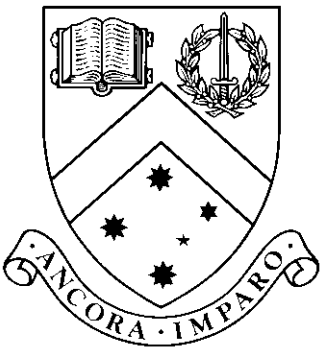
School of Computer Science
Monash University




Literature Review, 2017

# Improving cooperative pathfinding using a path oracle

Phillip Wong 25150510

Supervisors: Daniel Harabor,
Pierre Le Bodic

# Contents

**Abstract**

The order picking process is the number one expense in the operating cost of warehouse systems. This project will look at *part-to-picker*, a method of order picking where orders are retrieved and delivered to a number of picking areas located around the warehouse. Previous research has improved on multi-agent path finding (MAPF) algorithms but mostly overlooked the potential benefits gained by configuring the warehouse layout. Here, we will be exploring Kiva systems a part-to-picker system which uses autonomous vehicles and mobile storage. Our focus is to explore a number of adjustments and additions which we expect will greatly affect how we design warehouse layouts. These include: introducing an intermediate dropping zone, optimizing order processing and adding the capability for robots to maneuver under storage pods. The results of this project will help identify how we should position storage and picking stations in a warehouse. Additionally, we will be looking at developing a MAPF method which uses a pre-computed path oracle.

**Keywords**
Cooperative Multi-agent pathfinding, Kiva systems

# 1 Introduction

Order picking is a process in warehouse systems whereby products are retrieved from storage to satisfy incoming customer orders. This process has been identified by De Koster et al. (2007) as the most expensive process in operating a warehouse, estimated to take 55% of the warehouse operating cost.

Here we look at a method of order picking known as part-to-picker systems. Part-to-picker systems contain multiple picking stations located around the warehouse. Products are brought to picking stations where workers will manually pick and process the product. One of the disadvantages of part-to-picker systems is that there will be some downtime at the picking stations while waiting for an order to be delivered. To solve this, these systems often use an automated storage and retrieval system (AS/RS). AutoStore (2015) is a recent part-to-picker system where products are organized in a grid of stacked bins. Robots move around the top of the grid, lifting bins and delivering them to picking areas. Benefits of the AutoStore system include high storage density and expansion capability. While not much literature is published about the specifics of AutoStore, we suspect the major downsides are: slow, expensive order retrieval as well as high infrastructure and maintenance costs.

In this project, we look at Kiva Systems (now known as Amazon Robotics). In Kiva systems, products are stored in mobile shelves known as storage pods. Robots known as drive units are responsible for retrieving and delivering storage pods to picking stations. A human worker is stationed at each picking station who picks the item off the pod before processing it (Fig **??**). Once the pod has been processed, the drive unit will return the pod to an appropriate location in the warehouse.

Kiva systems do not require a complex infrastructure to operate, a warehouse needs only a suitible number of storage pods, picking stations and drive units to operate. As long as the warehouse has space, more robots, pods or stations can be easily be added to the system to satisfy the incoming flow of customer orders. When a drive unit malfunctions it can be easily accessed and replaced. In summary, the main benefits of Kiva systems are their low initial and maintenance costs and their rapid deployment and flexibility (Wurman et al. (2008)).

### 1.1 Research questions

We aim to explore two areas of Kiva systems, the layout and MAPF. These can be summarized in the following questions:

1. How will the adjustments and additions below affect our decision when it comes to configuring the warehouse layout?
   - Adding an intermediate zone where drive units may drop off storage pods
   - Adding the capability for drive units to maneuver under storage pods
   - Implementing an optimized order process
2. How much faster will the MAPF search run by pre-computing paths and storing them in a path oracle?

## 2 Background

In Kiva systems, we face a multi-agent pathfinding (MAPF) problem. MAPF aims to find a path for each agent to their goal while ensuring that no path conflicts with another. MAPF has usage in video games, robotics (Bennewitz et al. (2002)), search and rescue (Konolige et al. (2006)) and warehouse applications. When analyzing the efficiency of a MAPF algorithm we generally aim to reduce the makespan of the system. Additionally, in Kiva systems we want to reduce the downtime of picking stations.

