**Background**

Mobile robotics is a rapidly maturing field with a wide range of applications including industrial settings [1], space exploration [2], and medical applications [3]. Significant advancements have been made in robot capabilities, manufacturing costs, and safety to enable the integration of robots in a growing number of settings.

One setting has garnered particular attention in recent years, forming into a multi-billion-dollar industry: mobile warehouse robots. State of the art research and development in this area has been ongoing for some time at companies such as Boston Dynamics [4] and Agility Robotics [5], who seek to push the capability envelope of humanoid robots with their Atlas and Digit robots. Currently, however, the industry is dominated by simpler robots which are purpose-built for shifting specific loads in fulfillment centers by companies such as Amazon Robotics and Alibaba. Amazon fields over 750,000 robots in its fulfillment facilities, forming a complex problem of robot and human cooperation which must be solved continuously to deliver up to 1 billion packages per year [6]. Such processes are typically viewed as the way of the future for warehousing projects [7], and Amazon, while the largest company, is far from the only group pursuing implementation of such automated systems in fulfillment centers [8].

A key problem faced in warehouse robotics is that of finding appropriate paths and trajectories of robots (also called agents) in densely populated spaces with high throughput requirements. Similar instances of this problem can be found in a great number of fields: videogames [9], search problems including rescue and evacuation [10], and air-traffic management [11]. This problem has received significant attention from both robotics and artificial intelligence researchers, becoming well characterized but remaining definitively unsolved [10].

The problem of guiding a fleet of agents to a set of individual targets in an efficient and collision-free manner through an environment is called the Multi-Agent Pathfinding (MAPF) problem [12]. Each instance of the problem is defined by a number of agents, each with their own goal, located in a defined space with known movement costs. The solution therefore optimizes the cost function of movement within the system with the constraint that there be no collisions and that all agents finish on their target locations. Typically, the cost in these instances is time, although it is sufficiently trivial to augment the cost functions with fuel costs or other abstract measures of performance during implementation.

The MAPF problem is considered to be a “single shot” problem and solution. Agents will move to their targets and simply remain there so long as no collisions are caused by doing so. This falls short of being analogous to real-world warehousing applications, where it is desirable for agents to chain together sequences of optimal movement for various tasks such as pickup and delivery. Therefore an extension of the MAPF problem to make it “lifelong” is developed, called the Multi-Agent Pickup and Delivery (MAPD) problem [13].

The MAPD problem augments agent targets to be tasks with a pickup location and a delivery location. The path of an agent completing its objective now involves movement to the delivery location via a path which includes the pickup location. New tasks are also periodically introduced to the system, and assignments to agents are made dynamically. Success is measured when all tasks in the system are marked as complete, with the key indicator of performance being the service time per task. New challenges are introduced in the form of avoiding deadlock and cycling behaviors, where agents repeatedly move through the same motions without advancing toward their goals. Resiliency against disturbances is also of elevated concern [14]. This formulation of the problem more accurately captures challenges in real-world applications but requires different approaches and applies different constraints that make MAPF-solving algorithms ill-suited to the task without adaptation.

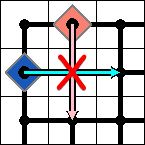


Figure 1: An instance in which two agents who have found best paths to target nodes using single-agent techniques would collide, rendered in FleetBench.

Single-agent pathfinding problems where there are no other moving objects in the space are trivial; a solution is known and returns provably optimal paths via the famous A\* algorithm [15]. This approach cannot be readily extended for multiple agents moving simultaneously in the system space. Agents need to use knowledge of each other’s positions and intent to avoid collisions during plan execution. More complex strategies are needed.

Solutions to the MAPF and MAPD problems generally fall into one of two categories: centralized, where plans for agents are created in parallel, or decoupled, where agents form their plans sequentially, avoiding collisions with the plans of other agents in the system.

Centralized approaches such as CBS [16], PRIMAL [17], and Push and Swap [18] generally produce solutions by searching a huge variety of motion options at every step. Thoroughly searching the valid state space where every agent represents a degree of freedom for possible state changes represents an immense endeavor, but the end result is a guarantee that the path is collision-free (complete) and the fastest possible solution (optimal) [18]. However, this approach is computationally intense, even for small scenarios, and is simply not scalable to larger applications [10]. If computation time begins to exceed action time systems will begin to lag, resulting in cascading underperformance as additional objectives are introduced. Advancements in this area often involve trying to prune the search space [18], reducing search time.

Decoupled approaches such as Windowed Hierarchical Cooperative A\* [19] and Token Passing [13] tend to experience the opposite; while the search space is greatly reduced by focusing on a single agent at a time—making the problem tractable at large agent counts—the solution found cannot be guaranteed to be optimal, or even complete unless certain preconditions are met [13]. For example, an agent can plan a path which cuts off another agent’s access to its target, forcing excessive wait times or resulting in a failure to solve the problem. The lack of such guarantees makes the algorithms unreliable, and there often are not strong criteria which can identify problem instances that are solvable among those which are not [20]. In these cases, it is difficult to justify implementation in real-world systems as-is.

