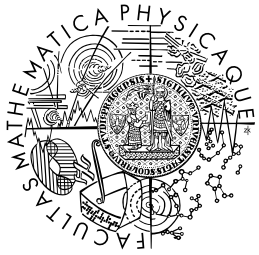

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Jiří Krejčí

Multi-agent picker routing problem

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis: prof. RNDr. Roman Barták, Ph.D.
Study programme: Computer Science
Study branch: General Computer Science

Prague 2020

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Multi-agent picker routing problem

Author: Jiří Krejčí

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: prof. RNDr. Roman Barták, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Abstract.

Keywords: multi-agent path finding, picker routing, warehouse, TSP

Contents

Introduction **2**

1 Multi agent picker routing **3**
1.1 Real-world warehouse specification . . . 3
1.2 Problem input . . . 4
1.3 Problem formulation . . . 4
1.4 Related problems . . . 5

2 Related work **7**
2.1 Picker routing . . . 7
2.1.1 Exact algorithms . . . 7
2.1.2 Heuristics . . . 8
2.1.3 Meta-heuristics . . . 9
2.2 Multi-agent path finding . . . 9
2.2.1 Search-based solvers . . . 9
2.2.2 Reduction-based solvers . . . 9
2.2.3 Rule-based algorithms . . . 9

3 Analysis **10**
3.1 Solution approaches . . . 10
3.1.1 Picker routing problem view . . . 10
3.1.2 Multi-agent path finding view . . . 11
3.2 Conflict based search . . . 12

4 Testing **13**
4.1 Testing methodology . . . 13
4.2 Results . . . 13
4.3 Discussion . . . 13

Conclusion **14**

Bibliography **15**

List of Figures **17**

List of Tables **18**

List of Abbreviations **19**

A Attachments **20**
A.1 First Attachment . . . 20

Introduction

Order picking, which is the process of retrieving items from storage to meet customer demand, and warehouse replenishment are critical functions to each supply chain. In prior studies, these have been identified as the most labor-intensive and costly activities for almost every warehouse. The cost of order picking is estimated to be as much as 55% of the total warehouse operating expense [1]. Even though the use of automated equipment in the distribution centers is on the rise and has potential to lower the cost of order picking over time, manual order picking is still prevalent. In this thesis, we focus on the manual order picking problem, but the results can be applied to the automated case as well.

In its essence, order picking problem is a variant of the well-known Traveling Salesman Problem. In the past, researchers have developed many approaches to solve this special case of TSP, that can be used in practice. Exact polynomial time algorithm was developed by Ratliff and Rosenthal in 1983 [2], which solves the problem for the most common single-block warehouse layout. Because the algorithm lacks flexibility to be used in many real-world scenarios, it has been extended multiple times, for example to include warehouses with more cross-aisles [3] or, more recently, to take into account precedence constraints [4].

Even though the exact algorithm is developed, it is the heuristic approach that is still arguably the most commonly used in practice. These routing policies - despite the resulting routes often being suboptimal - are very quickly computed, easy to implement and can be followed effortlessly by order pickers. The heuristic algorithms can often be described by a simple, easy to remember rule. They were originally developed to be used in a single-block warehouse with narrow aisles and a single depot [5], but similarly to the algorithm by Ratliff and Rosenthal [2], the heuristics have been studied further and extended to multiple-block layouts over the years [6][7].

In a typical warehouse, there are multiple order pickers working at the same time. Up until now, none of the methods mentioned considers interactions between pickers - namely picker blocking and congestion. This situation arises, when multiple pickers are given similar, intersecting routes. Many studies have been conducted on the effects of picker blocking [8][9][10]. Picker blocking is more pronounced in narrow aisle warehouses and can be a cause of significant delays and reduction of picking process efficiency.

Given the measurable efficiency cuts due to congestion, surprisingly little research has been done on picker routing with congestion consideration. It is goal of this thesis to develop an algorithm that will compute routes for order pickers while avoiding mutual blocking at the same time. TBD - how is the problem going to be solved.

TBD - In the first chapter, we begin with description of the real-world warehouse layout and processes to understand the motivation behind this thesis. Next, we will define important terms and finally, the definition of the problem we are solving will follow.

1. Multi agent picker routing

In this chapter, first we describe the real-world warehouses that motivate this thesis and after that, the formulation of the problem will follow. Lastly we will introduce related problems in this chapter.