Finding an optimal solution in MAPF is an NP-hard problem (Yu and LaValle (2013)) and mostly has found usage in systems containing a small number agents. This is not an option as Kiva systems deal with hundreds of agents, for example the Office Supply company, Staples uses 500 robots in their $30000m^2$ center (Guizzo (2008)). Here we look at finding a bounded suboptimal solution and this has been explored in Kiva systems by (Cohen and Koenig (2016)).

To improve MAPF, generally methods are created to simplify the problem, Cohen and Koenig (2016) define user-provided highways to help guide agents towards a specific direction, greatly reducing the chance of path collisions. Wilt and Botea (2014) identifies bottlenecks in the environment and assigns a controller which handles agents who want to pass through the bottleneck, simplifying agent behaviour in high collision zones. Another common technique is grouping agents into teams. Ma and Koenig (2016) splits agents into teams of 5 and presents a Conflict-Based Min-Cost-Flow algorithm which and shows that they can achieve a correct, complete and optimal solution.

Specific to the process of order picking, we will look at the method of order processing. Take an example where products of the same type are grouped together in a warehouse. If a large order of one product comes in, the agents will all try to find a path to this one area and create many collisions in the MAPF. We want the goal locations for our drive units to be spread evenly around the warehouse and order processing allows this by looking at two areas. Firstly, by evenly distributing products around the warehouse. If we place products of the same type across many different row around the warehouse, a large order of one product will be no issue. Secondly, is sequencing of incoming orders. Instead of processing the large orders of one product sequentially, we have some flexibility to interlace this large order with other orders which we know we will need to process. Essentially, we can move the mobile storage pods as well adjust the incoming ordering sequence to benefit the MAPF. Boysen et al. (2017) looks at both these aspects in unison and found that with optimized order processing, only half the units are required to provide the supply given by a non-optimized system.

## 2.1 Improving search times

### 2.1.1 Compressed Path Databases

### 2.1.2 Jump Point Search

Renukamurthy (2016)

## 2.2 Comparing MAPF algorithms

**Completeness:** if a path to the goal exists, it is always found
**Solution quality:**

- Optimal: finds the shortest path to the goal
- Bounded suboptimal: finds a path to the goal with path length less than some bound?
- Suboptimal: finds a path to the goal with no guarantee of path length

**Anonymous:** which agent goes to which goal is interchangeable
**Time complexity:**

- Non-Polynomial (NP) / Intractable
- Polynomial / Tractable

**Continuous???:** Does the entire search need to be recalculated if a new agent is added into the system (does it look at one batch of units at a time, it cannot handle a continuous stream)
**Centralized / Coupled:** *A centralised approach (Barraquand & Latombe 1991; LaValle & Hutchin- son 1998) has a single global decision maker for all agents, is theoretically optimal but, as discussed before, it does not scale up to many agents due to a prohibitive complexity. A decoupled (decentralized) approach decomposes the problem into several subproblems. The latter approach is faster but yields suboptimal solutions and loses the completeness* Wang et al. (2008)
**Online / Offline** Online is when you plan while you move?

| Search | Centralised | Complete | Tractable | SQ | Anon |
|---|---|---|---|---|---|
| WHCA* | N | N | N | O | No |
| FAR | N | Y | Y | BO | No |
| MAPP | N | N | O | BO | No |
| CBS | N | N | O | O | |
| BCP | N | | | | |
| Centralised A* | Y | | | | |

*Figure 1: Comparison of MAPF algorithms*

## 2.3 Cooperative Multi-agent pathfinding

When looking at multi-agent pathfinding, we will be first considering whether it is centralized. A centralized search usually indicates

### 2.3.1 CA*, HCA* and WHCA*

Silver (2005)

#### Cooperative A*

*The individual searches are performed in three dimensional space-time, and take account of the planned routes of other agents. A wait move is included in the agents action set, to enable it to remain stationary. After each agents route is calculated, the states along the*

