2018

Dynamic Batching for Order Picking in Warehouses

Jelmer Pier van der Gaast
University of Groningen, j.p.van.der.gaast@rug.nl

Bolor Jargalsaikhan
University of Groningen, b.jargalsaikhan@rug.nl

Kees Jan Roodbergen
University of Groningen, k.j.roodbergen@rug.nl

Dynamic batching for order picking in warehouses

J.P. van der Gaast
Faculty of Economics and Business
University of Groningen
j.p.van.der.gaast@rug.nl

B. Jargalsaikhan
Faculty of Economics and Business
University of Groningen
b.jargalsaikhan@rug.nl

K.J. Roodbergen
Faculty of Economics and Business
University of Groningen
k.j.roodbergen@rug.nl

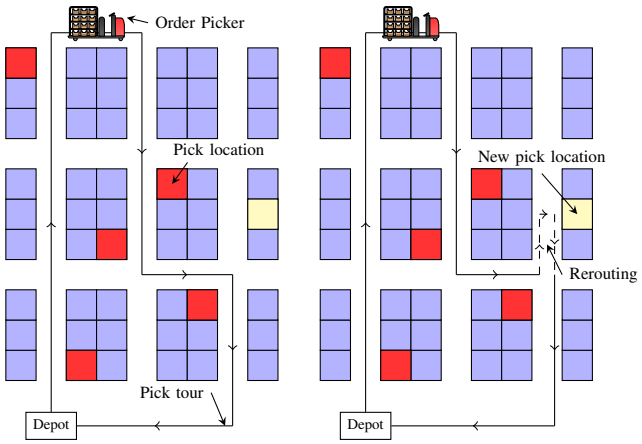
***Abstract*—Dynamic batch picking is characterized by combining product demand from multiple customer orders into one pick tour where new orders are continuously received. Using modern order-picking aids, updated picking instructions can be included in the current pick tours which allows pickers to be re-routed to pick for new orders even when they already started a pick tour. We develop a mathematical model for dynamic batch picking that minimizes the order throughput time of incoming customer orders. In case of new order arrivals, we can quickly re-optimize the model and determine new updated pick tours. This allows for short order throughput times and ensures that warehouse companies can set their order cut-off times as late as possible while still guaranteeing that orders can be delivered next day or in some cases even the same day.**

I. Introduction

E-commerce fulfillment competition evolves around cheap, speedy, and time-definite delivery. This puts a greater emphasis on the most critical operation in a warehouse, order picking. Order picking, the process of retrieving customer orders from their storage locations, needs to be robustly designed and optimally controlled in order for a warehouse to operate efficiently [15]. Any under performance in order picking can lead to unsatisfactory service and high operational cost for the warehouse, and consequently for the whole supply chain [15]. Therefore, many warehouses invest heavily in state-of-the-art order picking solutions to increase productivity and reduce any variability associated with order picking, e.g., in order arrivals, and in picking times.

A common way to organize the picking process where daily a large number of customer orders needs to be picked is *batch picking*. Batch picking is a picker-to-parts order picking method in which the demand from multiple orders is used to form pick batches. Pick tours are constructed for each pick batch to minimize the total travel time of the order picker or any other performance objective. A drawback of this approach is that batch formation takes time, and, as customers demand shorter lead times, more efficient ways to organize the order picking process exist.

In this paper we study *dynamic batch picking*. We consider an online setting where new customer orders are received continuously. In order to shorten batch formation, dynamic batch picking allows arriving customer orders to be directly included or swapped with current picks in one of the current pick tours, while taking into account potential other future customer order arrivals (Figure 1). In addition, disruptions occurring from the arrival of an urgent order or by noticing



(a) Static batch picking (b) Dynamic batch picking

Figure 1: The difference between static and dynamic batch picking, ▫ denotes a storage location, ▪ denotes a picking location for the current pick tour and ▫ denotes a picking location for an incoming customer order that can be picked in the current pick tour in case of dynamic batch picking.

inaccuracies between products requested and products actually picked can easily be dealt with using dynamic order picking [12]. Using modern order-picking aids like pick-by-voice techniques or by a handheld terminal, updated picking instructions can be included in the current pick tours. Different from regular (static) batch picking where a new customer order will always be picked in a subsequent future pick tour [4, 6, 11], pickers can be re-routed to pick for new customer orders even when they already started a picking tour. This makes dynamic batch picking particularly suitable for warehouses of e-commerce companies that often experience high order arrival rates, smaller order sizes, larger order variety, and a demand for fast order deliveries.

The objective of this paper is to study dynamic batch picking. Currently, in the literature only simple heuristics for this problem are known [12, 1] or assume a fixed pick tour [9, 15]. We develop a mathematical model that minimizes the order throughput time of a customer order. Using column generation allows us to quickly re-optimize the model in case of a new order arrival and determine the new updated pick tours. The model allows us to study the performance gains of dynamic batch picking compared to static batch picking. In

addition, we study the effect of different routing methods and arrival distributions in the case of dynamic batch picking.

