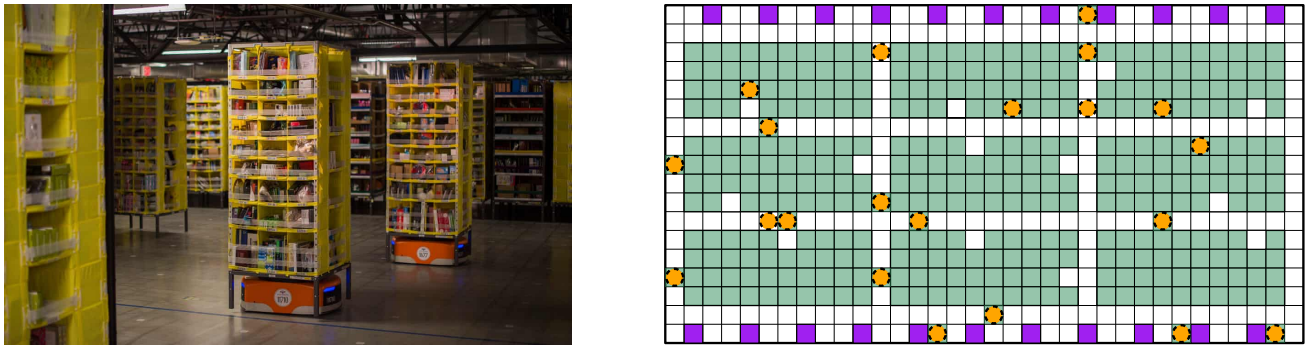
Project description—Multi-objective MAPF

Background

Coordinating the movement of a fleet of agents or robotsis a decades-old family of problems that has been intensively studied by the robotics and artificial intelligence communities. Applications of this family of problems can be found in diverse settings including assembly [7], evacuation [9], formation [8] and search-and-rescue [6].

One specific application of this general problem that gained significant interest in the research community is the warehouse domain. Here, storage locations host inventory pods that hold one or more kinds of goods. A large number (several hundreds and some times even several thousands) of robots operate autonomously in the warehouse picking up, carrying and putting down the inventory pods. The robots move the pods from their storage locations to designated dropoff locations where the needed goods are manually removed from the inventory pods (to be packaged and then shipped to customers). Each pod is then carried back by a robot to a (possibly different) storage location [13]. The successful use of such robots in warehouse applications led to a multi-billion industry led by tech-giants such as Amazon robotics and Alibaba [1]. For a visualization, see Fig. 1.



(a) (b)

Figure 1: Warehouse robots. (a) Amazon robots (orange) moving pods (yellow) containing goods in a warehouse environment. Figure adapted from `http://tiny.cc/ief6cz`. (b) Typical layout of pods in a warehouse. Pods and dropoff locations are depicted by green and purple squares, respectively. Agents are depicted using orange circles. Notice that some agents carry the pods and some don't.

In this work we are interested in one type of problem that can be used to model this application (as well as many others) called *Multi Agent Path Finding* (MAPF). Here, we are given a graph $G = (V, E)$ which, in our motivating example, is a discretization of the warehouse into cells where each cell represents a graph vertex and two vertices are connected in the graph if their corresponding cells are adjacent and do not contain pods. In addition, we are given $s : [1, \ldots, k] \to V$ and $t : [1, \ldots, k] \to V$

which map each one of the k agents to a start and target vertex, respectively. Time is typically discretized as well and at each time step an agent performs one of two *actions*: it can either wait in its cell or move to an adjacent cell. An action is considered *conflict free* or valid, if two agents do not occupy the same vertex or the same edge at a given timestep. The objective is to find conflict-free paths for the agents from their start to their target locations that minimize some objective. Typically, we wish to minimize the *makespan* which is the latest arrival time of an agent at its target location or the *flowtime* which is the sum of the arrival times of all agents at their target locations.[1]

Problem statement

In this project we will explore the problem of computing solutions to the MAPF problem that balance between the makespan and the flowtime. This is a specific instance of a general family of problems called *Bi-criteria optimization* [4, 10]. Here we wish to compute all solutions where no solution is strictly better then the others for both cost functions, a set called the *Pareto-optimal frontier*.

Algorithmic background Before we describe our approach, we provide some algorithmic background on state-of-the-art MAPF algorithms. One can apply A* [5] to optimally solve the MAPF problem by treating the fleet of agents as one composite system. However, this approach does not scale with the number of agents as the search space and branching factor is exponential in the number of agents. An alternative state-of-the-art approach that is not based (directly) on A* is Constraint-Based Search (CBS) [2, 3, 11]. In the basic version of CBS, agents are associated with constraints indicating that at a given timestep, an agent cannot occupy a given vertex. The algorithm builds a *Constraint Tree* which is a binary tree where each node contains a set of constraints, and a cost of a solution that is not in conflict with theses constraints. Given a node in the Constraint Tree, a low-level search (e.g., an A*-based search) is run for the individual agents that is consistent with the constraints (namely, constraints are treated by the search as moving obstacles). Once a consistent path has been found for each agent by the low-level search, these paths are validated with respect to the other agents by simulating the movement of the agents along their planned paths. If all agents reach their target without any conflict, the node is declared as the goal node. If a conflict is found for two (or more) agents, the validation halts and the node is split by adding two new nodes with new constraints.

Bi-criteria MAPF using CBS. To compute the Pareto-optimal frontier we will use the CBS algorithm. Consider running the CBS algorithm without a termination condition (but without expanding CT nodes that hold valid solutions). It is not hard to see that this algorithm will compute the set of Pareto-optimal solutions (as well as many other solutions). Implement this approach and add some pruning logic to avoid expanding parts of the search tree that will not lead to solutions in the Pareto-optimal set. Can you add heuristics to make the algorithm more efficient? You do *not* need to write

[1] For a complete taxonomy of different MAPF problems, including conflicts types, objective functions and more, see [12].

everything from scratch. C++ implementations of MAPF algorithms and benchmarks can be found at `http://mapf.info/index.php/Main/Software`.

References

[1] Three ways Amazon, Alibaba, and Ocado benefit from warehouse robots. `https://www.innovecs.com/ideas-portfolio/amazon-alibaba-ocado-use-warehouse-robots/`. Accessed: 19-07-19.

[2] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *ECAI*, volume 263, pages 961–962, 2014.

[3] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Solomon Eyal Shimony. ICBS: improved conflict-based search algorithm for multi-agent pathfinding. In *IJCAI*, pages 740–746, 2015.

[4] Altannar Chinchuluun and Panos M Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154(1):29–50, 2007.

[5] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems, Science, and Cybernetics*, 4(2):100–107, 1968.

[6] James S Jennings, Greg Whelan, and William F Evans. Cooperative search and rescue with a team of mobile robots. In *International Conference on Advanced Robotics (ICAR)*, pages 193–200, 1997.

[7] Bartholomew O Nnaji. *Theory of automatic robot assembly and programming*. Springer Science & Business Media, 1993.

[8] Sameera Poduri and Gaurav S Sukhatme. Constrained coverage for mobile sensor networks. In *ICRA*, volume 1, pages 165–171, 2004.

[9] Samuel Rodriguez and Nancy M Amato. Behavior-based evacuation planning. In *ICRA*, pages 350–355, 2010.

[10] Oren Salzman. Approximate bi-criteria search by efficient representation of subsets of the pareto-optimal frontier. *CoRR*, abs/2006.10302, 2020.

[11] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent path finding. In *AAAI*, 2012.

[12] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *SOCS*, pages 151–159, 2019.

[13] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–9, 2008.