https://pubsonline.informs.org/journal/ijaa

INFORMS JOURNAL ON APPLIED ANALYTICS
Vol. 53, No. 4, July–August 2023, pp. 266–282
ISSN 2644-0865 (print), ISSN 2644-0873 (online)

Algorithm for Robotic Picking in Amazon Fulfillment Centers Enables Humans and Robots to Work Together Effectively

Russell Allgor,[a] **Tolga Cezik,**[a] **Daniel Chen**[a,*]

[a] Amazon.com, Seattle, Washington 98109
*Corresponding author
Contact: rallgor@amazon.com (RA); cezikm@amazon.com (TC); chonglic@amazon.com, https://orcid.org/0000-0001-6538-5608 (DC)

Received: April 5, 2021
Revised: April 11; 2022; July 26, 2022
Accepted: August 1, 2022
Published Online in Articles in Advance: January 5, 2023

https://doi.org/10.1287/inte.2022.1143

Abstract. This paper describes how Amazon redesigned the robotic picking algorithm used in Amazon Robotics (AR) fulfillment centers (FCs) to enable humans and robots to work together effectively. In AR FCs, robotic drives fetch storage pods filled with inventory for associates to pick. The picking algorithm needs to decide which specific units of inventory on which pods should be picked to fulfill customer order shipments. We want to do so in a way that is most efficient and distance traveled by drives per unit picked is the key performance metric. This new algorithm reduced the distance traveled by drives per unit picked by 62% without negative operational impact and has since been implemented in all AR FCs. This improvement reduced the number of drives required in AR FCs by 31%, which amounted to half a billion dollars in savings. The redesigned algorithm enabled seamless collaboration between associates and robots, and its effectiveness in scaling up convinced Amazon to make AR FCs the standard for new FCs, allowing Amazon to reduce the storage footprint by about 29% compared with non-AR FCs.

History: This paper was refereed.

Keywords: e-commerce • logistics • warehouse order picking • optimization

Introduction

Since its founding in 1994, Amazon.com has grown from an online bookseller to a global retail company. As its customer base grew, inventory volume increased, and the network expanded, many teams at Amazon had to create solutions that would improve the overall customer experience and benefits of shopping with Amazon, such as Prime and one-day shipping.

Fulfillment centers (FCs) serve as one of the most critical elements in the supply chain as the FC is where inventory is stored and customer order shipments are picked, packed, and shipped. Quickly outgrowing its first FC (Jeff Bezos's garage), Amazon had to expand to larger buildings where more inventory could be stored to meet the growing customer base. In parallel, the e-commerce industry was booming, which put more pressure on Amazon to implement innovative solutions within its FCs in order to scale and continue to meet the demands of the global network.

Traditionally, units of inventory are stored at fixed locations on shelves. Picking and stowing both require humans to walk to the shelves to retrieve or store inventory. Therefore, stowing and picking operations are usually labor-intensive as both stowers and pickers spend a lot of their time traveling from one location to another in the FC. Labor costs associated with order picking are a large fraction of total FC operating costs. One of the most impactful changes for our FCs was the 2012 acquisition of Kiva Systems, now known as Amazon Robotics (AR), which led to the introduction of robotics in FCs. But this was not simply a case of installing exciting new technology and reaping the benefits. This paper tells the story of the crucial role that operations research (OR) played in enabling the success of the AR robotic picking system. More specifically, we focus on how a redesigned algorithm for robotic picking enabled robots and humans to work together efficiently at scale in Amazon FCs.

The rest of the paper is structured as follows. We describe the robotic picking system and how Amazon recognized an opportunity for improving the picking algorithm used in the system before giving a short review of related work. We next describe the robotic picking algorithm: the context in which it must operate, the legacy algorithm, and the redesigned algorithm. We end with sections describing the implementation of the redesigned algorithm, the benefits of the implementation, and concluding remarks.

Robotic Picking

The robotic picking system at Amazon enables a semiautomated storage system in which robots and teams of associates work alongside one another to efficiently fulfill customer orders. Rather than our associates going to the products' shelves to pick for a customer order shipment or stow new inventory, robots bring shelves of inventory to our associates who are at workstations either picking or stowing items.

More specifically, the inventory received by an FC is stowed on mobile storage pods. Each pod carries a mixture of items, and the inventory of each item is spread over multiple pods. The pods are mobile in that a pod can be lifted and transported by a robotic drive (Figure 1). These pods are stored within a grid storage zone that has stationary workstations for picking and stowing on its boundary (Figure 2), and each of these stations is staffed with an associate to perform these operations (Figure 3). A single FC may have more than 10 storage zones.