1.1 Real-world warehouse specification

Condider a parallel-aisle warehouse with narrow vertical aisles and several horizontal cross-aisles as shown in the Figure 1.1 for illustration. Furthermore, the warehouse is high bay, meaning it has storage racks with pallets stored at multiple height levels. In addition to the main storage area, the warehouse may also contain special zones, e.g. freezers and regions separated by walls that are connected to the main grid. There are two areas that are connected to the Input/Output ports of the warehouse - the inbound and outbound staging areas. Inbound area serves as a buffer for new products arriving to the warehouse that are waiting to be restocked. Outbound area has similar purpose, but in reverse - its where completed customer orders are taken after they have been picked up in the storage area and are waiting to be loaded into trucks for delivery. Products are stored in vertical aisles on both sides. One product type can be stored at multiple locations in the warehouse.

People or autonomous agents navigate through the warehouse. In this thesis, we will be dealing mostly with manual warehouses. There are two kinds of tasks we consider - order picking and warehouse replenishment. To recapitulate, order picking is the process of retrieving items from storage, usually given to the agent on the so called pick list. The reverse process is replenishment. From our perspective, the only difference between the two processes is the staging area the agent begins or ends his tour at. Therefore, to simplify the terminology, we will refer to all people performing these tasks as order pickers, or agents more generally.

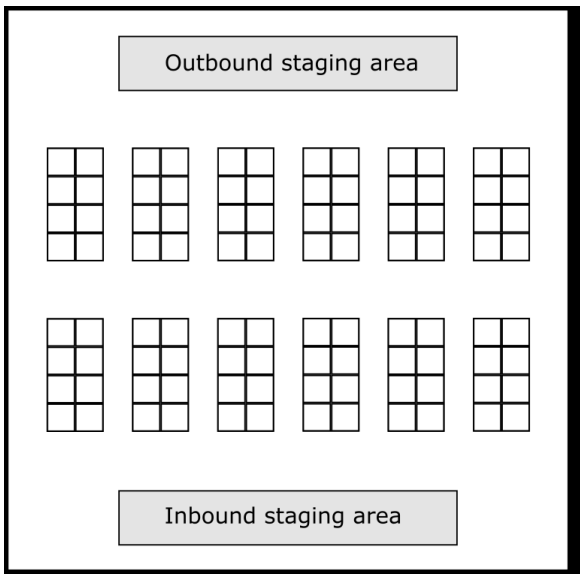

Figure 1.1: Warehouse layout

Figure 1.2: High bay storage

During the workday, order pickers will be assigned cutomer orders. The orders, their assignment to specific agents and the precise sequence is known in advance. Agents walk or ride through the warehouse with a picking device and pick items from a pick list, or replenish items respectively. We assume that the picking device has enough capacity to carry all items specified in any customer order. As already mentioned, the aisles are narrow, meaning that the order picker can reach product racks on both sides of an aisle without any time penalty, but one order picker cannot pass another in aisle.

The goal is to plan routes for order pickers so that the time spent picking is minimized, avoiding any collisions. We will split the problem into two parts. First part, since the problem is real-world motivated, is the transformation of real-world instances into abstract form. This abstract view will serve as an input for the second part of the problem - the algorithm that will calculate the optimal routes.

1.2 Problem input

We will neglect the real-world specifics for now and focus on the abstract problem only. The input to the abstract multi-agent picker routing problem is:

1. A directed graph $G = (V, E)$, which is assumed to represent a 4-connected grid with obstacles. The vertices of the graph are all possible locations for the agents, and the edges are possible transitions. All edges have unit weight.

2. Set of m tasks $O = \{O_1, ..., O_m\}$. Each task is specified by a start vertex, $start_{O_i} \in V$, a goal vertex, $goal_{O_i} \in V$ and a collection of sets, where sets contain vertices from G and are mutually disjoint.

3. k agents $a_1, ..., a_k$.

4. For each agent $a_k \in A$, an ordered list R_k is given, such that every task $O_i \in O$ is contained in exactly one list R_j.

Time is discretized into time steps. We assume without loss of generality, that the start vertex of any task is equal to the goal vertex of previous task in the corresponding ordered list, if such task exists.