The organization of this paper is as follows. In Section II an overview of existing literature for batch picking systems is presented with a focus on online and dynamic batch picking models. In Section III, the model for dynamic batch picking is discussed and the corresponding notation used in this paper is given. In Section IV a detailed description of our optimization methods is provided. We extensively analyze the results of our model and optimization framework in Section V via computational experiments for a range of parameters. Finally, in Section VI we conclude and suggest some extensions of the model and further research topics.

II. Literature review

In this section we will first discuss literature on static batch picking which has been particular well studied over the past decades. Next, we focus on online batch picking, as well as, the related literature on dynamic batch picking.

As mentioned in Section I, in batch picking a set of customer orders are grouped in pick batches, each of which can be picked by a single picking tour. For each of these picking tours the picking locations are sequenced in such a way to ensure a good route through the warehouse [2]. One of the first to consider order picker routing in a warehouse and to also present an optimal polynomial-time algorithm to minimize order picker travel distance are Ratliff and Rosenthal [14]. They consider a warehouse with a single-block parallel-aisle layout with narrow aisles and each pick tour starts and ends at the central depot. Their algorithm is extended for multiple different warehouse layouts, e.g., De Koster and Van der Poort [3] consider the case that picked customer orders can be deposited at the head of every aisle. However, these optimal algorithms for order picker routing are hardly used in practice [10]. A reason for this is that order pickers find these routes confusing which can cause unsafe situations in the warehouse. Therefore, order picker routing is often done using heuristics, e.g., S-shape routing or Largest-Gap routing.

The literature on static batch picking is quite varied on the assumptions and warehouse layout considered. De Koster et al. [4] study two groups of heuristic algorithms; seed algorithms and time savings algorithms. They test their algorithms for a single-block parallel-aisle layout and consider two different routing strategies; S-shape and Largest-Gap. Seed algorithms are best in conjunction with S-shape and large pick batch sizes, whereas time savings algorithms perform best in conjunction with Largest-Gap and small pick batch sizes. Hong et al. [11] develop an integrated batching and sequencing procedure in order to minimize the total retrieval time (the sum of travel time, pick time and congestion delays). This allows, compared to traditional batching formulations, to determine the position of each batch in the batch release sequence. Matusiak et al. [13] consider a picker-to-parts warehouse where there are precedence constraints while picking a customer order. They consider a joint order batching and picker routing method which takes the precedence-constraints into account. Their method consists of two parts: an optimal A*-algorithm for the routing; and a simulated annealing algorithm for the batching which estimates the savings gained from batching more than two customer orders to avoid unnecessary routing. Gademann et al. [7] study static order batching in a parallel-aisle warehouse, where the objective is to minimize the maximum lead time of any of the batches. This is a common objective in parallel (or zoned) wave order picking operations. Gademann and Van de Velde [6] develop a branch-and-price optimization algorithm to minimize order picker travel distance. They model the problem as a generalized set partitioning problem and applied a column generation algorithm to solve its linear programming relaxation. Due to speed and complexity of the algorithm, the authors suggested for larger instances to use an iterated descent approximation algorithm.

All of the above order batching algorithms assume that demand is given at beginning of the planning period, i.e., the number of orders and for each order the order lines that are requested. However, for a lot of e-commerce companies this assumption does not hold since customer orders might arrive during the day that need to be processed by the end of the day. Therefore, in online batch picking new pick batches are created continuously during the day. Le-Duc and De Koster [5] consider a 2-block rectangular warehouse with the assumptions that orders arrive according to a Poisson process. Furthermore, they assume that picker routing is done using the S-shape heuristic. They determine the first and second moment of the order picker's travel time to pick a batch, which are used to estimate the average throughput time of a random order. Van Nieuwenhuyse and De Koster [16] study different online batch picking methods; variable time window batching and fixed time window batching. Their methods allow for arbitrary distributions for customer order size, picking time, sorting time, and setup times for picking or sorting a batch. For both methods the optimal batch sizes for a given workforce allocation can determined, or for the optimal allocation of workforce to the picking and sorting processes. Finally, Henn [10] considers an online batch picking in which the maximum completion time of the customer orders arriving within a certain time period has to be minimized. The author shows how heuristic approaches for offline order batching can be modified in order to deal with the online situation.