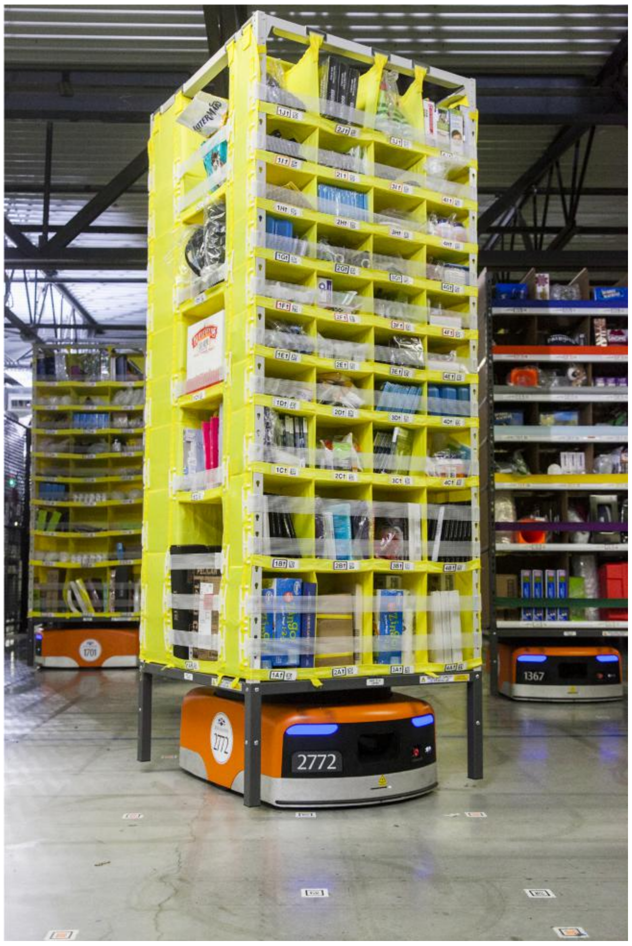
To pick an item, a drive must first travel to the pod that contains the required item that will be picked. The drive then carries the pod to a pick station. When the pod reaches the associate at the station, the associate picks the item and sends the item on its way to downstream operations. The drive then returns the pod to an open storage cell in the storage zone, which is usually different from its prior storage location. The operations to stow an item on a pod are similar. A drive first needs to travel to the pod that has been selected for stowing. The drive then carries the pod to the stow station, where the associate finds an open space in the pod and stows the item. The drive then travels with the pod back to the storage zone and leaves the pod in an open storage cell. Note that, typically, an associate picks several items from a single pod or stows several items to a single pod.

This new system brings two new advantages to our FCs. First, by bringing inventory to associates, we reduce time spent walking from one shelf to another and, thus, increase associate picking efficiency, and associates can even redirect that time to more value-added tasks, such as problem detection and IT management. Second, the new design of the storage system allows us to stow 40% more inventory in our FCs (Amazon 2020) because of the compact nature of how we fill and store our shelving pods. This robotic system became extremely efficient and allowed us to move items through our FCs more quickly, helping speed up delivery times for our customers.

Challenges and Opportunity for Improvement

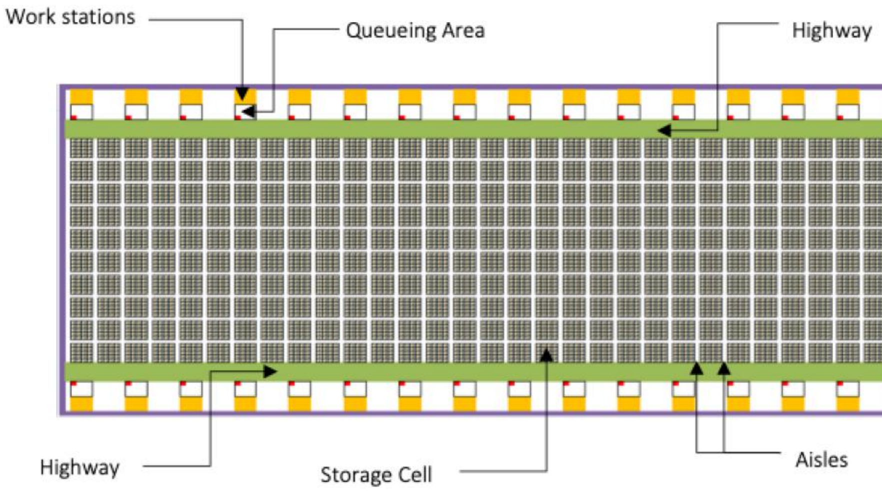
None of this happened overnight. In order for the new technology to work in Amazon FCs, teams first had to customize the system and software in order to scale to the needs of the buildings and Amazon's growing network. Multiple teams across Amazon worked on this, and by 2014, Amazon's first robotic FC was up and running. Initial results were encouraging: overall system efficiency was higher than at nonrobotic FCs, and associates appreciated the reduction in the amount of walking required under the new process. However, we saw an opportunity to improve the returns on such a large-scale change and sought to make the adjustments necessary to deliver better results on a global scale.

Figure 1. (Color online) A Robotic Drive Moves Under a Mobile Storage Pod (Essentially a Shelf Holding Units of Inventory), Lifts the Pod off the Floor, and Transports It



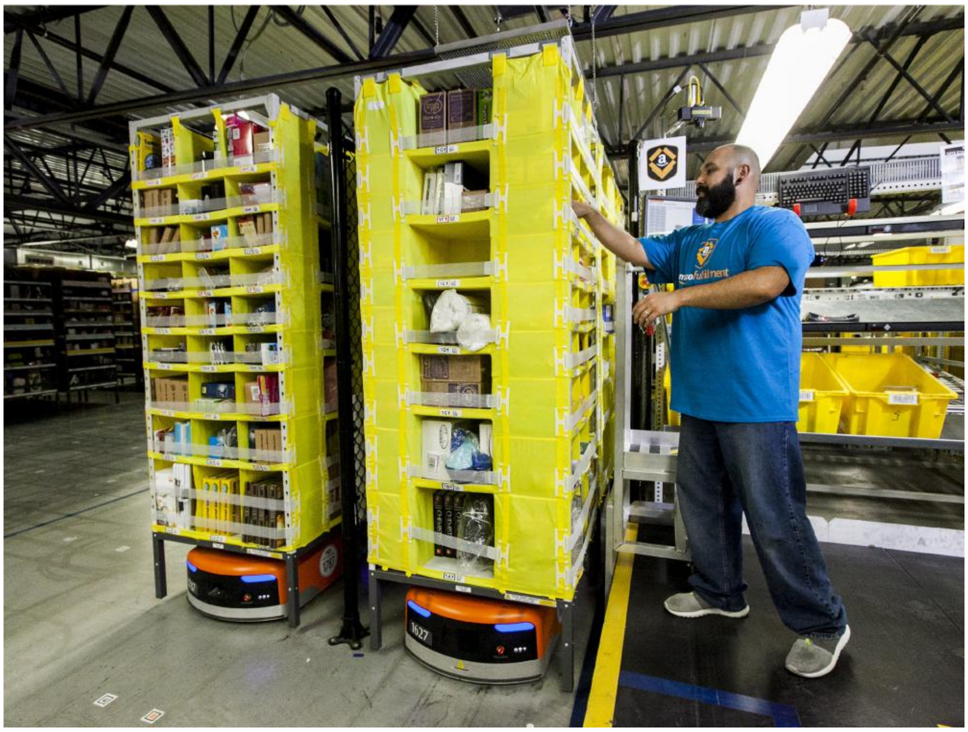
The work involved in customizing the system included redesigning pods and drives and modifying algorithms used in the operation of the system. The original picking algorithm that determined which items to pick from which pods was focused on one order at a time and picking all items for that order. However, an important feature of Amazon picking operations is that because of the large number of multi-item order shipments to fulfill, FCs pick each item of a multi-item shipment separately and aggregate them downstream using a system called the Amazon