1.3 Problem formulation

Let input to the problem be the same as just defined. Each agent occupies a single vertex at any given time and similarly, each vertex and edge can be used by at most one agent at any given time step. Agents can perform one of the three actions at a time - *stay* at the vertex, *move* to an adjacent vertex and finally, perform a *visit* action.

All agents are given a list of tasks. Each task contains a collection of sets. The sets consist of locations to be visited - from each set, exactly one location has to be *visited*. Such visit using the special action takes some positive time t specific for each item, location pair - an agent must stay at least t time units at

the vertex. For the other vertices and regular movement actions, such condition does not apply and an agent can stay at the vertex for arbitrarily many time steps.

Agents perform the tasks one at a time and after an agent finishes a task, he immediately begins to perform the next one. The order in which the tasks are performed is given.

The goal is to find a tour for each task and agent, such that exactly one vertex from each set is visited for a given task and the tours are minimizing the sum of costs objective function.

1.4 Related problems

Definition 1. *(Steiner TSP). The Steiner traveling salesman problem is a NP-hard problem derived from the TSP. Given a list of cities, some of which are required to be visited, and the lengths of roads between them, the goal is to find the shortest possible walk that visits each required city and then returns to the origin city. Vertices may be visited more than once and edges may be traversed more than once as well.*

In its basic, single agent form, the picker routing problem is classified as Steiner TSP. The issue is, besides dealing with single agent situation only, that the Steiner TSP requires visiting of vertices from one set only. To satisfy the real-world requirement, that one item can be stored at multiple locations in the warehouse, we need another TSP variant.

Definition 2. *(Generalized TSP), also known as the set TSP, is a generalization of the TSP. The goal is to find a shortest tour in a graph, which visits all specified subsets of vertices of a graph. The subsets of vertices must be disjoint.*

Any Generalized TSP solver can actually be used to find a solution to an instance of single agent, single order subproblem. We would only need to pre-process the input graph to include only required vertices for the given tour. As a generalization of the TSP, there is a direct transformation for an instance of the set TSP to an instance of the standard asymmetric TSP. But gTSP still doesn't fit the requirements and we need to introduce yet another related problem to help us deal with the multiple agents situation.

(Pozn - nechat definice nize primo z clanku, nebo prepsat vlastnimi slovy?)

Definition 3. *(Multi agent path finding)[11] The input to a classical MAPF problem with k agents is a tuple $\langle G, s, t\rangle$, where $G = (V, E)$ is an undirected graph, $s : [1, ..., k] \to V$ maps an agent to a source vertex, and $t : [1, ..., k] \to V$ maps an agent to a target vertex. Time is assumed to be discretized, and in every time step each agent is situated in one of the graph vertices and can perform a single* action*. An action is a function $a : V \to V$, such that $a(v) = v'$ means that if an agent is at vertex v and performs a then it will be in vertex v' in the next time step. Each agent has two types of actions:* wait *and* move*. A* wait *action means that the agent stays in its current vertex another time step. A* move *action means that the agent moves from its current vertex v to an adjacent vertex v' in the graph.*

For a sequence of actions $\pi = (a_1, ..., a_n)$ and an agent i, we denote by $\pi_i[x]$ the location of the agent after executing the first x actions in π, starting from the agent's source $s(i)$. Formally, $\pi_i[x] = a_x(a_{x-1}(...a_1(s(i))))$. A sequence of actions π is a single-agent plan for agent i iff executing this sequence of actions in $s(i)$ results in being at $t(i)$. A solution is a set of k single-agent plans, one for each agent.

The concept of collision will be introduced separately now. For the purposes of this thesis, we are interested in two types of *conflicts* - the *vertex conflicts* and the *edge conflicts*. Finally, o objective function for comparing the solutions has been defined yet. We will work with two objective functions throughout the thesis - the makespan, and the sum of cost.