Finally, similar as for online batch picking, dynamic batch picking considers the case that orders are received continuously during the day. However, the difference is that dynamic batch picking allows arriving customer orders to be directly included or swapped with current picks in one of the current pick tours. One of the first to study a dynamic batch picking was Gong and De Koster [9]. They studied a single-block parallel-aisle warehouse where each pick tour has the same route. Orders arriving in real-time can be integrated in the current picking tour which subsequently changes the stops on the order picker's picking route. Van der Gaast [15] extended the work of [9] by studying the mean order throughput time of a random customer order, i.e., the time between a customer order entering the system until the whole order is delivered at the depot. Lu

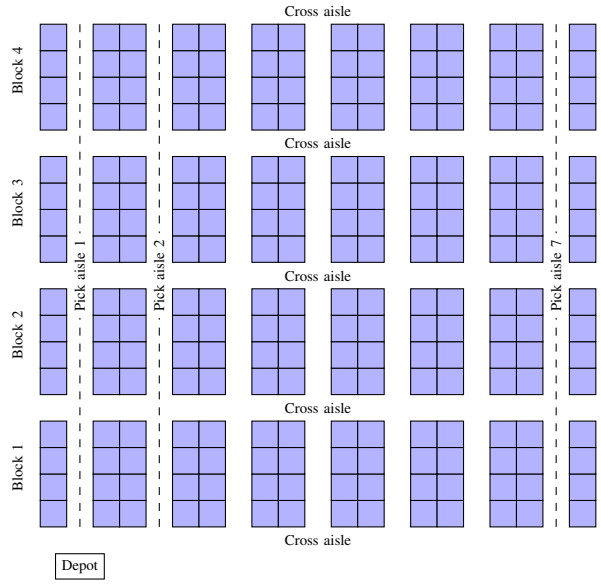


Figure 2: Example of a rectangular multi-block parallel-aisle warehouse layout with multiple cross aisles.

et al. [12] introduces an interventionist routing algorithm for optimizing the dynamic order picking routes. The authors study the case that batches are constructed on first-come-first-served basis where if a new customer order arrives it will be checked if it should be included in the current batch or not. Finally, Chen et al. [1] develop a non-parametric heuristic method named Green Area, which is independent of the parameters of a warehouse layout and the characteristics of customer orders. The heuristic checks if a new customer order can be picked downstream, otherwise it will be picked in a future pick tour.

In conclusion, the current models in the literature for dynamic batch picking either are heuristics [12, 1] or assume a fixed pick tour [9, 15]. In this paper we assume that order pickers can be rerouted in case of a new customer order arrival while ensuring the customer orders are batched optimally.

III. Warehouse description

In Figure 2 the layout of the picker-to-parts warehouse studied in this paper is shown. We assume a rectangular multi-block parallel-aisle layout where storage locations are located on both sides of the pick aisles. Within a pick aisle, this allows for two-sided picking, i.e., simultaneous picking from the right and left sides within an aisle. Different blocks are separated by a cross aisle which is used to change between blocks and allows for short-cutting parts of the warehouse.

In Figure 3 the different steps in dynamic order picking are presented. The order picking procedure can be described as follows. Initially, all order pickers start at the depot, i.e., the location where completed orders can be prepared for shipment. Eventually, new customer orders require to be picked and an order picker is assigned a pick tour, which determines the sequence in which storage locations are visited in order to pick the required products. The corresponding route is determined

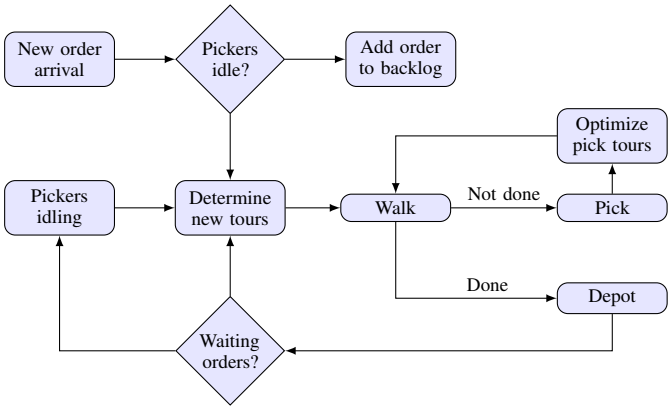


Figure 3: Flow diagram of the dynamic order picking system.

by a typical routing strategy, i.e., S-shape or Largest Gap, while considering the picking capacity of the order picker. Pick tours are constructed such that the mean order throughput time for all orders in the system is minimized for that given moment of time. We define the order throughput time of order i as the time between its arrival, AT_i, and the time the order is fully picked and delivered at the depot. Next, the order picker starts walking to the first location in its pick tour. Then, after a pick all the pick tours of all order pickers are re-optimized while taking into account any order arrival from the moment the pick tours were determined last. The pick tours of the picker can be modified; either by adding, removing, or swapping picks. Any unassigned pick is added to the backlog. We assume that picks for orders the order picker already picked for or any picks associated with the next picking location on the pick list cannot be modified. In addition, if no new orders entered the system after the last re-optimization, the order pickers continue with their current tours. This process repeats until the picker is instructed to return to the depot. Here the picker will drop off the collected orders such that they can be shipped to the customers. Afterwards, the order picker waits until to start a next pick tour or stops working at the end of his/her shift.