**Motivation**

Several factors contribute to difficulties in studying these algorithms, whether one is testing an implementation of an algorithm which exists in the research or designing a novel approach to the problem. This problem is especially difficult for those not already well-versed in the field and may act as a deterrent for industry implementation, hobbyist engagement, or prospective researchers.

By and large, practical implementations of algorithms found in research are not instantly accessible—the work must be requested from authors or produced by the reader. In the latter case, the reader will need expertise in programming in order to implement both their own test cases and the algorithm itself. Further difficulties arise when attempting to produce visualizations. In both cases, there is a lack of standardization—algorithms could be implemented in any language (though many are done C++), on any test case, with any style of input and output of data formatting. Available source code, particularly for data structures, is often poorly documented which adds further difficulty to the process of adjusting an algorithm.

The presentation of data in research often falls into evaluation of categories such as *makespan* (maximum arrival time), *flowtime* (total time loss), or a count of the number of successfully completed tasks over a defined period of time. While these data are useful in developing general notions of success, further optimization is likely to lie in dealing with edge cases which an algorithm handles poorly. Such situations may be washed out in a longitudinal study spanning hundreds or thousands of instances. In these cases, it is useful to have access to historical data in the problem solution in order to investigate the interactions which produce the problem, and therefore develop ideas about restrictions or potential augmentations to the algorithm. This will necessitate implementation of various statistical tracking methods across every selected algorithm, further consuming the researcher’s time.

When benchmarking the performance of an algorithm it is important that similar conditions are used in order to approach evaluating test results. The underlying characteristics of any multi-agent problem directly drive the performance of all algorithms. Minor changes produce cascading effects which result in extremely different outcomes. For decoupled systems, varying the agent activity order has a pronounced effect on the solutions found, and with no clear predictor that can identify when this would matter analysis is restricted to tedious variation of parameters. Topology of the system creates very different opportunities for solutions, such as in the case of BIBOX, which requires that the graphical abstraction of the system space be bi-connected [21]. The order in which tasks are inserted to the problem has significant downstream effects. Furthermore, the way in which they are assigned leaves open another axis for optimization. For example, assigning tasks in proximity order to the current agent may be efficient for the individual agent, but the system as a whole may suffer from the increased traffic in a region that other agents need to pass through. Small adjustments produce very different results which may obfuscate the behavior of algorithms in other situations. Due to non-standardized use of data structures, input and output formats, and use of different programming languages, it is difficult to assure equal playing fields for each algorithm.

Some strides have been made in these areas, defining a set of benchmark test maps (and map types) as well as strategies for generating tasks [12]. However, there is a reliance on large test sizes and pseudorandom generation which may not conform to real-world use-cases. There are also a number of publicly accessible code repositories which contain implementations of one or more algorithms, again in the author’s preferred programming language and style. Some repositories are implemented in the Godot Videogame Development engine [22], while others have implementations in raw C++ [23], [24] or Python [25], [26]. In all cases, it is not clear how the function of such implementations could be readily extended while maintaining a standard procession of logic across each implementation. Development of test cases is not done via any guided process, inviting errors at each step. Extension of the programs to include additional algorithms is not a straightforward process and often times it is unclear whether any behaviors are shared across implementations of each algorithm in the first place. The collection of empirical data in such works requires further modification much of time.

In an effort to bridge these gaps between theoretical knowledge and practical test implementation, Chapter ? presents a framework through which decoupled MAPF and MAPD algorithms may be implemented in a manner which separates the different axes of optimization and allows for the insertion of various real-world constraints. The implementation strategy is designed to be dynamic and flexible, while anchoring key facets present in most algorithms. With this approach, algorithms of any type may be implemented into one coherent system, ensuring ease of test and iteration. This represents the primary intent of the manuscript.

As proof of concept, Chapter ?? presents a pair of programs which implement the strategy for a selection of algorithms which exist in research. The first novel program, called GraphRendering is a utility for generating the system maps over which a fleet of agents operates. The second application, called FleetBench, provides a graphical user interface (GUI) which enables intuitive implementation of test cases and data collection for the end user, while exposing the internals in an explainable and modular fashion for developers and researchers who are creating algorithms. Visualization and tabulation of results is provided via library-like functions which may be used from within algorithm scripts. Layers of abstraction are used to generalize the problem, so that with careful planning most situations may be represented. Configurable options are implemented to allow approximation of real-world constraints where possible. A state history is made accessible to aid in identification of problematic states. An auxiliary program is also provided, which allows for rapid graphical generation of maps.

Together, these two applications create a workflow for a user to rapidly test algorithms on a variety of test cases with various restrictions. The user will be able to quickly tweak aspects of their test case such as the map layout, task ordering, and restrictions on agents when the user chooses to investigate performance bottlenecks. Data for three test cases of interest are presented in Chapter ??? as evidence that the approach and application are useful for analysis.

