School of Computer Science
Monash University

Research Proposal — Comp Sci Honours, 2017

Exploring improvements to Part-to-picker systems in Warehouses

Phillip Wong 25150510

Supervisor: Daniel Harabor

Contents

Abstract

The order picking process is the number one expense in warehouse systems. Here we look at part-to-picker a type of order picking where the products are autonomously retrieved and given to the pickers. Previous research have focused on improvements in the multi-agent path finding but they often overlook simple adjustments or additions which may help improve the overall complexity of the problem. This project will test the effect that these [**LIST THINGS**] have on improving order-picking. We will create a simulation and focus on the warehouse layout first. The results of this project will help identify how small adjustments may affect the efficiency of the order picking process.

1 Introduction

Order picking is a process in warehouse systems whereby a product is retrieved according an incoming customer order. This process has been identified by De Koster et al. (2007) as the most costly 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 which automates the movement of products from where they are stored to a picking stations, where workers will manually pick the orders. (Wurman et al. (2008)). Part-to-picker systems often employ the use of automated vehicles when retrieve the orders from where they are stored. AutoStore (2015) is a recent 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 a human picker. 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 some downsides of this system to be similar to conveyor belts with high operational and maintenance costs as well as high retrieval cost.

In this project we look at Kiva Systems (now known as Amazon Robotics). In Kiva systems, products are stored in shelves known as storage pods. Robots known as drive units are responsible for picking up and carrying storage pods to picking stations (see Fig 1 and 2).

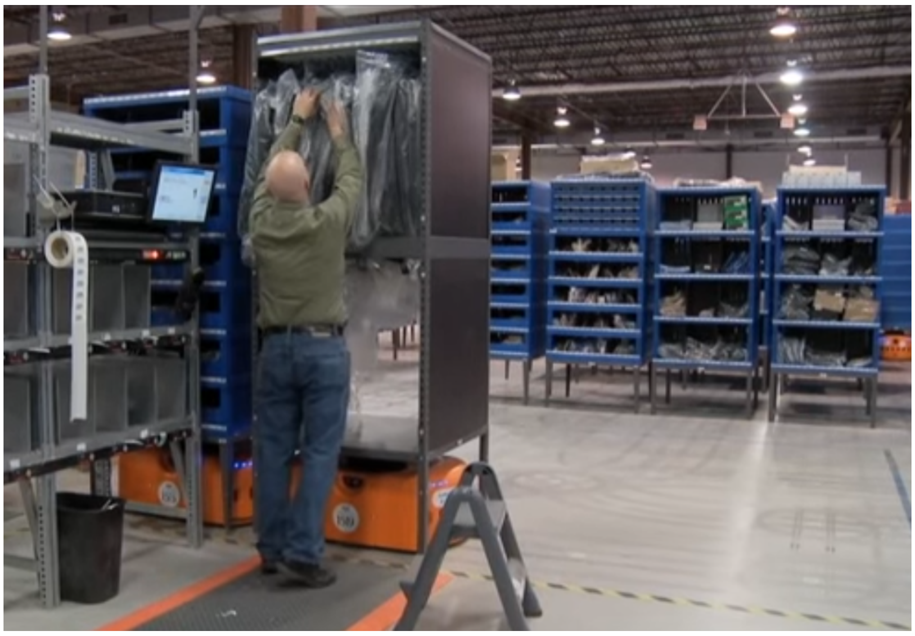

Figure 1: A worker picking an order from a storage pod. The orange robot underneath is the drive unit. (Al Dekin (2014))

The process for a drive unit is as follows:

1. Unit is told to retrieve a product
2. Unit moves to the storage pod containing the product and picks up the pod
3. Unit carries the pod to a picking station
4. Human worker picks the product from the pod and packs it
5. Unit returns the pod back to where it was picked up
6. Unit waits until it is told to retrieve another product

Kiva systems do not require a complex infrastructure to operate hence solving issues found in alternative solutions: maintenance and operational costs. When a unit malfunctions it can be easily accessed and replaced, moreover the system remains operational. The initial setup for a warehouse is cheap and fast as a warehouse needs only storage pods, a picking station and a number of drive units to operate.

1.1 Research questions

In order to explore the improvements that can be made to improving Kiva systems, we aim to answer two main questions:

1. What adjustments can be made to Kiva systems to simplify the pathfinding?
   - An improved layout of storage pods and picking stations
   - Allowing drive units are capable of maneuvering underneath storage pods
   - An intermediate zone for drive units to drop off storage pods
2. How much faster will the path search run by pre-computing paths and storing them in a path oracle?