IV. Methodology

In the following section we describe the model, as well as, the solution method for analyzing dynamic batch picking. First, in Subsection IV-A the mathematical program of the problem is provided. Next, in Subsection IV-B the various cost factors in the model are described in order to calculate the mean order throughput time of an order. Then in Subsection IV-C a column generation algorithm is presented that solves the relaxation of the model described in Subsection IV-A and it is also describes how new columns (a tour) for the column generation algorithm are being generated. Finally, Subsection IV-D summarizes our complete solution method.

A. Model

In order to analyze dynamic batch picking we formulate the problem as a set partitioning problem. For any particular time t during daily operations, a customer order $i \in M$ is

either assigned to the current pick batch of an order picker (current batches) or to the backlog of orders for which an initial batching is made (future batches). Orders in the backlog will eventually be assigned to a current batch such that an order picker will pick them.

Let Ω_j be the set of all feasible batches for order picker $j \in J$ given its location l_j in the warehouse. A feasible batch $r_k \in \Omega_j$ for order picker j is characterized by a zero-one vector $a_{jr_k} = (a_{1jk}, \ldots, a_{mjk})$, where $a_{ijk} = 1$ if and only if order $i \in M$ is included in batch r_k. Note that batch r_k is feasible only if $\sum_{i \in M} a_{ijk} \leq q_j$, where q_j is the remaining picking capacity of order picker $j \in J$. Furthermore, let Ψ be the set of all feasible future batches. Similar, a feasible batch $r_t \in \Psi$ is characterized by a zero-one vector $b_{r_t} = (b_{1t}, \ldots, b_{mt})$, where $b_{it} = 1$ if and only if order $i \in M$ is included in batch r_t. Batch r_t is feasible only if $\sum_{i \in M} b_{it} \leq q$, where q is the picking capacity of an order picker. Let c_{jk} be the cost to pick the orders in batch $r_k \in \Omega_j$ for order picker $j \in J$ and $\bar{c}_t$ be the total cost to pick the orders in batch $r_t \in \Psi$ for a future batch, where the costs will be described more elaborately in Subsection IV-B. Furthermore, let θ_{jk} and δ_k be a zero-one variable such that:

$$\theta_{jk} = \begin{cases} 1 & \text{if batch } k \text{ is selected as the current tour of} \\ & \text{picker } j, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\delta_k = \begin{cases} 1 & \text{if batch } k \text{ is selected as a future tour,} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the model can be formulated as follows:

$$\min \quad \sum_{j \in J} \sum_{r_k \in \Omega_j} c_{jk} \theta_{jk} + \sum_{r_t \in \Psi} \bar{c}_t \delta_t \tag{1a}$$

$$\text{s.t.} \quad \sum_{j \in J} \sum_{r_k \in \Omega_j} a_{ijk} \theta_{jk} + \sum_{r_t \in \Psi} b_{it} \delta_t = 1 \qquad i \in M, \tag{1b}$$

$$\sum_{j \in J} \sum_{r_k \in \Omega_j} \theta_{jk} = 1 \qquad j \in J, \tag{1c}$$

$$\theta_{jk}, \delta_t \in \{0, 1\} \tag{1d}$$

The objective of the model (1a) to determine values of θ_{jk} and δ_k in order to minimize the total cost. Constraints (1b) ensure that every order $i \in M$ is assigned to either a current batch or to a future batch. Constraints (1c) define that each order picker $j \in J$ has a current batch. Finally, Constraints (1d) are the integrality constraints. In the next section, a description of how to calculate costs c_{jk} and $\bar{c}_t$ is provided.

B. Cost of pick tour

As explained in Section III, new pick tours are constructed every time a picker completes a pick. This requires optimizing Model (1) with updated picking information. Assume that at time t order picker $j \in J$ finishes a pick. Then, a updated batch $r_k \in \Omega_j$ is determined for the picker, which consists of the orders that are already picked or still need to be picked, as well as, the remaining orders which can potentially consist of orders that just arrived. The cost of batch $r_k \in \Omega_j$ is defined as the mean order throughput time of the orders included in the batch:

$$c_{jk} = \frac{1}{|r_k|} \sum_{i \in r_k} \Delta_{r_k}^{l_j} + E(\tau) + t - AT_i, \tag{2}$$

where $\Delta_{r_k}^{l_j}$ is the remaining time starting at location l_j to return to the depot while simultaneously picking all outstanding orders in batch $r_k \in \Omega_j$ Note that $\Delta_{r_k}^{l_j}$ can be determined by a routing heuristic, e.g., S-shape or Largest Gap. In addition, $E(\tau)$ is the additional time due to replanning, since new orders can be assigned to the current pick tour which would increase the time before the picker returns to the depot. Due to the constant changing behavior of the system because of new arriving customer orders, we roughly estimate the additional replanning time per batch as follows;

$$E(\tau) = \max\left(E(\sigma) - |r_k|, 0\right) E(\varpi),$$

where $E(\sigma)$ is the average number of picks per tour and $E(\varpi)$ is the average contribution of one order to the tour duration. We estimate both averages by the actual realizations from orders picked in previous batches during the day.