Figure 2. (Color online) Robotic Drives Travel down Aisles and Across Highways to Move Pods to and from Workstations Within a Grid Storage Zone



Fulfillment Engine: a wall of shelves, where each partially picked order occupies a shelf until completely picked (we refer to this system as the order-aggregation wall in the rest of the paper or, more simply, the wall). In particular, this allows an associate to pick for multiple order shipments at the same time from the same pod and for the units from the same order shipment to be picked by different associates at different times. This is part of the design in both the legacy manual and new robotic picking processes and differs from the fulfillment systems for which the robotic picking system was originally designed. The picking algorithm, therefore, had to be revised to allow for this mode of operations. Another difference was that the enormous size of Amazon FCs meant that the pods and drives had to be divided into physically separate zones, so it was necessary for the algorithm to balance the amount of work across the zones. A final point to note is that Amazon's random-stow policy results in inventory for the same product being stowed among multiple different pods to allow for picking flexibility. This policy differs

Figure 3. (Color online) An Associate Picks Inventory from a Mobile Storage Pod



from other fulfillment systems for which the robotic picking system was designed. Therefore, it was necessary to consider whether the legacy picking algorithm would still work well under such a stowing policy.

Aside from all this work to adapt the algorithm for the Amazon context, more work was necessary to improve the AR FC pick efficiency. A critical performance metric for the robotic picking system is the distance traveled by drives per pick and/or stow operation; this metric (denoted by drive distance per pick) determines how many drives are needed in an FC for a given level of volume throughput. Drive distance per pick can be improved by increasing the number of picked units per pod trip (denoted by "pile-on") and by reducing the distance traveled by drives per pod trip. There are three operational policies relevant to the drive travel distance:

- Stowing policy: decides which pods stow received inventory.
- Picking policy: decides from which pods to pick inventory to satisfy orders.
- Storage policy: decides where to store pods upon completion of a stow or pick event.

The modeling and optimization team at Amazon has conducted research to develop improved algorithms for each of these decision policies. In this paper, we focus on the algorithm for the picking policy as this has been implemented with significant impact on improving the throughput and efficiency in AR FCs. Work on improving the stowing and storage policies is previously described by Yuan et al. (2018, 2019) and Cezik et al. (2021).

In 2013, our team conducted a preliminary study to understand how much pile-on (the number of picked units per pod trip as defined previously) could potentially be improved. This study looked at a simplified model (relaxing several real-world constraints) determining from what pods to pick to satisfy a set of orders in order to maximize pile-on over a period of time given full knowledge of all arrivals to the backlog over this time period as well as an online version of the problem in which pick decisions could not use information about future arrivals to the backlog. Both models were solved using mixed-integer programming (MIP). The study concluded that there was a significant opportunity to improve pile-on and it seemed likely that we could capture a significant portion of that opportunity. This motivated further research to redesign the picking policies used in the robotic picking system.

Related Work

We review some of the relevant literature. Wurman et al. (2008) and Enright and Wurman (2011) present a good overview of the setting and relevant research problems for the type of robotic system discussed in this paper. In addition, there are several relevant surveys of the literature: De Koster et al. (2007) look at order picking in warehouses in general, whereas Roodbergen and Vis (2009), Gagliardi et al. (2012), and Azadeh et al. (2019) focus on automated systems in warehouses. Boysen et al. (2019), in particular, survey literature on warehousing systems in the context of e-commerce retailers.

Several papers consider operational problems in the domain of managing mobile robot fulfillment systems that complement the order-picking problem considered in this paper. Lamballais et al. (2017) explores how the layout of the pod storage area impacts performance. Stowing and storage policies are considered by Weidinger et al. (2018), Lamballais et al. (2019), Yuan et al. (2018, 2019), and Cezik et al. (2021). Routing the robotic drives is a well-researched problem discussed in surveys on the routing of automated guided vehicles in various contexts, such as Qiu et al. (2002) and Vis (2006), whereas Gharehgozli and Zaerpour (2020) wrote a more recent paper looking at routing in the specific context of order picking for e-commerce retailers.

In the context of order picking, Zou et al. (2017) utilize a queuing network model to analyze the picking throughput rates of various policies for allocating drives to pick stations but do not consider any properties of the orders, such as deadlines or order consolidation. Rimélé et al. (2021) model the order-picking process as a stochastic dynamic program but require orders to be assigned for picking immediately, whereas in practice, there is usually a backlog of orders to be fulfilled. Merschformann et al. (2019) propose a discrete event simulation to model and evaluate various policies for order assignment, pod selection, and pod storage assignment. Boysen et al. (2017) formulate an integer program to sequence the orders and pods assigned to a single pick station to minimize the number of pod visits. Valle and Beasley (2021) formulate an integer program to assign pods and orders to pick stations, but they assume that items in the same order must be picked at the same pick station and focus on the static rather than the dynamic problem. Wang et al. (2021) model the problem of scheduling the assignments of pods and orders to pick stations with schedule-induced fluctuations of human pickers' picking speed as a stochastic dynamic program, but they still restrict orders to be assigned to the same pick station, and their solution takes on the order of two minutes to solve even for moderately sized problems that would be considered small problems in the Amazon context. Indeed, this is representative of the literature in general, in which the scale of Amazon's operations tends to dwarf the size of problems considered large in the literature.

