Optimizing Warehouse Operations

Casper Bazelmans[1], Frans Fonville[1], Han Hoogeveen[2], Johann Hurink[3], Jan Posthoorn[2], Corné Suijkerbuijk[4]

Abstract

We consider the problem of order picking in a warehouse from which items have to be picked to fulfill orders. The items are stored in (possibly multiple) positions along the aisles. The warehouse is divided in zones, which can consist of multiple aisles, that are served by one picker per zone. The picked orders are put in a tote with a limited capacity, after which the filled totes are transported to a waiting room. Next, a set of totes are moved to the gathering room, where the items are gathered to constitute a set of at most N orders. The fulfilled orders are then sent to the final station for wrapping up. The goal is to minimize the total time needed for the pickers, where the workload of each individual picker should be approximately the same.

To solve this problem we must specify in which order the totes are filled. First of all, this depends on how the set of orders are decomposed into subsets. Next, when a item is available in several zones, we must decide from which aisle to pick it, such that we can fill the totes in the best way.

[1]Fontys University of Applied Sciences, The Netherlands
[2]Utrecht University, The Netherlands
[3]Twente University, The Netherlands
[4]Sioux/LIME, The Netherlands

We used a heuristic approach to solve this order picking problem. Since an exact approach was not feasible, we developed a heuristic approach that first uses a constructive heuristic to generate an initial solution. Details are given on how the solution could be improved further using local search.

KEYWORDS: OPTIMIZATION, WAREHOUSING, LOCAL SEARCH, HEURISTICS

5.1 Introduction

The problem we worked on was about creating efficient pick-batches for order picking in a warehouse. These warehouses had a special layout in which certain items could be placed in multiple locations which reduces picking times. The warehouses were split into a flexible number of zones and each zone contains multiple items. A consequence of this layout is that some orders with multiple different items can be picked in multiple different zones. The goal is to assign orders to pick-batches so that as little zones as possible are opened and the workload per zone is balanced. The approach needs to quickly find a good solution for different warehouses with different numbers of zones.

5.2 Definitions

Before we explain the problem we worked on and our approach to solve it in more detail, we will first introduce some definitions and concepts.

SKU or Item Stock Keeping Unit, also sometimes called item, is a type of item that is stored in the warehouse at a certain location. A SKU can also be stored in multiple locations, each with their own stock level. We call the number of different zones that an SKU is stored in the frequency of that SKU or item. As an example we here take 1, 2 or 5.

Order A set of SKU's with an order quantity for each SKU. Orders with a single SKU are defined as single-item orders. Order with multiple SKU's are defined as multi-item orders.

Put wall A put wall is a closet that is open on both sides that is used to sort all the picked items into multi-item orders for customers such that they can be packaged. This wall can contain at most N multi-item orders at once.

Minibatch A set of multi-item orders that is distributed over a set of pickbatches that can be picked in parallel and then processed in the put wall.

Pickbatch A set of SKU's that need to be gathered from a zone and put into totes. A tote can contain at most T items.

Zone A section of the warehouse where items can be stored. An item can have at most one location in a zone, but it does not have to be present in a zone.

Tote A tote is a box that is used for order picking. A tote can hold T items at the same time. At most one zone is allowed for filling a tote during picking.

5.3 Problem Description

We are given a set of orders $\{O_1, O_2, \ldots, O_n\}$ in advance. Each order O consists of a set of Items/SKU's and quantities $\{(I_1, Q_1), (I_2, Q_2), \ldots, (I_m, Q_m)\}$. For each item we know the in which zones $Z_I \subseteq \{Z_1, Z_2, \ldots, Z_n\}$ it can be picked and we know the stock levels for each of these zones. In our problem there are two types of orders. We consider single-item orders and multi-item orders.

Single-item orders need to be picked with one tote. A tote can contain multiple single-item orders.

Multi-item orders need to be grouped in minibatches. A minibatch is just a set of at most N multi-item orders. This size restriction comes from the size of a put wall, since each minibatch is later sorted in a put wall. We also need to divide each minibatch into pickbatches. A pickbatch is a set of items and quantities that are all picked in the same zone. these items need to be picked by an operator by putting them into totes. Each tote can hold at most T physical items at once. This

T does not consider the size or shape of the items that go into a tote, but this number has to be chosen such that it works in practice.