Definition 4. *(Vertex conflict)[11]. A* vertex conflict *between π_i and π_j occurs iff according to these plans the agents are planned to occupy the same vertex at the same time step.*

Definition 5. *(Edge conflict)[11]. A* edge conflict *between π_i and π_j occurs iff one agent is planned to occupy a vertex that was occupied by another agent in the previous time step.*

Definition 6. *(MAPF objective funtctions)[11]*

1. ***Makespan.*** *The number of time steps required for all agents to reach their target. For MAPF solution $\pi = \{\pi_1, ..., \pi_k\}$, the makespan of π is defined as $max_{1 \leq i \leq k} |\pi_i|$.*
2. ***Sum of costs.*** *The sum of time steps required by each agent to reach its target. The sum of costs of π is defined as $\sum_{1 \leq i \leq k} |\pi_i|$.*

The multi agent path finding problem has many variants and extensions, but the definitions above will suffice in providing the terminology and concepts needed for our purposes. MAPF might be the closely related problem of the three mentioned. Suppose that for some agent a_i, $\pi = \{\pi_1, ..., \pi_k\}$ is a permutation of vertices that are required to be visited by the agent a_i in a given task. To convert the gTSP solution into a single-agent plan, we just need to solve $k - 1$ single agent path finding problems and concatenate the paths into a single path, or, to be more precise, a walk. Likewise, if we look at the problem from a different perspective, the difference lies in the fact, that instead of simple and non-conflicting routes, we search for non-conflicting tours for the agents instead.

2. Related work

In this chapter, we will explore in detail the research that has been carried out on the topic. We will go through the related problems and sum up the most promising results.

2.1 Picker routing

There are number of issues we have to look out for when reviewing literature on the picker routing problem. First, algorithms that solve the problem are narrowly specialized. Usually the algorithm is developed for a certain warehouse layout and cannot be used with any other. Most of the time, even less significant aspects like depot position matter. But the biggest issue is connected with the objective of this thesis - most of the algorithms are designed for single storage warehouses that have one item type stored at single location only. In this thesis, we need to find routes in a warehouses with scattered storage, which on the other hand can contain multiple locations with the same item stored there. Secondly, the performance of the algorithms varies according to specific instances. For example, some heuristic might give short routes when customer orders are larger, but it could perform poorly otherwise. Thirdly, connected with the performance from the computational side, we could encounter larger warehouse instances, therefore some optimal exponential time algorithms could be too slow to be usable.

2.1.1 Exact algorithms

Even though picker routing problem is a special case of the Traveling Salesman Problem and hence is generally NP hard, optimal algorithm linear in the number of aisles was developed by Ratliff and Rosenthal[2] for the single-block warehouse layout. The authors assumed a low-level storage rack, single depot in the front cross aisle and narrow picking aisles. Ratliff and Rosenthal used dynamic programming approach, gradually building *partial tour subgraphs*, which could in short be described as subgraphs, that can be extended into a valid tour. The authors noticed, that the PTSs can be divided into fixed number of equivalence classes and therefore the algorithm needs to consider only fixed number of minimum length PTSs at any given step.

Since warehouses often have more complex layouts, the extension of the algorithm to three cross aisles followed[3]. The latest contribution is by Pansart et al.[12], who applied an algorithm for the rectilinear TSP developed by Cambazard and Catusse[13]. However, the complexity of the algorithm increases rapidly with the number of cross aisles. More specifically, the time complexity of the algorithm is $\mathcal{O}(nh7^h)$, where n is the number of vertices and h is the number of horizontal lines. The situation changes, when we introduce scattered storage into the problem. No such algorithm can have polynomial time complexity for any common warehouse layout, because it was proved that the problem is NP hard for any warehouse with rectangular storage[14].

Another approach to solving the picker routed problem optimally is modifying some algorithm that was originally developed for solving the TSP. Drawback of

this approach is the often unpredictable run-time behavior. For the purpose of finding a tour in narrow-aisle warehouse, Roodbergen and de Koster used a branch-and-bound method[7]. Theys et al. applied the exact Concorde TSP solver, though for comparison with other methods only.[15].

2.1.2 Heuristics

Another possible and arguably the most frequently used practice for calculating picking routes is by the means of heuristic algorithms. Unlike the exact algorithms, heuristics are not guaranteed to find the shortest tour possible, but this attribute is offsetted by the ease of implementation and faster calculation times. Moreover, heuristic rules are usually easy to remember and follow for order pickers, who might struggle with following more complex routes.

For narrow-aisle warehouses, Hall proposed and evaluated performance of three simple heuristic routing strategies[5], namely the *Traversal* strategy, *Midpoint* and *Largest Gap* strategy. Another common routing strategy is *S-shape* heuristic. We will briefly describe the traversal strategy for illustration. The traversal strategy tells the picker to cross through the entire lenght of any aisle containing at least one pick location. The other strategies can be described in analogous manner. Later, the *return* and the *composite* heuristics were further developed[16]. What all of these heuristics have in common is that they were originally developed for single block layout only. The multi-block layout variants will follow in the next paragraph.

