBOT.2009.5152326.

[2] R. Luna and K. E. Bekris, “Efficient and complete centralized multi-robot path planning,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA: IEEE, Sep. 2011, pp. 3268–3275. doi: 10.1109/IROS.2011.6095085.

[3] G. Sharon, R. Stern, A. Felner, and N. Sturtevant, “Conflict-Based Search For Optimal Multi-Agent Path Finding,” *Proc. AAAI Conf. Artif. Intell.*, vol. 26, no. 1, Art. no. 1, 2012, doi: 10.1609/aaai.v26i1.8140.

[4] G. Sartoretti *et al.*, “PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning,” *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 2378–2385, Jul. 2019, doi: 10.1109/LRA.2019.2903261.