As emphasized by van Gils et al. (2018), there is a need to go beyond isolated problems and consider practical and holistic models. To the best of our knowledge, the only other work that describes a successfully implemented algorithm in the context of robotic picking systems for e-commerce warehousing operations is the article by Qin et al. (2022), which matches drives to pods to pickers, and the items each picker has to pick is taken as an input. In contrast, an important decision in the problem considered in this paper is which items from the backlog to pick next, which adds to the difficulty of the problem but also increases the potential for reducing the total distances traveled by the drives by allowing a greater opportunity for increasing the pile-on.

Robotic Picking Algorithm

This section describes the context and requirements of both the legacy and redesigned robotic picking algorithms. When a customer places an order with Amazon, this order is converted to one or more order shipments, each of which gets assigned to an FC and a fulfillment path. This decision is made upstream of the picking process and, therefore, outside the scope of this paper. Subsequently, for each such order shipment, this assignment translates into a due time for picking at the assigned FC. At an FC, the assigned order shipments that have not yet been picked compose the pick backlog. From this backlog, the picking algorithm has to select a set of shipments eligible for picking to form the pick window—the set of shipments that should be scheduled to be picked—and then decides a pick schedule for this pick window. This pick schedule entails the set of pods to retrieve, the items to pick from each pod (and the corresponding shipment each picked item fulfills), and the pick station to which each pod is to be sent for picking. The size of the pick window is measured in terms of number of minutes of work (how long it takes for the entire pick window to be picked), typically on the order of tens of minutes. This schedule is handed over to the AR material handling system for execution but is executed for only the first minute, after which the pick algorithm is run again with updated information, refreshing the pick schedule. Thus, items in a pick schedule that have not yet been picked at the time of the refresh may have their assignments reconsidered. Because customer orders can be placed at any time of the day, the backlog is continuously updated, and therefore, the pick algorithm must execute more than a thousand times a day and provide pick schedules without advance knowledge of future customer order shipments entering the backlog. These processes are illustrated in a flowchart in Figure 4.

The robotic picking algorithm also needs to account for the following requirements of the order fulfillment system:

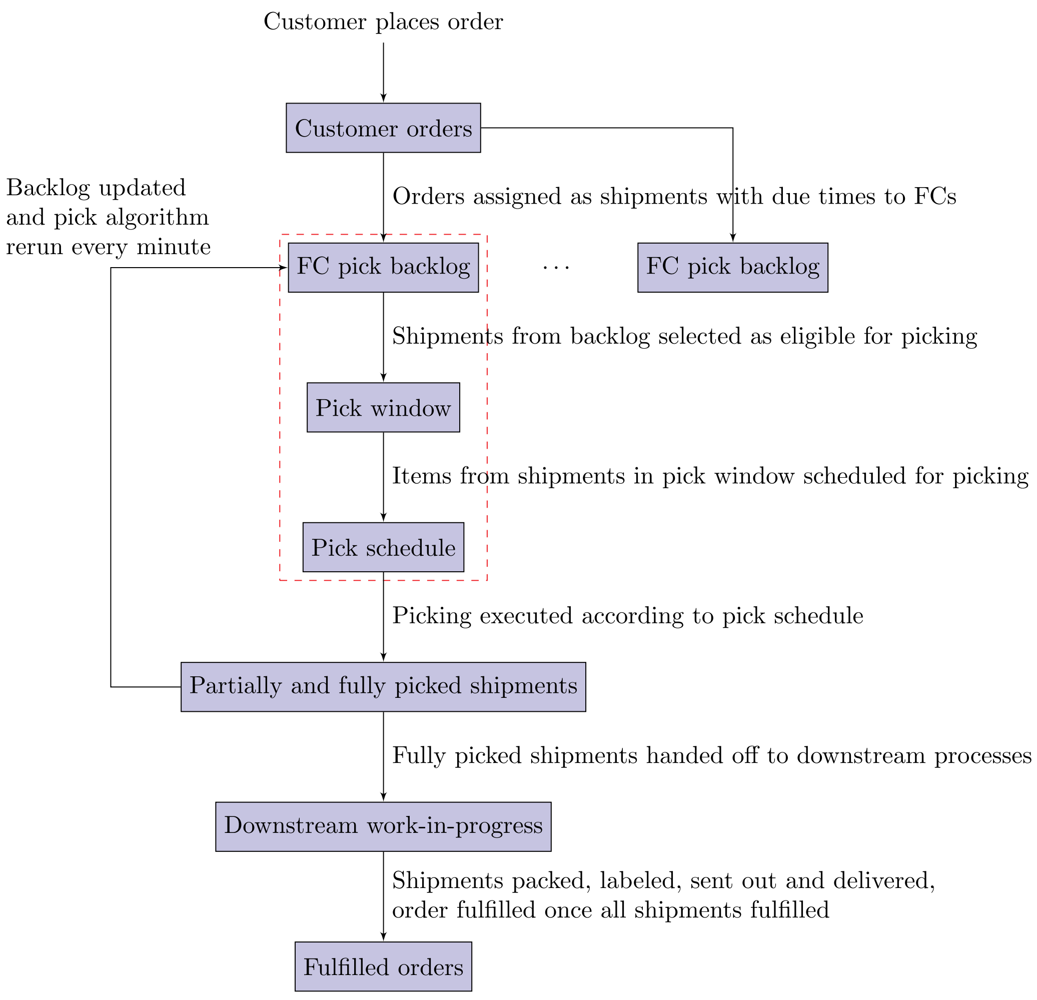
- Satisfying target rates defined for each process path: Each order shipment in the pick backlog is categorized by a process path, which encodes the sequence of downstream processes through which the order shipment must go before departing the FC. In order to ensure that the flow of order shipments to downstream processes is operationally feasible (adhering to labor, mechanical, and other constraints), each process path has a target rate (determined by an upstream system)—a rate of picking units per unit time on that process path that we want to get as close to as possible without exceeding.
- Tracking to cycle-time targets defined for multi-item order shipments: Each multi-item order shipment occupies its own shelf on the order-aggregation wall when partially picked. Hence, this puts an upper limit on how many partially picked orders there can be at any point in time. If this limit is exceeded, "gridlock" occurs, in which the wall runs out of space for items arriving for orders above the limit and becomes a bottleneck delaying the upstream picking operations (Gallien and Weber 2010). The amount of time each order is left partially picked is, therefore, a critical measure affecting how many orders for which we can pick yet remaining below this wall capacity. For any order shipment, the time between picking the first and last item (i.e., its cycle time) is a reasonable proxy for the amount of time that this order shipment takes up capacity in the wall. Keeping to the cycle-time targets helps prevent overwhelming the wall, avoiding gridlock. Addressing the cycle-time targets also necessitates that all picks for multi-item shipments need to be included in the same pick window (except for a small percentage to be included in the next pick window).
- Remaining feasible for storage zone–level picker allocations and not leaving workers idle: Because the picking algorithm operates at an FC level with multiple zones, it needs to ensure that each zone is allocated picks according to its pick throughput capacity. If a zone is allocated fewer picks than appropriate for the number of active picking stations, the associates picking in that zone are likely to face a shortage of work, leading to unnecessary idle time.
- Satisfying due times for order shipments: Each order shipment in the pick backlog has a due time. In order to satisfy these due times, pick throughput needs to be allocated to each of these due-time groups (in other words, priority groups) appropriately. These allocations are determined by an upstream system.