Extension to multi-block layout for S-shape and largest gap heuristic was introduced by Roodbergen and de Koster, along with the new *combined* heuristic[7]. Chen et al. developed two modifications of S-shape heuristics[17], that attempt to avoid congestion. In the first modification, this is achieved by the condition that if an aisle is already occupied by one picker, the other must wait at an entrance of the aisle. The other modification considers spatial relationships between picked item and the next item to determine the travel time and the waiting time. If congestion occurs, it can dynamically reroute the picker. These heuristics were introduced as a benchamark for the Ant Colony Optimization algorithm, that was developed in the same paper as well.

Recently, Weidinger developed new heuristic approach to address routing in scattered storage warehouses[14]. His approach proves to be very efficient, the autor states just 0.5% mean optimality gap in the paper. Because of the complexity of the problem, author partitioned the problem into two parts - storage positions selection and picker routing. The positions selection is done by three simple priority rules, that compute route distances implicitly, and are capable of finding the shortest tour for the given problem instace.

The choice of the best heuristic depends on many factors, among the most important are pick density, number of cross aisles and storage policy. The difference between heuristic and optimal route can be as low as 1%, but it can get much higher as well[7], especially if the wrong heuristic for the situation is chosen. There are more rule-based heuristics, but as an overview, the list provided is sufficient.

Another kind of heuristics worth mentioning are so called *improvement heuristics*. They try to improve an initial solution generated by some rule-based heuris-

tic by means of local search. Improvement heuristics are oftentimes more general than order picker routing algorithms, because they are usually used when solving TSP directly. Improvement heuristics are for example the *2-opt* and the *3-opt* local searchech or their generalization - the *LKH* TSP heuristic. The LKH heuristic is applied in the paper written by Theys et al[15].

2.1.3 Meta-heuristics

To complete the list of all approaches to the picker routing problem found in literature, we must not forget the meta-heuristics, which gained popularity mostly in the recent years. In the single block case, meta-heuristic approaches are used mostly to solve complex combined optimization problems, for example the joint order batching and picker routing problem. The exception is a paper, in which the authors used hybrid genetic algorithm to solve picker routing with product returns and even considered interactions between the order pickers[18].

For the multi-block warehouses, use of meta heuristic algorithms is more frequent for solving picker routing independently, but the combined approaches are still prevalent. One of the earliest uses of meta heuristic was by Chen et al.[17], whose work was already mentioned in the heuristics subsection. They applied an Ant Colony Optimization approach to multi-block narrow-aisle warehouse with two order pickers while also accounting for congestion. The work was extended in 2016 to include online routing method under nondeterministic picking time[19]. Authors concluded, that the new method can reduce the order service time by coping with the congestion.

Another algorithm, that uses ACO optimization, is FW-ACO[20]. The algorithms is a combination of the ACO meta-heuristic and Floyd-Warshall algorithm. Authors claim, that the algorithm is best suited for complex warehouse configurations, where the shortest path generated by the FW-ACO is usually significantly better than the path returned by regular heuristic algorithms.

2.2 Multi-agent path finding

TBD

2.2.1 Search-based solvers

Search-based solvers A* based (WHCA*, EPEA*, M*), ICST, CBS

2.2.2 Reduction-based solvers

SAT, MILP, ASP

2.2.3 Rule-based algorithms

Push and Swap, Bibox

3. Analysis

In this section, we will discuss approaches, that could possibly be used to solve the problem.
Pozn. V této fázi jsem se především snažil o to, abych shrnoul své myšlenky do souvislého textu. Takto bude možné snadněji najít případné nesrovnalosti.

3.1 Solution approaches

We have identified the closely related problems and reviewed the literature in order to compile a list of possible approaches. Now, we will go through the individual approaches and look into possible uses and extensions. Since the problem is actually twofold, we have two ways of looking at the problem - as a picker routing problem with extension to multiple agents, or as a multi-agent path finding problem modified to search for tours instead. The goal of this chapter is to find the most promising approach. One of the issues we face, as was already stated, is that both problems are NP hard. As a consequence, we will have to be careful when considering optimal algorithms, as the time needed to find a solution could easily exceed any limits we might impose. In other words, the use of heuristics might be required in order to get reasonable computation times. On the other hand, we benefit from the extensive research in both areas that has already been done.