The goal is to pick all orders such that the total number of totes needed is minimized. The total number of totes is the sum of totes needed for single-item and multi-item orders. Preferably the workload over the zones is balanced. The workload in a zone is the number of physical items that need to be picked in that zone.

5.4 The Algorithm

We had the following idea for our algorithm. We determined that an exact approach would not work because of flexibility in the problem. If we determine the minibatches we do not yet know how good these minibatches are because we will need to determine the pickbatches before we can do that. This makes solving the problem exactly infeasible for almost all practical instances. Because of this we choose to develop a heuristic solution approach. We concluded that it would be a good idea to work on creating a heuristic starting solution and developed ideas for local search to be able to improve this solution further.

The algorithms that we developed can be described on a high level by the following steps.

Step 1: Constructive heuristic for creating mini-batches,

Step 2: Assignment of pickbatches for all of the mini-batches,

Step 3: Balancing the workload with the single-item orders,

Step 4: Local search on the minibatches,

We will first assign all the multi-item orders to minibatches in such a way that orders that can be picked in the same zones are grouped in one minibatch. After this we divide each minibatch into pickbatches while trying to minimize the total number of totes used. Up until this point we have not considered the balancing of the workload yet, and because we expect that there are many more single-item orders than multi-item orders we believe that postponing the balancing is fine. At this stage we assign all the single-item orders to the zones in such a

way that the number of totes is minimized while keeping the workload per zone roughly balanced. After this step we have our initial solution. This solution or any other solution can then be improved with a local search based approach.

In the next section we will explain the four steps and the ideas behind them in more detail and also give some advice about possible improvements that can be made for each step.

5.4.1 Step 1: Creating Minibatches

Our first step is the assignment of the orders to the minibatches. Before we do this we first compute some metrics that will help us in the process of grouping similar orders together to create minibatches. The first metric we compute is called the *ZoneScore* and it resembles the complexity of an order. The zone score is computed for each order and is determined by the number of zones in which the item can be found.

$$ZoneScore(O) = \sum_{I \in O} \frac{1}{freq(I)} \tag{5.1}$$

In Equation 5.1 the freq(Item) function returns the number of zones in which an item can be found, for example, 1, 2 or 5 zones. O refers to an order, I refers to an item in an order. We start out by computing the zone score for each order. The idea of the zone score is that it resembles the complexity of an order and the number of zones that the order requires us to visit. Equation 5.1 shows how we compute the zone score for a given order.

Next we look at the total number of orders we have in our order set. Because we know that a mini-batch can contain at most N orders because of the put wall, we can estimate the number of minibatches that we are going to need in advance. We estimate the number of minibatches $n = (\#MultiItemOrders/N) + slack$. The slack term in the formula for the number of minibatches is a parameter because the minibatches do not have to be filled all the way. Based on the total number of orders and the maximum minibatch size N, we create s empty minibatches and take the s most complex orders (s orders with the highest zone score) and add each one of them to a different minibatch.

After sorting all remaining unassigned orders from smallest to largest zone score, we can assign all orders to the minibatches one by one, until all orders have been assigned.

For a given order O, we estimate the match between the order and a minibatch using a fit score function. Our goal is add each order to a minibatch that fits the best with this order. We estimate for a given Order O the match between the order and a mini-batch with fit score function.

$$P_M(z) = \prod_{I \in M_z} (1 - \frac{1}{freq(I)}) \tag{5.2}$$

$$p_O(z) = \sum_{I \in O_z} freq(I) \tag{5.3}$$

$$FitScore(O, M) = |M|^{w_1} \cdot \sum_{z=1}^{Z} P_M(z) \cdot p_O(z) \tag{5.4}$$

The fit score represents how well an order fits in minibatch M. The score is based on the probability of visiting zone z for a minibatch M, and the importance of zone z for an order. The more items of the order can be found in zone z, the higher the importance. For each order, fit scores are calculated with respect to each minibatch. The order will be assigned to the minibatch with the best fit score. A weight parameter was introduced to balance between adding orders to minibatches with the best fit score and adding orders to the smallest minibatches.