2 Background

Kiva Systems deal with a number of problems, in this project we will be focusing on the multi-agent pathfinding (MAPF) problem. MAPF aims to find a path from for each agent so that they can reach their goal while ensuring that no path conflicts with another, in our case an agent is a drive unit and the goal is either a storage pod or picking station. When analyzing the performance of a MAPF solution we generally look at the makespan of the system. In our case of Kiva systems we will also be looking at the downtime of picking stations. Outside of order picking and warehouse applications, MAPF has usage in video games, robotics (Bennewitz et al. (2002)), search and rescue (Konolige et al. (2006)) and automated ports.

Finding an optimal solution to multi-agent pathfinding is an NP-hard problem (Yu and LaValle (2013)), has applications in systems containing few 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)). At the current time, the best we can get is a bounded suboptimal solution and this has been explored by a number of existing literature (Cohen and Koenig (2016)).

Generally methods are provided to simplify the MAPF problem, Cohen and Koenig (2016) uses user-provided highways which guide agents towards a specific direction. Wilt and Botea (2014) identifies bottlenecks in the environment and creates a controller which handles the agents who want to pass through the bottleneck. Another common simplification 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. **I'm not sure what the downside of this is...besides the fact that you generalize agents into teams of 5**.

Specific to order picking, it is possible to assist MAPF by manipulated incoming orders to suit the location of products. For example if a large order of one product came in, this could be interlaced with other orders, so agents do not need to all head to the same location. In a similar fashion, MAPF can be assisted by smart distribution of products across the warehouse. Take the same example where a large order of one product comes in, if the products were spread out around the warehouse and not situated next to one another this would not be an issue. This method is especially relevant in Kiva systems as storage pods can be easily moved around the warehouse by the drive units. Boysen et al. (2017) looks at both of these aspects 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 Path oracle

Strasser et al. (2015)
PLACEHOLDER Decentralised MAPF algorithms usually involve search. A typical problem solving process (e.g. FAR (Wang & Botea, ICAPS 2008)) involves each agent finding a path independent from all the rest (i.e. if there are k agents we solve k single-agent problems separately). When all agents have a path they each take turns moving one step at a time towards their goal. Conflicts are resolved locally choosing in favour of one agent over another in some way (e.g. assign a priority to each agent and always favour the agent with highest priority). We aim to improve efficiency by introducing a path oracle which removes entirely the need to search. The oracle is pre-computed up front and reused for every subsequent pathfinding query thereafter. Since the cost of the initial path searches tends to dominate runtime in MAPF we expect this approach will significantly improve performance. **PLACEHOLDER**

3.2 Warehouse layout

Usually the picking station is positioned on one side of the warehouse and the pods are laid out in rows (Fig. 2).

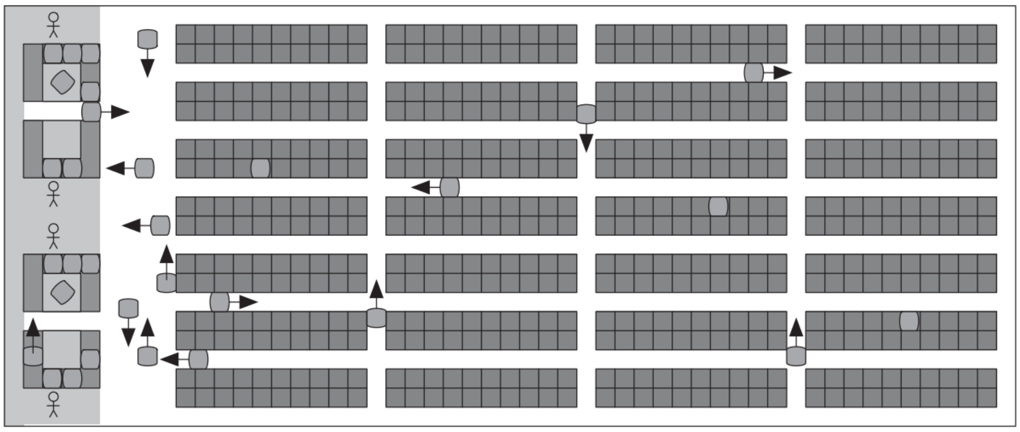

Figure 2: A Small Region of a Kiva Layout (Wurman et al. (2008)). Picking stations located on the left and storage pods laid out in rows.