First idea might be to express the problem as a whole as an mixed integer programming problem. Unfortunately, even though the solvers are powerful, we can assume, that real-world instances would be too large to be solved, or even too large to be loaded into computer memory.

3.1.1 Picker routing problem view

The picker routing problem is probably as close to our problem as we can get. Nevertheless, since the algorithms for picker routing problem are so specialized, modification to match our problem specification could prove to be challenging.

Optimal

Starting with the optimal polynomial time algorithm by Ratliff and Rosenthal[2], we can immediately conclude from the work of Weidinger[14], that even if the modification of the algorithm to scattered storage was possible, it would not preserve the time complexity. Another issues we would face are the multi-agent scenario and the non-standard warehouse features. The algorithm is too specific. Therefore, use of this algorithm in our work doesn't seem plausible. Similarly, incorporating our requirements into TSP instance would probably be way too complex. It is possible to reduce gTSP instance into asymmetric TSP instance without much effort. The complex part would be expressing the multi-agent constraints, which would probably cause, that the off-the-shelf TSP solvers would struggle with instances of real-world size.

Heuristic

With rule-based heuristic approaches such modifications are definitely possible, since Chen et al. already developed modification to mitigate picker blocking based on the S-shape heuristic[17] and Weidinger developed heuristic for scattered storage[14]. The only thing that would be left to consider in the hypothetical combined heuristic is, how to deal with the non-standard areas we might encounter in our test instances. This could probably be solved by adding a set of new rules. In the modified S-shape heuristics, pickers wait at entrance of the subaisle, if the subaisle is already occupied. This alleviates congestion, but at the expense of time spent waiting. Better alternative could be having one-way subaisles. That way, pickers would still have to wait, but for shorter periods, which might offset the extra time spent travelling to the right subaisle entrances. In conclusion, rule-based heuristic routing policy seems possible, but the resulting routes would most likely be far from optimal. We might consider using such approach as a benchmark. Lastly, the heuristic TSP approaches have the same issue as exact. If we attempted to reduce the problem into TSP, the reduced instance would be too large and too complex.

The last picker routing problem approach to consider is the use of meta-heuristic algorithm. Because the meta-heuristic approach is versatile, it might work reasonably well. The issue here is, that developing and fine-tuning meta-heuristic algorithm requires considerable amount of experience to do right. Another reason not to try this method is simply that there might be better and more reliable approaches.

3.1.2 Multi-agent path finding view

The problem we are dealing with is multi-agent in its nature, therefore, multi-agent point of view seems logical. Surprisingly, the number of options to consider is smaller than in the previous case. On the other hand, the advantage is that some of the recent state-of-the-art algorithms are customizable, so it might be possible to employ problem-specific knowledge into the solution.

Optimal

We have two options when considering optimal MAPF algorithms, the search-based algorithms and the reduction-based algorithms. Unproven heuristic shared among researchers says that reduction-based algorithms usually perform better, when the graphs are smaller and densly populated, while the search-based approaches are performing better, when the instance is large and the number of expected conflicts is low. Therefore, we will focus mainly on the search-based approaches.

Because of the added complexity of searching for gTSP tours instead of paths, it's probably safe to say, that A* based approaches wouldn't perform well. It is not even clear what the heuristic function of A* should look like, since there is not a single location the agent is trying to reach.

The ICST seems to be performing reasonably well on the MAPF test instances. It's two level approach fits our scenario better, since we would have to modify the low-level only. The issue we would face is, how to efficiently enumerate all

solutions of cost C_i. Another issue is, that picking can take considerable amount of time and in the worst case scenario, some agent would have to wait for another for a long time period, causing many nodes on the high-level to be opened.

The CBS is one of the most recent algorithms for MAPF. It is also a two-level algorithm with many possible extensions, that could improve performance. On high-level, we could implement prioritizing conflicts according to some heuristic rules, or, we could use the concept of corridor symmetry to resolve corridor conflicts in a smart way[21]. On the low-level, we just need gTSP solver, that would be capable of blocking specific edges at specific time. The solver could be heuristic or optimal, depending on the instance size.

The last recent contribution is the Branch-and-Cut-and-Price algorithm. Authors have shown, that it has potential to outperform CBS, especially on larger instances. Furthermore, this approach is also capable of domain-specific reasoning to improve its performance. To incorporate gTSP into the framework, we would need to implement a *pricer* – an algorithm, that solves single-agent problem with modified objective function to find improving tours.