The $P_M(z)$ is the probability that minibatch M will not use zone z. M_z and O_z are the sets of all Items in a minibatch or order respectively, that can be picked in zone z, these sets can be empty. $P_M(z)$ will be 1 if we have no item in the minibatch that can be picked in zone z and 0 if we have an item that must be picked in zone z and otherwise it will have a value in between that represents the probability of not using that zone. Z is the total number of zones in the warehouse.

$p_O(z)$ is the importance of zone z for Order O. This value will be 0 if no item in this order can be picked in zone x, and it will increase if more items in this orders can be picked in this zone.

$FitScore(O, M)$ represents how well an Order O fits with a minibatch T. A lower fit score means a better fit so if we want to pick a

mini-batch to which we are going to add our order than we compute the fit score between for each minibatch with our order and we add the order to the minibatch that gives the lowest score that does not yet contain N orders. The fit score formula shown in Equation 5.4 contains a parameter w_1 which can be used to create a balance between adding orders to the smaller minibatches or by reducing the importance of the size and focussing more on the fit between the currently selected orders and the order we are considering to add.

The summation part of the fit score formula cannot increase if we add more items to a minibatch so to compensate, because the for each zone the chance that it will not be opened will only decrease or stay the same if we add more items to a minibatch. This is why we also take into account the number of orders that is already present in this minibatch in our fit score formula.

Now that we have added all the multi-item orders to our minibatches we can move on to the next step, creating the pickbatches.

Advise on possible improvements for step 1

In this section we will describe some of the ideas we had, which we could not implement and test, but we did think that these could be valuable additions to the algorithm.

- When putting the most complex n multi-item orders in a separate minibatch, it is not compared how well these most complex ones fits to each other. Instead, the most non-overlapping n orders can be put in a separate minibatch.
- Split the set of orders into 2 sets. Set 1 contains all the orders which contain exactly 2 items/SKU's for which the frequency of these items/SKU's has the form of $(1, 5)$, $(2, 5)$ or $(5, 5)$. Set 2 contains all the remaining multi-item orders. The idea behind this split is to seperate the orders into two groups where we have a group of large and versatile orders and a group of smaller orders with some flexibility that we can use to fine tune our minibatches by adding this Set 1 last.
- Now assign all the orders in set 2 to minibatches like we described in Section 5.4.1.

- Already assigning certain Orders to certain zones. We could try to reduce the number of zones we consider for each order in each minibatch based on the zones that are already fixed in a minibatch or the ones that will very likely be chosen. This filtering should not be too aggressive as it might result in an infeasible solutions later on. The idea of this step is to make the fit score a better representation of the zones that will be selected later on in step 2, the pickbatch assignment.

- Now assign the remaining orders to minibatches like we described in Section 5.4.1, these are the orders in set 1.

- Computing bounds for the minibatches. We could compute the minimum and maximum number of items that can be picked in each zone for each minibatch. In the assignment of set 1 to the minibatches try to make sure that at least on of these numbers, the minimum or maximum, is close, but below a T fold. You could try and make the numbers a $T-1$ fold for example. We want a good indication of what a minibatch will look like as early as possible and this might help with that. The idea of the $T-1$ fold is to be a bit more on the safe side, because going over the T fold will make our solutions much worse and only getting $T-1$ items in a zone instead of T is not a real problem.

5.4.2 Step 2: Assigning the Pickbatches

We have assigned all orders to the minibatches, but because many items can be picked in multiple zones we still have to make this assignment, which determines the pickbatches. We will model this assignment problem as a flow problem for each minibatch individually. We create a bipartite graph with all the items with frequency 2 or frequency 5 on the left, and with all the zones on the right. We do not have to include all the frequency 1 items because their zone assignment is fixed. We create a source and connect it to all the items, and each connection has a capacity equal to the quantity of the corresponding item/SKU. We connect each item to all the zones in which it can be picked with infinite capacity. And we connect each zone to a sink with a capacity set to the sum of the numbers of all the SKU‘s that we have to pick in this zone