Next, if an order is not assigned to an order picker, it is added to the backlog for which initial batches are constructed. The cost of batch $r_t \in \Psi$ is defined as the mean order throughput time of the orders included in the batch

$$\bar{c}_t = \frac{1}{|r_t|} \sum_{i \in r_t} \Delta_{r_l}^{d} + E(\tau) + E(\phi) + t - AT_i, \tag{3}$$

where $\Delta_{r_t}^{d}$ is the tour duration starting and returning to the depot while picking all the required products in batch $r_t \in \Psi$. The additional time due to replanning $E(\tau)$ is defined similar as before, while $E(\phi)$ is defined as the average waiting time to start a future tour. We define this waiting time as follows:

$$E(\phi) = E(\xi)\, OT/q,$$

where $E(\xi)$ is the average tour length, OT is the number of outstanding orders at time t and q is the order picker capacity. Again, we estimate $E(\phi)$ and $E(\xi)$ by the actual realizations from orders picked in previous batches during the day.

C. Column generation

In order to optimize Model (1) we apply column generation. We initially solve the linear programming relaxation of Model (1) on a subset of current and future batches (master problem). We verify the global optimality of the relaxation using a pricing algorithm by checking if there are new current or future batches with negative reduced cost that can be added, since they might improve the solution value of the model. This procedure is repeated until no more batches with negative reduced cost can be found anymore. If the solution is integral, then we have also found an optimal solution for the original integer linear programming problem. If it is not, then a branch-and-bound algorithm is required to find an optimal solution.

Since we consider a minimization problem, we know from the theory of linear programming that a solution to the linear programming relaxation is optimal if the reduced cost of each variable, that is, batch, is non-negative. Therefore, the reduced cost of a current batch $r_k \in \Omega_j$, for picker $j \in J$ is given by:

$$d_{jk} = \Delta^j_{r_t} - \sum_{i \in M} a_{ijk}\lambda_i - \mu_j, \tag{4}$$

where $\lambda_i, i \in M$ are the given values of the dual variables corresponding to the conditions of constraint (1b) and $\mu_1, \ldots, \mu_J$ are the given values of the dual variables corresponding to the conditions of constraint (1c). On the other hand, the reduced cost of a future batch $r_t \in \Psi$ is given by:

$$\bar{d}_t = \Delta^d_{r_l} - \sum_{i \in M} b_{it}\lambda_i. \tag{5}$$

To find out whether the current solution is optimal, we need to determine whether there exists a batch $r_k \in \Omega_j$, $j \in J$ or $r_t \in \Psi$ with a negative reduced cost. In order to find a batch with negative reduced cost we need to solve the pricing problem. In the next subsections we will describe two strategies how generate these batches for the pricing problem.

1) Tabu search: The first method we use to rapidly generate batches with negative reduced cost is Tabu search [8]. Tabu search is a local search algorithm that has been applied successfully in the past to solve difficult problems in many fields. The algorithm starts with an initial solution and by simple operators different parts of the search space will be explored every iteration. Out of the neighborhood of the current solution, feasible solutions reached by a single operator, the best neighbor in terms of objective is chosen and is set as the current solution. Tabu search allows for non-improving solution when a local minima is encountered and cycling back to previous solution is avoided using a Tabu list that forbids certain moves for a couple of iterations. Then after a fixed number of iterations, I_{max}, the best batch is returned. An important aspect of Tabu search is to diversify the search such that different parts of the search space are explored.

The Tabu search that we propose for dynamic batch picking tries to find batches with negative reduced cost to include in the set of feasible batches Ω_j for order picker $j \in J$ or for the set of feasible batches Ψ for the future tours. Note that we run the Tabu search $|J| + 1$ different times, each time for the different set of batches. Starting from an initial solution, every iteration we apply two different operators. The first operator inserts a customer order at every possible position into the current solution, whereas the second deletes a customer order from the current solution. For each neighbor we check its feasibility in terms of the picking capacity and whether the operator to reach the neighbor is not in the Tabu list. In addition, we assume that the search space only consists of feasible solutions.

In order to diversify the search we restart the Tabu search multiple times with different starting solutions. The set of starting solution are given by the batches associated with the basic variables of the current restricted master problem solution. All these batches are good initial candidates because they have a zero reduced cost. In a column generation context, all negative reduced cost best neighbors are retained to be added to the restricted master problem. Also, the algorithm is halted either when all initial solutions have been used or when a maximum number C_{max} of negative reduced cost batches have been found.