Heuristic

From the heuristic approaches, prioritized planning seems promising. Because its minimal overhead, it would most likely enable us to use optimal gTSP algorithm for finding the tours. Furthermore, because the warehouse is using scattered storage, the algorithm would still have some degrees of freedom even when optimizing the tour for the last agents. In order to improve the quality of solutions, we could make heuristic rules for determining the order of agents, or recalculate the solution with different ordering of agents, if the tour was far longer than optimal for one of the agents.

push and swap?

3.2 Conflict based search

4. Testing

4.1 Testing methodology

4.2 Results

4.3 Discussion

Conclusion

Bibliography

[1] J.A. Tompkins, J.A. White, Y.A. Bozer, and J.M.A. Tanchoco. *Facilities Planning*. Wiley, 2010.

[2] H Donald Ratliff and Arnon S Rosenthal. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations research*, 31(3):507–521, 1983.

[3] Kees Jan Roodbergen and René De Koster. Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43, 2001.

[4] Ivan Žulj, Christoph H Glock, Eric H Grosse, and Michael Schneider. Picker routing and storage-assignment strategies for precedence-constrained order picking. *Computers & Industrial Engineering*, 123:338–347, 2018.

[5] Randolph W Hall. Distance approximations for routing manual pickers in a warehouse. *IIE transactions*, 25(4):76–87, 1993.

[6] TS Vaughan. The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37(4):881–897, 1999.

[7] Kees Jan Roodbergen and RenÉde Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001.

[8] Jason Chao-Hsien Pan and Ming-Hung Wu. Throughput analysis for order picking system with multiple pickers and aisle congestion considerations. *Computers & Operations Research*, 39(7):1661–1672, 2012.

[9] Brian L Heath, Frank W Ciarallo, and Raymond R Hill. An agent-based modeling approach to analyze the impact of warehouse congestion on cost and performance. *The International Journal of Advanced Manufacturing Technology*, 67(1-4):563–574, 2013.

[10] M Klodawski, R Jachimowski, I Jacyna-Golda, and M Izdebski. Simulation analysis of order picking efficiency with congestion situations. *International Journal of Simulation Modelling*, 17(3):431–443, 2018.

[11] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Barták. Multi-agent pathfinding: Definitions, variants, and benchmarks. *CoRR*, abs/1906.08291, 2019.

[12] Lucie Pansart, Nicolas Catusse, and Hadrien Cambazard. Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127, 2018.

[13] Hadrien Cambazard and Nicolas Catusse. Fixed-parameter algorithms for rectilinear steiner tree and rectilinear traveling salesman problem in the plane. *CoRR*, abs/1512.06649, 2015.

[14] Felix Weidinger. Picker routing in rectangular mixed shelves warehouses. *Computers & Operations Research*, 95:139–150, 2018.

[15] Christophe Theys, Olli Bräysy, Wout Dullaert, and Birger Raa. Using a tsp heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763, 2010.

[16] Charles G Petersen. An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, 1997.

[17] Fangyu Chen, Hongwei Wang, Chao Qi, and Yong Xie. An ant colony optimization routing algorithm for two order pickers with congestion consideration. *Computers & Industrial Engineering*, 66(1):77–85, 2013.

[18] Albert H Schrotenboer, Susanne Wruck, Kees Jan Roodbergen, Marjolein Veenstra, and Arjan S Dijkstra. Order picker routing with product returns and interaction delays. *International Journal of Production Research*, 55(21):6394–6406, 2017.

[19] Fangyu Chen, Hongwei Wang, Yong Xie, and Chao Qi. An aco-based online routing method for multiple order pickers with congestion consideration in warehouse. *Journal of Intelligent Manufacturing*, 27(2):389–408, 2016.

[20] Roberta De Santis, Roberto Montanari, Giuseppe Vignali, and Eleonora Bottani. An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research*, 267(1):120–137, 2018.

[21] Jiaoyang Li, Graeme Gange, Daniel Harabor, Peter J Stuckey, Hang Ma, and Sven Koenig. New techniques for pairwise symmetry breaking in multi-agent path finding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 193–201, 2020.

List of Figures

1.1 Warehouse layout 3
1.2 High bay storage 3

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment