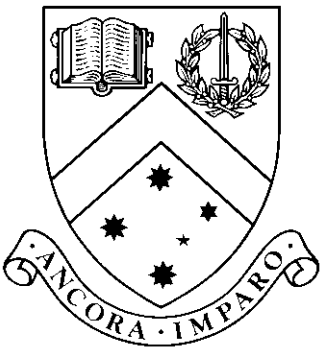
School of Computer Science
Monash University



Research Proposal — Comp Sci Honours, 2017

Exploring the effect of warehouse layout on multi-agent pathfinding in part-to-picker systems

Phillip Wong 25150510

Supervisor: Daniel Harabor

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.

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 1). 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)).

Figure 1: A worker picking an order from a storage pod. The orange robot underneath is the drive unit. (Al Dekin (2014))

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.

3 Methodology

3.1 Warehouse layout

In previous studies, we usually see the picking station positioned on one side of the warehouse and pods are laid out in rows (Fig. 2). Warehouse layout looks at configuring the location of both storage pods and picking stations. We will determine a good warehouse layout to be one which minimizes makespan as well as downtime of picking stations. Firstly, we will investigate by looking at a user-defined set of warehouse layouts. A possibility later in the project is the use of genetic algorithms to find a good warehouse layout.

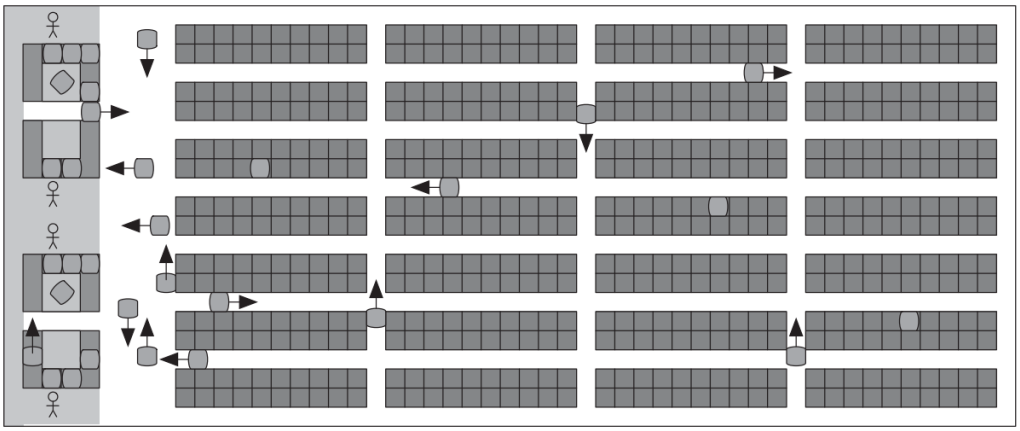


Figure 2: A Small Region of a Kiva Layout (Wurman et al. (2008)). Picking stations located on the left and storage pods laid out in rows.

Below we describe a number of adjustments and additions which are expected to play a large role in determining how we create a good warehouse layout.

3.1.1 Intermediate zone

In Section 2, we saw that Wilt and Botea (2014) identified bottlenecks in the environment and assigned controllers to simplify behaviour for any agents who wanted to travel through the bottleneck. Inspired by this, we plan to split the warehouse into two halves and introduce an intermediate zone (See Fig 3). Now, drive units located in the far zone

deliver pods to the intermediate zone instead of a pickup station. Drive units located in the delivery zone will be additionally responsible for retrieving pods in the intermediate zone and bringing them to pickup stations. We suspect this to provide benefits in traditional warehouse layouts where we have picking stations on one side and storage pods laid in rows.

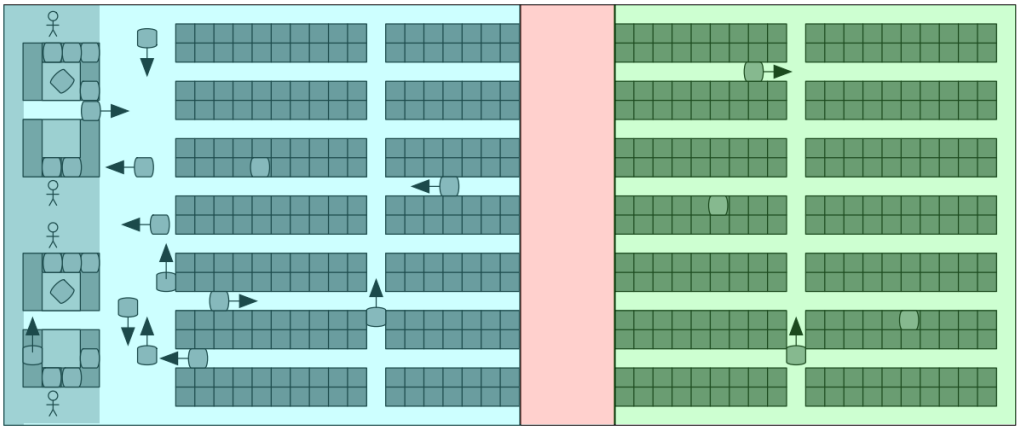


Figure 3: Intermediate zone in red, delivery zone in blue and far zone in green

3.1.2 Order processing

This process is described in detail in Section 2. To reiterate, the main points of order processing look at distribution of products around the warehouse and smart sequencing of incoming orders. This is important to consider as we expect that an optimized order processing sequence will decrease the effects of having a poor warehouse layout. Here we may take inspiration from Robin-hood hashing as well as the previous work from Boysen et al. (2017).

