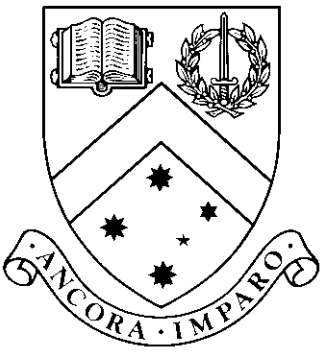
School of Computer Science (BICA)
Monash University



Literature Review, 2017

Improving cooperative pathfinding using a path oracle

Phillip Wong

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).

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 suitable 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)).

2 Introduction 2

This project looks at improving multi-agent pathfinding (MAPF) system in the context of Kiva Systems. Our approach is focused on optimal MAPF and we will be looking at two major areas:

1. Using Branch and Price to identify and resolve path-conflicts
2. Improving search speeds?

The usage of branch and price is the focus of this project and as such this literature review will cover a wide range of optimal MAPF algorithms. The major part of this section will show a comparison of these algorithms across a number of different measures / features. **Additionally we hope to find the difference between heuristic searches and reduction searches.**

As for the search techniques, we will cover only those which we intend to implement into this project which are: Jump Point Search (JPS) and Compressed Path Databases. Here we will describe how the algorithm works, any major improvements and describe the speedups.

Finally the last section of this review will overview literature on Warehouse Automation. There are many optimization aspects specific to Warehouse Automation. In this project a wide number of simplifications will be made in order to focus on MAPF. This section will recognize these as future work and briefly look at any past work that has been done.

3 Background

We will be looking at pathfinding in a orthogonal grid-based map. Formally we describe this as:

- A graph $G = (V, E)$
  - V is a set of nodes
  - E is a set of edges
  - $|E| \leq 4$

Multi-agent Pathfinding. In multi-agent pathfinding (MAPF) we look at a set of agents, K where $location(k) \in V$. MAPF aims to find a path p from start node s to goal node g for each agent in K. A path is a sequence specifying agent is on node n at timestep t. Finally no path can conflict with another path.

Formally defining this:

- P is the set of all paths
- $\forall K(location(k) \in V)$
- $\forall K(\exists s(s \in V) \wedge \exists g(g \in V))$
- $\forall (k_1 \in K) \forall (k_2 \in K)(k_1 \neq k_2)(\exists p \wedge \forall K)$
- $\forall p_1 \in P, \forall p_2 \in P(\forall tl_1 \in p_1, \forall tl_2 \in p_2) tl_1 \neq tl_2$

MAPF describes pathfinding

Multi-agent pathfinding involves finding a path for each agent to their goal such that no path conflicts another.

3.1 Single agent pathfinding

Single-agent pathfinding involves searching for the shortest path from a start node, to goal node. We will refer to single-agent pathfinding as searching. There are a number of areas where searching can be sped up.

1. Better Heuristics

2. Pruning the search space

3. Precomputating paths

In this project we will be looking at *Jump Point Search* within the area of pruning and *Compressed Path Databases* for pre-computation.

3.2 Better Heuristics

Briefly covering the state of the art in heuristics. Heuristics - **List some new heuristic?** Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

3.3 Pruning the search space

Pruning the search space is

3.4 Precomputating paths

Delling et al. (2009)

3.5 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?

4 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 (2013b)) 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.

4.1 Objective function

Yu and LaValle (2015) looked at:

1. Total Arrival time:
2. Makespan: The time between
3. Total Distance:
4. Maximum Distance:

1. Test

4.1.1

4.2 Improving A*

4.2.1 Compressed Path Databases

4.2.2 Jump Point Search

Renukamurthy (2016)

4.3 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 where the difference between the found path length and the optimal path length is always 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

Dynamic?: If the environment is changed does anything need to be recomputed i.e. did the algorithm require preprocessing
Anytime: The search can spend more computation to improve the solution quality or number of solutions. It can stop earlier at any time and return a path.
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 *An online algorithm is one that can only process its input piece-by-piece in a serial fashion, in the order that the input is fed to the algorithm, without having the entire input available from the start. An offline algorithm is given the whole problem data from the beginning and is required to output an answer which solves the problem at hand.*

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

4.4 Cooperative Multi-agent pathfinding

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

5 Suboptimal

5.0.1 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

5.1 Variants of optimal algorithms

5.1.1 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.*

6 Reduction-based solvers

Surynek et al. (2016) "Many recent optimal solvers reduce MAPF to known problems such as COP [15], SAT [26], Inductive Logic Programming [33] and

Answer Set Programming [9]. These papers mostly prove a polynomial-time reduction from MAPF to these problems. These reductions are usually designed for the makespan variant of MAPF; they are not applicable for the sum-of- costs variant. (2)"

6.1 LP

Linear programming is commonly used when dealing with NP-hard problems. In pathfinding it excels at finding an optimal path. **In this technique, we look at an optimization function and constraints. A solver will work to find the optimal solution to the problem.**

The book by Ahuja et al. (1993), introduces many algorithms and applications, most related to this research is Section 4 which overviews shortest path searches. Planes and Beasley (2009) applies integer programming to search through a metabolic pathways. Here they find a series of biochemical reactions which a living organism will transform an compound to the target compound.

MAPF

Schouwenaars et al. (2001) Multi-vehicle path planning. Mixed integer programming.

Richards and How (2002) Aircraft collision avoidance. Mixed integer linear programming.

6.2 SAT

Surynek et al. (2016) applies Linear program and looks at MAPF as a Satisfiability problem (SAT). They create a SAT solver for the bounding sum-of-costs objective function.

- Cardinality Constraints [3, 19]
- Time expansion graph [27, 28, 29] - Makespan variant
- Uses Multi-value Decision Diagram from Sharon et al. (2011)

6.3 Multi-flow based integer linear programming

Yu and LaValle (2013a) Multi-robot path planning. Multiflow based integer linear programming (ILP).

Branch and price is a method for solving integer programming problems

6.4 Answer Set Programming?

7 Procedure based

"Complete. Very fast. Far from optimal. Solve very large problems"

7.1 Slidable

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.

7.2 Push and Swap and Push and Rotate

7.3 Tree-based agent swapping strategy

8 Search trees

8.0.1 Increasing cost search tree (ICTS)

8.0.2 Conflict-Based Search (CBS)

Conflict-Based Search (CBS) is a optimal search algorithm.

When a collision occurs a constraint Constraints describe the agent, time and location.

The algorithm uses a constraint tree to describe these constraints and searches through this tree.

Tree-based agent swapping strategy (TASS)

tree-based agent swapping strategy

9 Optimality

9.1 Combined target assignment and pathfinding (TAPF)

Combined target assignment and pathfinding (TAPF) couples the target-assignment and the pathfinding problems and defines one common objective for both of them. Agents are partitioned into teams. Each team is given the same number of unique targets as there are agents in the team. The task of TAPF is to assign the targets to the agents and plan collision-free paths for the agents from their current vertices to their targets in a way such that each agent moves to exactly one target given to its team, all targets are visited and the makespan is minimized

9.2 Conflict-based Min-cost flow (CBM)

Conflict-based Min-cost flow

The anonymous variant of MAPF (also called goal- invariant MAPF). Agents can get assigned any target and are thus exchangeable. It can be solved optimally in polynomial time with flow-based MAPF methods

9.3 Package-Exchange Robot-Routing (PERR)

Package-Exchange Robot-Routing (PERR)

9.4 Highways

Cohen and Koenig (2016)

9.5 Operator Decomposition (OD)

Standley proposes the mechanism of operator decomposition (OD) [30]. Instead of considering that the set of agents decides its joint action (operator), the agents decide their elementary action in turn, one after each other. With some care to special cases, A associated with OD and a perfect heuristic is admissible and complete [30]*

9.6 Independance Detection (ID)

Each agent plans its optimal path to the goal with A. Then conflicts are detected, and resolved if possible. When a conflict between two agents is not solvable, the two agents are gathered into a group, and an optimal solution is found for this group. Conflict between groups are detected and so forth. When few conflicts occur this method avoids the exponential size of the set of actions, but in the worst case, this method is inferior to any centralized method.*

10 Other factors

Hönig et al. (2016) factored in kinematic constraints.

11 Warehouse

11.1 Overview papers

Gu et al. (2007) (Research on warehouse operation: A comprehensive review)
Gu et al. (2010) (Research on warehouse design and performance evaluation: A comprehensive review)

11.2 Other factors

Correll et al. (2016)

12 Information outside of the literature

System and method for coordinating movement of mobile drive units - https://www.google.com/patents/US7
https://www-google-com/patents/US9511934
ICAPS 2014 Invited Talk: Peter Wurman - https://www.youtube.com/watch?v=YSrnV0wZywU

12.1 Summary

12.2 Further research

References

Ahuja, R. K., Magnanti, T. L. and Orlin, J. B. (1993). Network flows: theory, algorithms, and applications.

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.

Correll, N., Bekris, K. E., Berenson, D., Brock, O., Causo, A., Hauser, K., Okada, K., Rodriguez, A., Romano, J. M. and Wurman, P. R. (2016). Lessons from the amazon picking challenge, *arXiv preprint arXiv:1601.05484* .

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.

Delling, D., Sanders, P., Schultes, D. and Wagner, D. (2009). Engineering route planning algorithms, *Algorithmics of large and complex networks*, Springer, pp. 117–139.

Gu, J., Goetschalckx, M. and McGinnis, L. F. (2007). Research on warehouse operation: A comprehensive review, *European journal of operational research* **177**(1): 1–21.

Gu, J., Goetschalckx, M. and McGinnis, L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review, *European Journal of Operational Research* **203**(3): 539–549.

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

Hönig, W., Kumar, T. S., Cohen, L., Ma, H., Xu, H., Ayanian, N. and Koenig, S. (2016). Multi-agent path finding with kinematic constraints., *ICAPS*, pp. 477–485.

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.

Planes, F. J. and Beasley, J. E. (2009). Path finding approaches and metabolic pathways, *Discrete Applied Mathematics* **157**(10): 2244–2256.

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

Richards, A. and How, J. P. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming, *American Control Conference, 2002. Proceedings of the 2002*, Vol. 3, IEEE, pp. 1936–1941.

Schouwenaars, T., De Moor, B., Feron, E. and How, J. (2001). Mixed integer programming for multi-vehicle path planning, *Control Conference (ECC), 2001 European*, IEEE, pp. 2603–2608.

Sharon, G., Stern, R., Goldenberg, M. and Felner, A. (2011). The increasing cost tree search for optimal multi-agent pathfinding, *Twenty-Second International Joint Conference on Artificial Intelligence*.

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

Surynek, P., Felner, A., Stern, R. and Boyarski, E. (2016). Efficient sat approach to multi-agent path finding under the sum of costs objective, *Proceedings of 22nd European Conference on Artificial Intelligence (ECAI 2016)*, pp. 810–818.

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. (2013a). Planning optimal paths for multiple robots on graphs, *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, pp. 3612–3617.

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

Yu, J. and LaValle, S. M. (2015). Optimal multi-robot path planning on graphs: Structure and computational complexity, *arXiv preprint arXiv:1507.03289* .