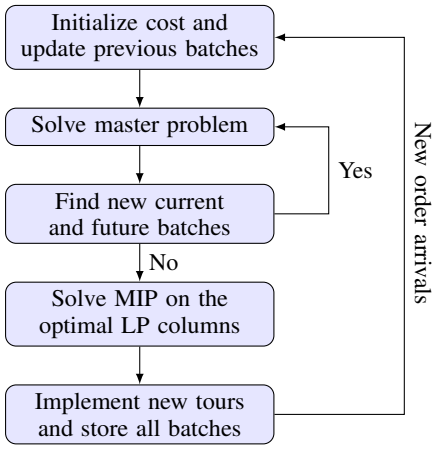


Figure 4: Algorithm for dynamic order picking.

2) Branch-and-bound pricing: In case the Tabu search did not found any batches with negative reduced cost, we will resort to an exact method to find remaining batches with negative reduced cost. If no batches are found, then the current solution of linear relaxation of the master problem is optimal. Otherwise we add the new batches to the master problem and optimize the master problem again. The branch-and-bound algorithm we use to find batches with negative reduced cost either for Ω_j for order picker $j \in J$ or for the set of feasible batches Ψ is the similar to the one used in Gademann and Van de Velde [6]. For further information about the algorithm the reader is referred to this paper.

D. Complete algorithm

Finally, we summarize the complete algorithm to analyze dynamic batch picking (Figure 4). After an order picker completes a pick and new customer orders have arrived in the system, new pick tours are constructed by optimizing Model (1) with the updated picking information. The first step is to initialize $r_k \in \Omega_j$, $j \in J$ and $r_t \in \Psi$ and calculate the associated cost c_{jk} and $\bar{c}_t$ or update old batches from the previous optimization round. Afterwards, we optimize the master problem (linear programming relaxation of Model (1)). Then, we check via the Tabu search of Subsection IV-C1 and the Branch-and-bound algorithm of Subsection IV-C2 if there exists batches with negative reduced cost. If so we solve the master problem again with the updated sets of batches. If no batches are found with negative reduced cost, we optimize Model (1) again, but now with the integrality constraints enforced. We do not apply a branch-and-bound algorithm to find the best

Table I: Parameters of the system instances test set.

	Value
Dimensions	5 pick aisles, 5 blocks, 5 pick location per block
Number of pickers	1, 2, 3
Arrival rate, λ	$1/60, 1/55, \ldots, 1/20$
Routing method	NN, S, LG
Planning method	**S**tatic, **D**ynamic
Capacity picker, q	3

solution for Model (1) as the additional time spent finding this solution does not outweigh the improvement in solution quality based on our initial computational experiments. Finally, all the batches are saved for the next time Model (1) needs to be optimized and the new pick tours are implemented.

V. Results

In order to test the performance of dynamic batch picking we carry out a small initial numerical study on toy data. The data used in this experiment is shown in Table I. We assume a small multi-block parallel-aisle layout with 5 pick aisles, 5 blocks, and 5 pick locations on each side per block. The travel times between pick locations is 1 second, picking at a pick location takes 1 second on average and the depot is located in the bottom left corner of the warehouse. Next, we vary the number of order pickers between 1, 2, and 3. We assume that new customer orders arrive according to a Poisson process with rate λ, whereas the rate varies between $1/60, 1/55, \ldots, 1/20$. The routing strategy for the order pickers is assumed to be Nearest Neighbor, S-shape, or Largest Gap. Finally, the picking capacity of the order picker is assumed to be 3 customer orders.

In order to test the performance of dynamic batch picking, we assume two cases. The first case, *Static*, assumes that when an order picker starts a pick tour, the tour cannot be replanned anymore. Its pick batches are constructed on a first-come-first-serve basis. The second case, *Dynamic*, allows for replanning and is determined using the algorithm described in Subsection IV-D. In total we have 162 different instances and each instance is ran 5 times for a daily operation of 8 hours.

In Table II the results for the toy data is presented. First, in Table IIa we study the effect of higher arrival rates. For both Static (S) and Dynamic (D) it can be seen that as the arrival rate increases the order throughput time sharply increases, and the differences become bigger whereas dynamic batch picking performs significantly better when the arrival rate is $1/20$. Similar the number of orders in the backlog increases when the arrival rates increases. On average half of the dynamic batch picking routes are replanned once and the utilization of the pickers is lower compares to Static. This also results in less total walking distance for the order pickers. In Table IIb we can see the results for varying the number of pickers. When the amount of pickers is equal to three the results for Dynamic and Static are more or less the same, but when there is only one order picker dynamic batch picking performs way better compared to Static. Finally, in table IIc we can see the results of varying the routing method. All three routing heuristics, Nearest Neighbor (NN), S-shape (S), and Largest Gap (LG) perform almost identical to each other both in case of Static and Dynamic.

VI. Conclusion and further research

In this paper we studied dynamic batch picking. We developed a mathematical model that minimizes the order throughput time of a customer order. Using column generation allowed us to quickly re-optimize the model in case of a new order arrival and determine the new updated pick tours. The model allowed us to study the performance gains of dynamic batch picking compared to static batch picking.

Further research topics include robust route planning. In this case pick routes should be planned to take into future customer arrivals, i.e., if a new customer order arrives it can easily be included without too much rerouting. Secondly, the effect of product allocation on dynamic batch picking is worth investigating.

References

[1] F. Chen, Y. Wei, and H. Wang. "A heuristic based batching and assigning method for online customer orders". In: *Flexible Services and Manufacturing Journal* (2017), pp. 1–46.

[2] M. B. M. De Koster, T. Le-Duc, and K. J. Roodbergen. "Design and control of warehouse order picking: A literature review". In: *European Journal of Operational Research* 182.2 (2007), pp. 481–501.

[3] M. B. M. De Koster and E. S. Van der Poort. "Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions". In: *IIE Transactions* 30.5 (1998), pp. 469–480.

[4] M. B. M. De Koster, E. S. Van der Poort, and M. Wolters. "Efficient order batching methods in warehouses". In: *International Journal of Production Research* 37.7 (1999), pp. 1479–1504.

[5] T. Le-Duc and M. B. M. De Koster. "Travel time estimation and order batching in a 2-block warehouse". In: *European Journal of Operational Research* 176.1 (2007), pp. 374–388.

[6] A. J. R. M. Gademann and S. L. Van de Velde. "Order batching to minimize total travel time in a parallel-aisle warehouse". In: *IIE Transactions* 37.1 (2005), pp. 63–75.

[7] A. J. R. M. Gademann, J. P. Van den Berg, and H. H. Van der Hoff. "An order batching algorithm for wave picking in a parallel-aisle warehouse". In: *IIE Transactions* 33.5 (2001), pp. 385–398.

[8] F. Glover and M. Laguna. "Tabu Search". In: *Handbook of combinatorial optimization*. Springer, 2013, pp. 3261–3362.

[9] Y. Gong and M. B. M. De Koster. "A polling-based dynamic order picking system for online retailers". In: *IIE Transactions* 40.11 (2008), pp. 1070–1082.

Table II: The results for the toy data.

(a) The effect of varying arrival rates on different performance statistics

Arrivals	Plan	Order throughput time (s)	# orders in backlog	Tour duration (s)	# orders picked per tour	# replanned per tour	Picker utilization	Picker walking dist (m)
1/60	D	43.67 (0.86)	0.05 (0.01)	41.03 (0.77)	1.18 (0.02)	0.14 (0.02)	0.34 (0.01)	9155.38 (345.62)
	S	47.04 (1.16)	0.09 (0.01)	40.23 (0.72)	1.08 (0.01)	0.00 (0.00)	0.37 (0.02)	9838.71 (412.30)
1/55	D	44.23 (0.91)	0.05 (0.01)	41.27 (0.87)	1.20 (0.02)	0.15 (0.02)	0.37 (0.01)	9857.57 (310.93)
	S	47.76 (1.14)	0.11 (0.01)	40.28 (0.66)	1.10 (0.02)	0.00 (0.00)	0.39 (0.02)	10 502.03 (441.66)
1/50	D	45.08 (1.20)	0.07 (0.01)	41.55 (0.90)	1.22 (0.03)	0.17 (0.02)	0.40 (0.01)	10 638.44 (370.43)
	S	49.58 (1.57)	0.15 (0.02)	40.71 (0.78)	1.12 (0.02)	0.00 (0.00)	0.43 (0.02)	11 556.49 (488.21)
1/45	D	46.29 (1.21)	0.09 (0.02)	42.24 (0.73)	1.28 (0.02)	0.20 (0.02)	0.44 (0.01)	11 660.84 (343.78)
	S	50.70 (1.36)	0.18 (0.02)	40.93 (0.70)	1.15 (0.02)	0.00 (0.00)	0.47 (0.02)	12 589.77 (448.03)
1/40	D	47.38 (1.28)	0.12 (0.02)	42.63 (0.87)	1.31 (0.03)	0.22 (0.02)	0.48 (0.02)	12 829.42 (416.09)
	S	53.37 (1.80)	0.24 (0.03)	41.67 (0.82)	1.19 (0.02)	0.00 (0.00)	0.52 (0.02)	13 816.80 (421.51)
1/35	D	49.55 (1.35)	0.18 (0.02)	43.34 (0.88)	1.38 (0.03)	0.26 (0.02)	0.53 (0.01)	14 174.80 (389.76)
	S	56.65 (1.86)	0.33 (0.04)	42.56 (0.72)	1.26 (0.02)	0.00 (0.00)	0.57 (0.01)	15 113.51 (383.14)
1/30	D	52.89 (1.33)	0.29 (0.03)	44.38 (0.65)	1.49 (0.02)	0.32 (0.02)	0.59 (0.01)	15 778.49 (361.78)
	S	61.14 (2.02)	0.50 (0.05)	43.52 (0.72)	1.35 (0.03)	0.00 (0.00)	0.64 (0.01)	16 928.51 (354.02)
1/25	D	60.86 (3.95)	0.60 (0.15)	45.99 (0.69)	1.64 (0.03)	0.41 (0.02)	0.67 (0.01)	17 764.83 (387.50)
	S	76.48 (6.16)	1.11 (0.24)	45.93 (0.87)	1.54 (0.05)	0.00 (0.00)	0.71 (0.01)	19 004.79 (300.23)
1/20	D	103.91 (20.14)	2.84 (1.05)	47.30 (0.53)	1.87 (0.03)	0.59 (0.04)	0.76 (0.01)	20 008.30 (242.16)
	S	346.57 (157.90)	14.60 (7.99)	49.25 (0.74)	1.82 (0.03)	0.00 (0.00)	0.79 (0.01)	21 112.89 (344.06)