modulo T. This will give all the zones without any fixed items/SKU's a capacity of 0 and all the other zones a capacity such that if we fill it, we exactly get a T-fold in the number of items that we need to get in this zone. For example, if we already know that 6 items will need to be picked in a zone then the capacity on that connection will be set to $M - 6$. With these capacities we now solve the max flow problem but it can occur that not all SKU's have received a full assignment of their full count of items we need to retrieve. In this case we look at the incoming residual capacity for each zone and we open the zone with maximal incoming residual capacity and solve the flow problem again. We can repeat this step until all the quantities of each of the SKU's in this minibatch is assigned to a zone. At this point we have determined all the pickbatches.

Advice on possible improvements for step 2

- Instead of opening the zone with maximal incoming residual capacity, multiple options can be evaluated in order to make it more optimal.

- We can keep the stock of each SKU into account using a global constraint. We keep track for each SKU how much we pick at that location over all the minibatches. If we violate on of these constraints somewhere during step 2, we can set the capacity of the arc connecting this SKU to the zone where the stock violation occurred to the remaining stock of this SKU. In this way we are guaranteed to not violate this constraint anymore.

- We could also add weights to the arcs in this flow graph that represent some estimation of the walking time. For each zone we already have a set of items that we are going to pick in that zone. Now we set the arc weight to 0 for any arc going from an SKU to this zone if the location of this SKU is closer than the furthest SKU we are already picking in this zone. And otherwise we set the weight to something proportional to the how much further we have to enter the zone compared to the current furthest point we need to visit in this zone.

- Adding single-item order can be considered in order the fill up gaps in the multi-item order pickbatches. This reduces the number of totes but increases can increase the time needed for sorting the pickbatches into orders and the packaging time. The impact of this will need to be investigated to see if this would be worthwhile.

5.4.3 Step 3: Balancing Workload

We still have the single-item orders left and we will use them to try and balance the workload while keeping the number of totes used to a minimum. These orders are processed in a different manner than the multi-item orders and because of this these types of orders cannot be put in the same totes. We sort all the orders based on two criteria. We first sort on frequency in ascending order and secondly on quantity in descending order. This means we will likely start with a single-item order that is only present in one zone of which we need a large quantity and we will probably end with a single-item order which can be picked in five zones of which we only need a single physical item. For each order we look at all the zones it can be placed in and then we look for the zone in which the number of extra totes needed to add this order is minimal. It can be minimal in multiple zones so our second selection criterion is which zone has the current lowest workload. Then we add it to this zone and we again want it to be as close to a multiple of T as possible. We tested our approach with a single problem instance. The results of this can be seen in Figure 5.1.

Advise on possible improvements for step 3

Allowing the single- and multi-item orders to be grouped in the same totes will cause more work to be done later on in the sort-to-order part of the chain, but it might be worth investigating what the benefits for the order picking part of the process would be if we would allow this grouping. You would still need to consider the maximum capacity of N orders of a minibatch when doing this. Another change that can be made if the workload balancing does not seem to work satisfactory is to swap the order of the rules for assigning orders to zones. We could also

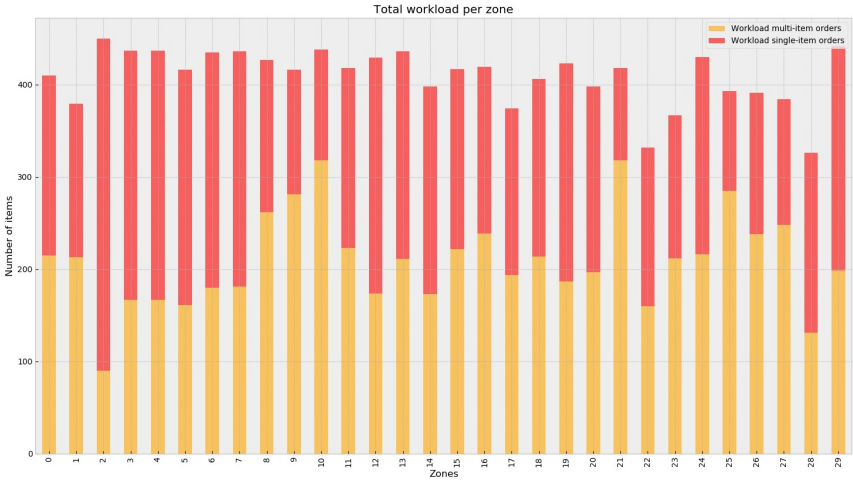

Figure 5.1: The resulting workload per zone after adding the single-item orders

start out by picking the zone with minimal workload and then look at the totes instead of the other way around.