At a high level, the goal of the robotic picking algorithm is to schedule the picks so as to minimize the drive distance per pick and maintain a low drive distance per pick over time, subject to the preceding constraints.

Redesigning the Robotic Picking Algorithm

This section describes how our team redesigned the robotic picking algorithm in 2015 so that the drives would bring items to associates more efficiently. We first describe the legacy algorithm and the scope for improvement, then the redesigned algorithm, and finally we discuss

Figure 4. (Color online) The Robotic Picking Algorithm Described in This Paper Is Responsible for the Processes in the Dashed Box in the Context of All the Processes Taking Place from the Time a Customer Places an Order to the Time the Customer Receives the Order


Customer places order
Customer orders
Orders assigned as shipments with due times to FCs
Backlog updated and pick algorithm rerun every minute
FC pick backlog
· · ·
FC pick backlog
Shipments from backlog selected as eligible for picking
Pick window
Items from shipments in pick window scheduled for picking
Pick schedule
Picking executed according to pick schedule
Partially and fully picked shipments
Fully picked shipments handed off to downstream processes
Downstream work-in-progress
Shipments packed, labeled, sent out and delivered, order fulfilled once all shipments fulfilled
Fulfilled orders

how we address an important trade-off in the context of the redesigned algorithm.

Legacy Algorithm

The legacy algorithm selected items for the pick window according to a strict priority rule on due times. After the pick window was created, picks from the pick window were then greedily allocated to pick stations and pods based on a drive distance per pick estimate for each pod. Though the legacy algorithm was very simple and fast, there were two major opportunities for improvement. First, the structure of this algorithm involved decomposing the problem into two phases: a first phase of selecting a pick window without considering pick efficiency and then only considering pick efficiency in the second phase of scheduling picks from pods. This meant that the legacy algorithm was limited in the pick efficiency it could achieve by the output of the first phase. An algorithm that considered the pick-window selection and pick-scheduling problem jointly could potentially attain a much higher pick efficiency. Second, there was no guard against excessive greediness (or "cherry picking") when scheduling picks, which meant that pick efficiency could start high but degrade significantly over time until the backlog was sufficiently replenished. Because there is a finite number of drives in each FC, if pick efficiency drops below a certain threshold, we are likely to end up with idle pickers and be unable to sustain the maximum pick throughput rate. This left an opportunity to improve the stability of the pick efficiency over time.

Redesigned Algorithm

The redesigned picking algorithm, therefore, aims to jointly optimize the selection of shipments into the pick window and the assignment of picks to pods in order to minimize drive distance per pick. This allows the algorithm to take advantage of the entire pick backlog when optimizing for the assignment of picks to pods. Unfortunately, including the assignment of pods to pick stations is challenging because directly optimizing the distance traveled by drives per pick requires solving a large combinatorial optimization problem with a nonlinear objective, in which the drive distance per pick would be a ratio of two variables, total drive distance and number of picks. Therefore, we decompose the problem by first maximizing pile-on (equivalently, minimizing pod trips required per pick) subject to the requirements of the order fulfillment system (the "picking module"), then minimizing the distance traveled by drives per pod trip (the "assignment module"). In this decomposition, the picking module is responsible for determining which subset of the pick backlog composes the pick window and which specific units of inventory on which specific pods are used to fulfill the pick window, whereas the assignment module is responsible for assigning the specified pods to pick stations. This removes the nonlinear aspect of the problem, but the picking module still needs to solve a large problem (the full problem can be written using on the order of 3.8 million variables, of which 0.5 million are binary with 7.6 million constraints) and is solved using a four-phase decomposition heuristic.