3.1.3 Capability for movement underneath storage pods

One of the specifications for creating a drive unit is the ability to maneuver underneath a storage pods to pick it up. This means that by slightly adjusting the dimensions of storage pods it is possible to allow drive units to maneuver underneath pods. Until a unit starts carrying a storage pod, it will not see storage pods as any obstacles in the environment. Essentially this simplifies the MAPF problem as we have decreased the number of obstacles and will increase the number of possible paths during retrieval. We expect this to reduce both the makespan of the system and downtime of picking stations. Furthermore, with this in place we suspect the warehouse layout to be less affected by the positioning of storage pods as they are not obstacles for most of the time.

3.2 MAPF and the use of a path oracle

Decentralised approaches are usually used in MAPF algorithms due to their lower CPU and memory requirements (Wang et al. (2009)) compared to a centralised approach. Decentralised MAPF involves agents independently searching for a path to their goal. Here, we aim to introduce a path oracle which will pre-compute paths, removing the cost of searching for a path at runtime and hence significantly improving performance. We expect to base this off the work by Strasser et al. (2015), utilizing a Compressed Path Database.

Path collisions in MAPF is a type of spatio-temporal conflict. We will be exploring two possibilities to resolve them. When a collision occurs, we choose one agent based on

a user-defined utility function and this agent will continue along their path. We have two methods of resolving these collisions, firstly we can make the other agent wait until the chosen agent moves out of the way. Alternatively we can use a spatio-temporal reservation table which looks forward to find a path without conflicts. This method will be based off the work by Wilt and Botea (2014).

3.3 Timetable

There are a large number of individual tasks involved in this project which have been spread across the two semesters. I have identified two major tasks which are: implementing an optimized order processing method and implementing path oracle. These have been allocated multiple weeks in semester 2 and the path oracle has been prioritized over optimized order processing as we will have implemented robin-hood hashing earlier in week 11 of semester 1. Finally, some time has been allocated to presentations or deliverables related to this project and two weeks have been allocated to focus on university examinations.

Semester 1

Week(s)	Plan
7	Create warehouse simulation with simple A* pathfinding.
8-9	Add multiple agents to the simulation using with Cooperative A* and moving between pickup stations. Agents will wait during collisions (3.2).
10	Implement an order sequencer assigning generating an inflow of products to be retrieved. Implement agents using a reservation table to resolve collisions (3.2).
11	Implement robin-hood hashing (3.1.2)
12-14	Focus on interim presentation, literature review and examinations
Holidays	Look at adding an intermediate dropping zone (3.1.1). Implement ability to move under pods (3.1.3). Begin implementing a path oracle (3.2).

Semester 2

Week(s)	Plan
1-3	Continue implementation of a path oracle (3.2)
4-7	Implement an optimized order processing method (3.1.2) based on Boysen et al. (2017)
8-11	Run simulation looking at combination of modifications and analyze results. Use any extra time to explore the use of genetic algorithms to generate warehouse layouts (3.1).
12	Finish first draft of Final Thesis
13-15	Focus on final presentation, final thesis and examinations

4 Expected Outcomes

Overall, we aim to contribute one major insight and one deliverable. In Section 3, we described methods to add: an intermediate zone, the capability for drive units to path under storage pods and an optimized order process. What we aim to answer is how these modifications may affect our decision when it comes to designing a good warehouse layout. To showcase our results, we hope to produce two main graphs comparing the makespan of the system and the idle time of picking stations across our user-defined warehouse layouts. Accompanying this would be any interesting finding we see from mixing and matching the adjustments and additions described in Section 3.

For our deliverable, we hope to produce an improved MAPF solution utilizing a precomputed path oracle and showcase its usage in a computer simulated Kiva system. We will compare it to other MAPF techniques, looking at the makespan of the system and idle time of picking stations.

References

Al Dekin, A. V. (2014). Kiva systems warehouse automation at quiet logistics.
URL: *https://youtu.be/3UxZDJ1HiPE?t=2m2s*

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.

Strasser, B., Botea, A. and Harabor, D. (2015). Compressing optimal paths with run length encoding, *Journal of Artificial Intelligence Research* **54**: 593–629.

Wang, K.-H. C. et al. (2009). Bridging the gap between centralised and decentralised multi-agent pathfinding, *Proceedings of the 14th Annual AAAI/SIGART Doctoral Consortium (AAAI-DC 2009)* pp. 23–24.

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*.