5.4.4 Step 4: Local Search

After all the previous steps we now have a solution. But it might be the case that we are not yet satisfied with this solution. In this section we will describe some an approach that can be used to further improve the current solution. This would also work with any other solution to the same problem. The idea we had was to apply local search. We did not implement this part because of the time restrictions, but we are convinced that this addition would be very beneficial to the solution quality.

There are two parts of the solution that we can use local search on. The first part is the minibatches and the second part is the single-item orders. For the minibatches we propose the following local search operators.

- Move an order to a different minibatch
- Swap two orders between different minibatches

After applying such a change to the minibatches we will need to resolve the pickbatch problem. But we can use the fact that the change we made to the two affected minibatches is quite small. We can use our previous solution of the pickbatch assignment problem as a basis for our new pickbatch assignment. We again create the flow network and fill in all the previous flow values that we calculated. From this point it is quite easy and fast to resolve the pickbatch problem.

The orders that are moved or swapped could be selected at random or some bias could be introduced. It is likely very beneficial to remove/move an order from a minibatch that is the only order which is picked in a certain zone. So some weighted selection might improve the performance.

For the single-item orders a similar approach seems suitable. Moving or swapping orders to different zones is much easier for the single-item orders because we do not have to resolve the pickbatch problem.

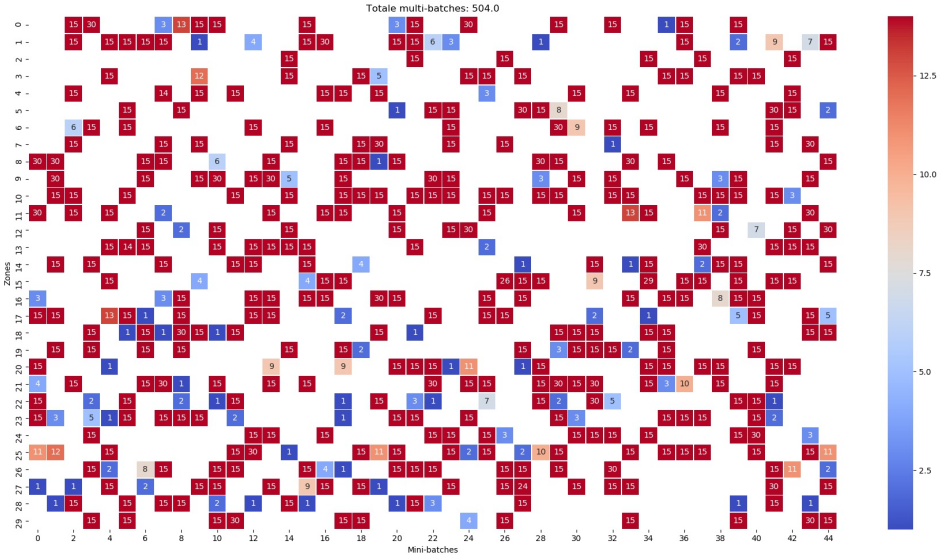

Figure 5.2: The resulting pickbatches per zone and minibatch for our instance

5.5 Results

In the end we implemented everything from Section 5.4.1 till 5.4.3 and did not implement the local search part described in Section 5.4.4. We applied our algorithm to a single problem instance and the result can be seen in Figures 5.1 and 5.2. The performance of our algorithm was comparable to the provided benchmark. Further evaluation of the algorithm is recommended to verify the performance and flexibility of the algorithm when using different problem instances, e.g. with a different zone configuration.

VDLreport