The picking module considers the order shipments to be picked and their due times, inventory, available labor, throughput, and cycle-time targets and determines which order shipments to be picked next and from which pod each item in those order shipments should be picked. This is a complex problem requiring decisions ranging from higher level ones, such as the number of picks by process path and storage zone, to lower level ones, such as picking a particular item from a particular pod for a particular order shipment. As mentioned, this problem is too large to solve directly. Therefore, we decomposed the problem into phases with each successive phase making increasingly granular decisions.

This module proceeds in four phases: feasibility, pod selection, provisional schedule creation, and then final schedule creation. The solution of each phase is essentially used to restrict the feasible space for the subsequent phases. Figure 5 illustrates how each phase relates to the others. Each phase is solved with an appropriate mathematical program. We now summarize these phases, referring the reader to Appendix A for further details.

1. In the feasibility phase, we determine a feasible number of picks by process path and storage zone and do so by selecting a suitable subset of available shipments. In this phase, the picks in each process path are restricted to those with due times not later than a due-time threshold. This due-time threshold is initialized from values taken from an upstream system. In each iteration of this phase, any process path with picks lower than its target rate has its threshold adjusted to a later due time (if there exist orders with a later due time in that process path) in the next iteration. At each iteration, a mixed-integer quadratic program (MIQP) determines the number of picks that will minimize the sum of squared shortages from the process path target rates and zone targets on a percentage basis to thereby minimize shortages in a manner that tends to balance them (proportionally) across the process paths and zones. The output of this phase is a reduced pick backlog that approximately corresponds to the shipments that are planned to be picked over the next day or so.

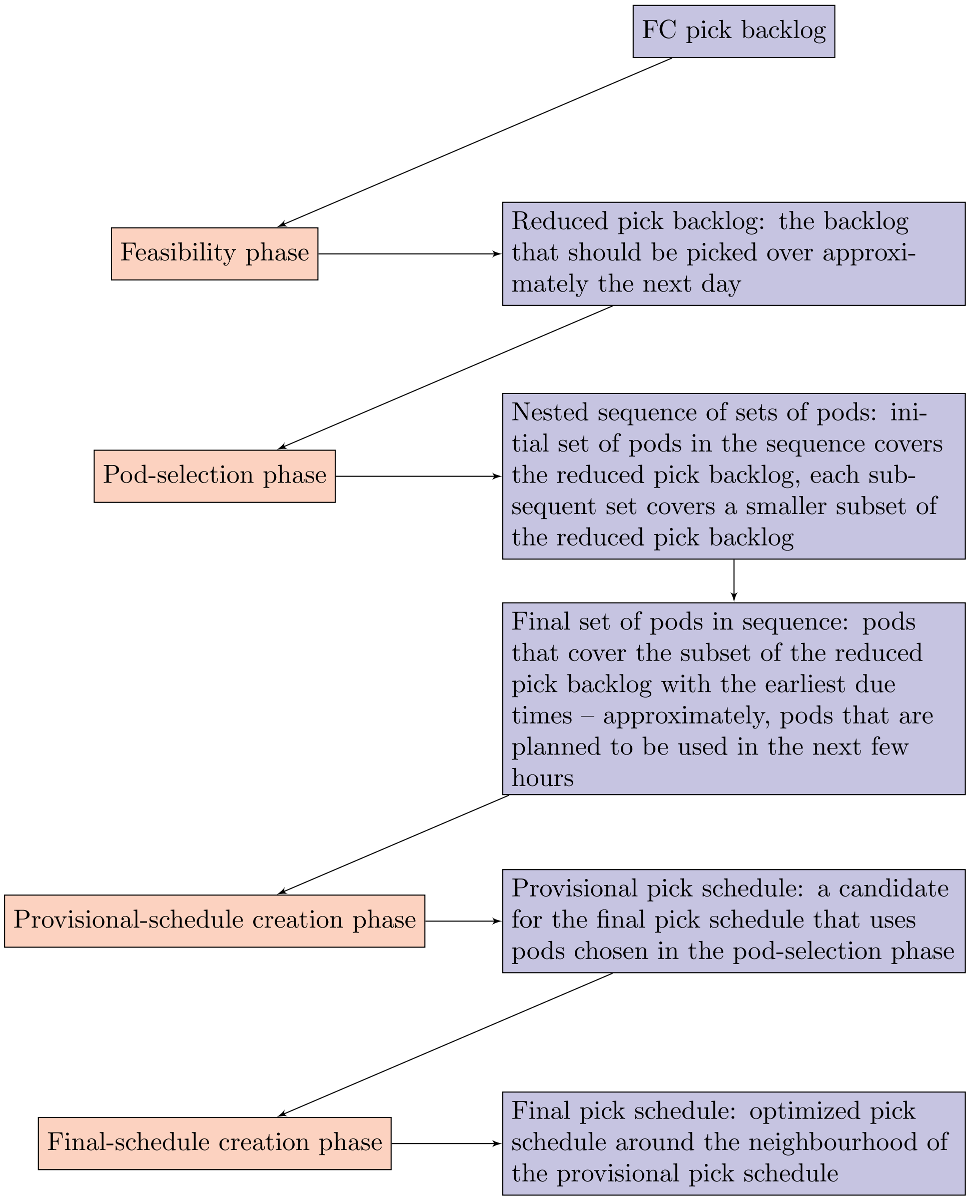
2. In the pod-selection phase, we solve a sequence of MIPs, each generating a set of pods to cover the items that need to be picked up to a certain due time, approximately satisfying the feasible output of the feasibility phase. The primary goal is to have a small subset of pods for subsequent phases, and a secondary goal is to respect the feasible zone utilizations of the previous phase. The output of this phase is a nested sequence of sets of pods: the initial pod set holds inventory required for all the shipments in the reduced pick backlog, and the final pod set holds inventory required for all the shipments in the subset of the reduced pick backlog with the earliest due times, and each set of pods is a subset of the previous pod set. Thus, the final set of pods in the sequence corresponds to the pods planned to be assigned over the next few hours or so.