(b) The effect of varying number of pickers on different performance statistics

Pickers	Plan	Order throughput time (s)	# orders in backlog	Tour duration (s)	# orders picked per tour	# replanned per tour	Picker utilization	Picker walking dist (m)
1	D	80.86 (9.43)	1.34 (0.43)	49.13 (1.09)	1.90 (0.05)	0.56 (0.04)	0.72 (0.02)	18 968.79 (442.11)
	S	174.28 (56.49)	5.56 (2.78)	48.08 (1.03)	1.71 (0.05)	0.00 (0.00)	0.78 (0.02)	20 692.13 (463.34)
2	D	43.66 (0.77)	0.07 (0.01)	41.32 (0.67)	1.22 (0.02)	0.20 (0.02)	0.47 (0.01)	12 461.85 (353.87)
	S	47.65 (1.12)	0.17 (0.02)	40.83 (0.66)	1.13 (0.01)	0.00 (0.00)	0.50 (0.01)	13 318.34 (383.76)
3	D	40.10 (0.54)	0.02 (0.00)	39.46 (0.53)	1.07 (0.01)	0.07 (0.01)	0.35 (0.01)	9192.05 (260.04)
	S	41.17 (0.72)	0.04 (0.01)	39.46 (0.55)	1.03 (0.01)	0.00 (0.00)	0.36 (0.01)	9477.36 (350.63)

(c) The effect of varying routing methods on different performance statistics

Routing	Plan	Order throughput time (s)	# orders in backlog	Tour duration (s)	# orders picked per tour	# replanned per tour	Picker utilization	Picker walking dist (m)
LG	D	55.57 (3.79)	0.51 (0.15)	43.38 (0.72)	1.40 (0.03)	0.28 (0.02)	0.51 (0.01)	13 554.37 (314.59)
	S	85.06 (21.50)	1.77 (1.00)	42.77 (0.82)	1.29 (0.03)	0.00 (0.00)	0.54 (0.02)	14 495.81 (410.06)
NN	D	53.86 (3.69)	0.45 (0.15)	43.03 (0.80)	1.39 (0.02)	0.27 (0.02)	0.51 (0.01)	13 506.91 (361.12)
	S	91.45 (18.81)	2.15 (0.90)	42.53 (0.63)	1.28 (0.02)	0.00 (0.00)	0.54 (0.02)	14 477.99 (403.84)
S	D	55.20 (3.26)	0.47 (0.13)	43.50 (0.77)	1.40 (0.03)	0.28 (0.02)	0.51 (0.01)	13 561.40 (380.30)
	S	86.59 (18.02)	1.85 (0.90)	43.07 (0.79)	1.30 (0.03)	0.00 (0.00)	0.55 (0.01)	14 514.04 (383.83)

[10] S. Henn. "Algorithms for on-line order batching in an order picking warehouse". In: *Computers & Operations Research* 39.11 (2012), pp. 2549–2563.

[11] S. Hong, A. L. Johnson, and B. A. Peters. "Batch picking in narrow-aisle order picking systems with consideration for picker blocking". In: *European Journal of Operational Research* 221.3 (2012), pp. 557–570.

[12] W. Lu, D. McFarlane, V. Giannikas, and Q. Zhang. "An algorithm for dynamic order-picking in warehouse operations". In: *European Journal of Operational Research* 248.1 (2016), pp. 107–122.

[13] M. Matusiak, M. B. M. De Koster, L. Kroon, and J. Saarinen. "A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse". In: *European Journal of Operational Research* 236.3 (2014), pp. 968–977.

[14] H. D. Ratliff and A. S. Rosenthal. "Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem". In: *Operations Research* 31.3 (1983), pp. 507–521.

[15] J. P. Van der Gaast. *Stochastic models for order picking systems*. EPS-2016-398-LIS. ERIM, 2016.

[16] I. Van Nieuwenhuyse and M. B. M. De Koster. "Evaluating order throughput time in 2-block warehouses with time window batching". In: *International Journal of Production Economics* 121.2 (2009), pp. 654–664.