*route are marked into a reservation table. Entries in the reservation table are considered impassable and are avoided during searches by subsequent agents.*

#### Hierarchical Cooperative A*

*Hierarchical Cooperative A* (HCA*) uses a simple hierarchy containing a single domain abstraction, which ignores both the time dimension and the reservation table. In other words, the abstraction is a simple 2-dimensional map with all agents removed. Abstract distances can thus be viewed as perfect estimates of the distance to the destination, ignoring any potential interactions with other agents. This is clearly an admissible and consistent heuristic.*

#### Windowed Hierarchical Cooperative A*

*The cooperative search is limited to a fixed depth specified by the current window. Each agent searches for a partial route to its destination, and then begins following the route. At regular intervals (e.g. when an agent is half-way through its partial route) the window is shifted forwards and a new partial route is computed*

### 2.3.2 FAR

Wang et al. (2008)

*When building a search graph from a grid map, FAR implements a flow restriction idea inspired by road networks. The movement along a given row (or column) is restricted to only one direction, avoiding head-to-head collisions. The movement direction alternates from one row (or column) to the next. Additional rules ensure that two locations reachable from each other on the original map remain connected (in both directions) in the graph. After building the search graph, an A* search is independently run for each mobile unit. Dur- ing plan execution, deadlocks are detected as cycles of units that wait for each other to move. A heuristic procedure for deadlock breaking attempts to repair plans locally, instead of running a larger scale, more expensive replanning step.*

### 2.3.3 MAPP

Wang and Botea (2011)

*For each problem instance, MAPP systematically identifies a set of units, which can contain all units in the instance, that are guaranteed to be solved within low-polynomial time*

Based on sliding tile puzzle. Always tries to keep a blank on an alternative path.

### 2.3.4 CBS

## 2.4 Summary

## 2.5 Further research

# References

AutoStore (2015). Autostore introduction.
**URL:** *http://autostoresystem.com/thesystem*

Bennewitz, M., Burgard, W. and Thrun, S. (2002). Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots, *Robotics and autonomous systems* **41**(2): 89–99.

Boysen, N., Briskorn, D. and Emde, S. (2017). Parts-to-picker based order processing in a rack-moving mobile robots environment, *European Journal of Operational Research* .

Cohen, L. and Koenig, S. (2016). Bounded suboptimal multi-agent path finding using highways, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, AAAI Press, pp. 3978–3979.

De Koster, R., Le-Duc, T. and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review, *European Journal of Operational Research* **182**(2): 481–501.

Guizzo, E. (2008). Three engineers, hundreds of robots, one warehouse, *IEEE spectrum* **45**(7): 26–34.

Konolige, K., Fox, D., Ortiz, C., Agno, A., Eriksen, M., Limketkai, B., Ko, J., Morisset, B., Schulz, D., Stewart, B. et al. (2006). Centibots: Very large scale distributed robotic teams, *Experimental Robotics IX*, Springer, pp. 131–140.

Ma, H. and Koenig, S. (2016). Optimal target assignment and path finding for teams of agents, *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1144–1152.

Renukamurthy, S. (2016). Improving spatially distributed multiagent pathfinding using cooperative jps.

Silver, D. (2005). Cooperative pathfinding., *AIIDE* **1**: 117–122.

Wang, K.-H. C. and Botea, A. (2011). Mapp: a scalable multi-agent path planning algorithm with tractability and completeness guarantees, *Journal of Artificial Intelligence Research* **42**: 55–90.

Wang, K.-H. C., Botea, A. et al. (2008). Fast and memory-efficient multi-agent pathfinding., *ICAPS*, pp. 380–387.

Wilt, C. M. and Botea, A. (2014). Spatially distributed multiagent path planning., *ICAPS*.

Wurman, P. R., D'Andrea, R. and Mountz, M. (2008). Coordinating hundreds of cooperative, autonomous vehicles in warehouses, *AI magazine* **29**(1): 9.

Yu, J. and LaValle, S. M. (2013). Structure and intractability of optimal multi-robot path planning on graphs., *AAAI*.