3. In the provisional schedule creation phase, we use a greedy algorithm to select and schedule order shipments to satisfy overall pick requirements (as determined by the previous two phases), matching them to the final set of pods in the nested sequence of pod sets from the pod-selection phase. The goal is to limit the number of times that a matched pod is used again in a future matching and avoid a myopic decision at the expense of future pick efficiency. The output of this phase is a candidate pick schedule: the shipments that are planned to be picked in the next 15 minutes or so (making up the candidate pick window), the specific units of inventory that fulfill these shipments, and the pods holding these units.

4. In the final schedule creation phase, we search in the neighborhood of the candidate pick schedule from the previous phase by augmenting the selected pods with a set of pods that makes it feasible to completely pick any partially scheduled shipment, thus enabling a different choice of subset of shipments to be completely versus partially picked. We solve an MIP to determine a final output of order shipments to be picked next and from which pod each item in those order shipments should be picked. This returns the final pick schedule (along with the corresponding final pick window).

The proposed assignment module then takes these picks and assigns them to stations. This module has two goals: guarantee that each station does not experience idle time and, subject to that, minimize the distance the pods

Figure 5. (Color online) The Four Phases of the Picking Algorithm Successively Reduce the Search Space of the Problem


FC pick backlog
Feasibility phase
Reduced pick backlog: the backlog that should be picked over approximately the next day
Pod-selection phase
Nested sequence of sets of pods: initial set of pods in the sequence covers the reduced pick backlog, each subsequent set covers a smaller subset of the reduced pick backlog
Final set of pods in sequence: pods that cover the subset of the reduced pick backlog with the earliest due times – approximately, pods that are planned to be used in the next few hours
Provisional-schedule creation phase
Provisional pick schedule: a candidate for the final pick schedule that uses pods chosen in the pod-selection phase
Final-schedule creation phase
Final pick schedule: optimized pick schedule around the neighbourhood of the provisional pick schedule

chosen by the picking module have to travel. To that end, the assignment problem solves an MIP to minimize the distance traveled by the pods and give each station at least a fixed amount of work. When a pick station's queue of incoming work runs low, the assignment module assigns pods (with the associated picks from the pick schedule) to stations, and the assignment module triggers a rerun of the picking module to schedule more picks when the number of picks it has to work with is low.

Trade-off Between Multi-item Order Shipment Cycle Time and Pile-on

Two important parameters that affect the performance of our algorithm are the overall size of the pick window (here, size is measured in terms of minutes of work, that is, how long it takes for the pick window to be picked) and how frequently it is updated (how much of the pick window is picked before it is refreshed). The picking module takes the pick-window size as an input parameter and returns the pick window as part of the output of phase 4 as often as required by the update frequency. Thus, these parameters are not optimized by the picking module, but both parameters influence the overall throughput of multi-item order shipments. Items for such shipments are not necessarily picked all at the same time or by the same picker, and partially picked shipments occupy space on the order-aggregation wall. If the multi-item order shipment cycle time is high, shipments take longer to get completed, and we run a higher risk of hitting capacity on the wall, causing delays. Selecting a suitable pick-window size and update frequency offers a degree of control over cycle time. In this section, we discuss the trade-off between cycle time and pick efficiency.

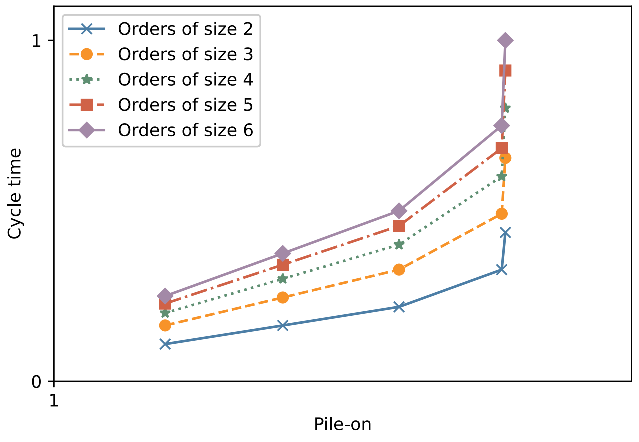
Intuitively, there is a trade-off between these two measures: the more freedom we give to the picking algorithm, the more efficient we can be with our drives (and the higher pile-on we can achieve) but the less control we have over the cycle times of multi-item order shipments. We propose models that approximate these measures given some high-level parameters about the system. Both models are derived by making simplifying assumptions in a probabilistic framework. In the cycle-time model, given the size of the pick window and the frequency with which it is updated, the model outputs an estimate of the cycle time for a multi-item order shipment containing any specified number of items. This estimate is given in the form of a closed-form expression. In the pile-on model, given the size and update frequency of the pick-window and the eligible backlog size, the model outputs an estimate of the pile-on that can be attained. This estimate is derived via an iterative method. Appendix B contains more details on both models.

Based on these models, we observe that increasing pick-window size generally increases the pile-on. When the pick-window size is very small relative to backlog, we can cherry-pick the shipments from a small number of pods that contain the most items from the backlog, resulting in a relatively high pile-on. As we increase the pick-window size, the opportunity to cherry-pick becomes more limited because more pods need to be selected. As the pick-window size continues to increase, we are allowed a greater degree of freedom that increases the average number of items from the backlog in a typical pod, which provides a more sustainable improvement of pile-on. In most situations, the pick-window size is in the region where increasing pick-window size would increase pile-on. On the other hand, increasing the pick-window size increases cycle time because, once the first unit of an order shipment has been picked, it is less likely for the other units of that order shipment to be picked when there is more freedom to select other units.