Wilt and Botea (2014) looked at identifying zones by areas which are bottlenecks and assigning a controller, for that zone which manages any agents who need to travel through the bottleneck. Inspired by this and assuming pickup stations, we plan to split the warehouse into two halves and introduce an intermediate zone (See Fig 3). Delivering pods which are situated in the far zone is a two step process:

1. Units in the far zone move pods to the intermediate zone instead of a pickup station
2. Units in the delivery zone will pickup pods in the intermediate zone

These zones will have their own controller which handles any agents within the zone and tells them what behaviour should occur.

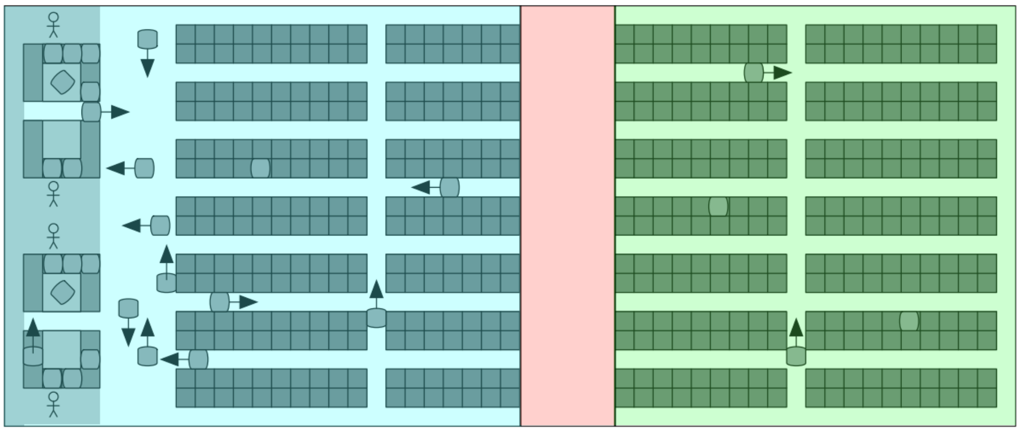
Figure 3: Intermediate zone in red, delivery zone in blue and far zone in green

3.3 Order picking

As pods can be dynamically move around the warehouse, the layout pods can be changed to suit incoming orders. Storage pods containing popular orders may be placed next to a picking station so a drive unit can readily access it and unpopular orders will be placed further away. Vice-versa the distribution of orders can be re-ordered to suit the current layout of the warehouse. Here we may take inspiration from Robin-hood hashing and apply it to sorting the inventory pods. Boysen et al. (2017) covered both of these aspects in detail and revealed that after optimizing orders, the total number of drive units can be cut by half and retain the same supply to picking stations.

3.4 Allowing for movement underneath storage pods

As drive units are capable of moving underneath pods when carrying them, this means that with small adjustments to the dimensions of storage pods it is possible to allow drive units to maneuver underneath the pods. With this the only obstacles in the environment are other drive units.

3.5 Timetable/plan

HALF DONE

Semester 1

Week(s)	Plan
7	Model warehouse and simple A* pathfinding
8	Add multiple agents with A* assigned random pods (no picking station) arsasarsa
9	Implement Cooperative A*
11	Add simple scheduler which assigns agents a location to fetch a random pod and return to the picking station
12	Focus on Interim Presentation
13	Focus on Literature Review
14	Focus on Examinations
Holidays	Implement Path Oracle with Compressed Path Databases

Semester 2

Week(s)	Plan
1	Add more complex scheduler, distributing requested inventory and allow agents dynamically sort pods according to popularity. Agents dynamically sort pods according to popularity. Decide on the focus for the rest of the project.
2-4	Implement Path Oracle with Compressed Path Databases
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	Finish Final Thesis
13	Additional tasks
14	Focus on Final Presentation
15	Focus on Final Thesis

4 Expected Outcomes

We suspect that these adjustments will reduce the number of conflicts in the MAPF problem hence allowing for a better bounded suboptimal solution to be found. Regardless of results, we expect to be able to better understand the explored properties described in Section 3 and their effect on Kiva systems.

Additionally we hope to deliver an improved MAPF solution utilizing a pre-computed path oracle and a simulation showcasing its usage in a Kiva system.

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.

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*.