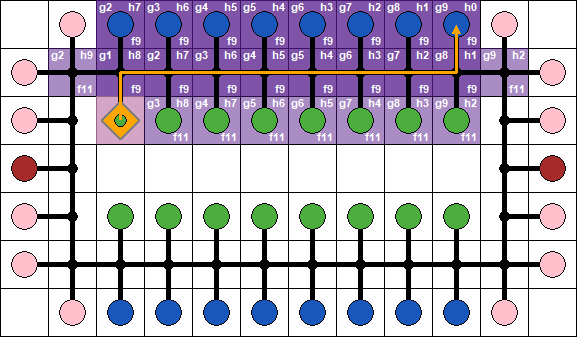


Figure 2: An agent seeks a path to a node through an abstraction of space using the A\* algorithm, rendered in FleetBench. The map represents an MAPD situation, where blue nodes are for pickup, and green are for deposit.

The applications are provided as-is in a public repository. Reference documentation and a guide for extending functionality are located in the appendices of the manuscript.

[1] “Industrial robots,” KUKA AG. Accessed: Nov. 05, 2023. [Online]. Available: https://www.kuka.com/en-us/products/robotics-systems/industrial-robots

[2] mars.nasa.gov, “Summary | Rover,” NASA Mars Exploration. Accessed: Nov. 05, 2023. [Online]. Available: https://mars.nasa.gov/msl/spacecraft/rover/summary

[3] “5 Medical Robots Making a Difference in Healthcare,” CWRU Online Engineering (Live). Accessed: Nov. 05, 2023. [Online]. Available: https://online-engineering.case.edu/blog/medical-robots-making-a-difference

[4] B. van der List, “Boston Dynamics wants to change the world with its state-of-the-art robots,” Strategy+business. Accessed: Nov. 05, 2023. [Online]. Available: https://www.strategy-business.com/article/Boston-Dynamics-wants-to-change-the-world-with-its-state-of-the-art-robots

[5] J. Dorrier, “Agility’s New Factory Can Crank Out 10,000 Humanoid Robots a Year,” Singularity Hub. Accessed: Nov. 05, 2023. [Online]. Available: https://singularityhub.com/2023/09/20/agilitys-new-factory-can-crank-out-10000-humanoid-robots-a-year/

[6] “How Amazon deploys robots in its operations facilities.” Accessed: Nov. 05, 2023. [Online]. Available: https://www.aboutamazon.com/news/operations/how-amazon-deploys-robots-in-its-operations-facilities

[7] “Automation, robotics, and the factory of the future | McKinsey.” Accessed: Nov. 05, 2023. [Online]. Available: https://www.mckinsey.com/capabilities/operations/our-insights/automation-robotics-and-the-factory-of-the-future

[8] J. Prisco, “Why online supermarket Ocado wants to take the human touch out of groceries,” CNN. Accessed: Nov. 05, 2023. [Online]. Available: https://www.cnn.com/2021/04/26/world/ocado-supermarket-robot-warehouse-spc-intl/index.html

[9] O. Majerech, “Solving Algorithms for Multi-agent Path Planning with Dynamic Obstacles”.

[10] O. Salzman and R. Stern, “Research Challenges and Opportunities in Multi-Agent Path Finding and Multi-Agent Pickup and Delivery Problems,” *N. Z.*, 2020.

[11] D. Šišlák, P. Volf, and M. Pěchouček, “Agent-Based Cooperative Decentralized Airplane-Collision Avoidance,” *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 1, pp. 36–46, Mar. 2011, doi: 10.1109/TITS.2010.2057246.

[12] R. Stern *et al.*, “Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks,” *Proc. Int. Symp. Comb. Search*, vol. 10, no. 1, pp. 151–158, Sep. 2021, doi: 10.1609/socs.v10i1.18510.

[13] H. Ma, J. Li, T. K. S. Kumar, and S. Koenig, “Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks.” arXiv, May 30, 2017. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/1705.10868

[14] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, “Beyond Robustness: A Taxonomy of Approaches towards Resilient Multi-Robot Systems.” arXiv, Sep. 25, 2021. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/2109.12343

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[16] 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.

[17] 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.

[18] 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.

[19] D. Silver, “Cooperative Pathfinding,” *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, vol. 1, no. 1, pp. 117–122, Sep. 2021, doi: 10.1609/aiide.v1i1.18726.

[20] K.-H. C. Wang and A. Botea, “MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees”.

[21] 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.

[22] N. Franke, “gdmapf.” Sep. 14, 2023. Accessed: Nov. 05, 2023. [Online]. Available: https://github.com/nathanfranke/gdmapf

[23] K. Okumura, “mapf-IR.” Nov. 01, 2023. Accessed: Nov. 05, 2023. [Online]. Available: https://github.com/Kei18/mapf-IR

[24] E. Lam, “BCP-MAPF.” Oct. 13, 2023. Accessed: Nov. 05, 2023. [Online]. Available: https://github.com/ed-lam/bcp-mapf

[25] A. Bose, “Multi-Agent path planning in Python.” Nov. 04, 2023. Accessed: Nov. 05, 2023. [Online]. Available: https://github.com/atb033/multi\_agent\_path\_planning

[26] H. Peng, “Multi-Agent Path Finding.” Nov. 05, 2023. Accessed: Nov. 05, 2023. [Online]. Available: https://github.com/GavinPHR/Multi-Agent-Path-Finding