Therefore, in general, we face a trade-off between pile-on and cycle time, and we want to understand how the trade-off behaves quantitatively in order to be most effective operationally. Figure 6 shows an example of the trade-off that can be achieved.

Based on this analysis, a suitable pick-window size and update frequency were selected as the standard parameters with which to run the algorithm. This also potentially allows us to use pick-window size as a lever to control the trade-off according to the situation. For example, during peak seasons, we are more likely to set a larger pick-window size to attain a high pile-on in order to sustain a high pick rate and accept larger cycle times. On the other hand, during off-peak seasons, a lower pile-on might be sufficient to sustain the required pick rate, and we can set a smaller pick-window size to reduce cycle time and speed up order completion time.

Figure 6. (Color online) There Is a Trade-off Between the Cycle Time of Multi-item Order Shipments and Pile-on as Shown in This Example Graph (Cycle Times Are Normalized; Pile-on Numbers Are Shown Relative to One but Without Scale)



Implementation

A major challenge in implementing this algorithm was that it had to enter into production in existing, operational AR FCs, and any issues, whether bugs in the software code or unexpectedly poor performance of the algorithm, would immediately bottleneck the operations of those FCs, possibly leaving them unable to clear their backlogs and impacting their performance for days after. Those managing these FCs would understandably be concerned about their ability to fulfill customer demand should anything go awry. As with the implementation of any new system, there were risks that things could go wrong, and the stakes were high. Therefore, we had to do all we could to identify and eliminate any potential issues prior to rolling out the algorithm to production and earn the confidence and buy-in from those managing picking operations on the ground, who would be most impacted by the change.

We first designed and prototyped the new AR picking algorithm within a simulation environment and conducted extensive testing. This was completed in 2015. Then, we ported this implementation to production with the help of our technology teams. The picking technology team implemented a flexible architecture for the AR pick scheduling space, breaking down the problem into multiple microservices, enabling separation of responsibilities and parallel development. A team of specialists across multiple disciplines, including engineering and research, tested the software at a handful of

FCs in 2016. The team took on a goal of running incremental pilots at the end of two-week sprints, allowing early and parallel discovery of any production issues, incrementally gaining confidence in the ability of the new algorithm to meet the business objective of picking the right items within their due times and also optimizing for picking efficiency.

Because our objective was to provide incremental validated progress for our business, we only activated the picking module at first. The largest subproblem (corresponding to the final schedule creation phase described in Appendix A), which is solved multiple times during this phase, has on the order of 50,000 variables, most of which are binary, and 500,000 constraints for a typical AR FC. So we initially focused on satisfying the run-time requirement for creating pick schedules, which is on the order of one minute. By 2017, the teams were confident that the implemented code was ready for production use.

Subsequently, we deployed the production implementation of the picking module in several AR FCs to collect data on its performance. To this end, for several months, we monitored the key performance metrics and the metrics related to system requirements for order fulfillment. In this phase, we validated our experimental results obtained during the design of the algorithm.

After long discussions with the business and technology teams responsible for warehouse operations and software, we also came to an agreement to exclude the assignment module from the production implementation and continue using the legacy assignment algorithm. Technology teams argued that the picking module already allowed us to capture a significant benefit and would still work well with the legacy assignment algorithm, avoiding the developmental risks of overhauling both systems at the same time. Thus, only the picking module was implemented though this decision may be revisited in the future if the relative benefits and costs of implementing the assignment module change.

The models built for quantifying the trade-off between cycle time and pile-on were used to help understand what a good pick-window size should be. Business teams chose a pick-window size that corresponded to a suitable point on the Pareto curve with expected cycle time and pile-on numbers that were in line with business requirements. Currently, pick-window size is set as a static parameter, unchanged from its initial setting, but there is the potential to dynamically adjust it.

In production, the picking algorithm runs in the background without user intervention approximately every minute. The scope of our production implementation has been incrementally widened, and it has been operational in all (more than 50) Amazon AR FCs in North America since 2017.

From the successful implementation of the redesigned picking module, a major lesson learned is the tremendous value unlocked from having multiple teams working together. The redesigned algorithm worked well, but merely coming up with a new algorithm would not have been enough to achieve the impact this work eventually had. It took the combined effort of multiple teams, including but not limited to the research team designing the algorithm, the software development team planning and coding the incremental pilots, and the business team making the case to invest resources into the project, for the algorithm to make it to production and achieve the efficiency gains projected from computational experiments.

Benefits

The deployment of the redesigned robotic picking algorithm reduced the drive distance per pick by 62% in production compared with the legacy algorithm, having no negative impact on operational efficiency. This improvement reduced the number of drives required in AR FCs by 31%, which, in turn, amounted to half a billion dollars in direct savings from the algorithm over the period between 2017, when the algorithm was first implemented in production, and 2020, when the savings were evaluated. This savings figure was evaluated purely based on the cost savings from the drives, and the annual savings from the algorithm are projected to increase in scale as the number of FCs grows. Moreover, the reduction in drive distance resulted in a corresponding 31% reduction in drive energy usage though neither the financial nor environmental impact of this has been quantified.

The success of the use of robots in AR FCs at Amazon is now celebrated, but at the time of Amazon's acquisition of Kiva, it was not yet clear that the robotic picking system would work sufficiently well at the scale and context required in Amazon to justify the capital investment involved in setting up each AR FC. A huge indirect contribution of this algorithmic work was to convince Amazon that robotic picking could scale well to our FCs. This work helped drive Amazon toward making AR FCs the standard for new FCs, storing, as mentioned earlier, 40% more inventory than non-AR FCs (Amazon 2020). Equivalently, this means building AR FCs required 29% (because $1 - \frac{1}{1.4} = 0.29$) less area allocated to storing inventory, compared with building non-AR FCs that were able to hold the same amount of inventory.

More broadly, these results contribute toward a vision of seamless collaboration between associates and robots and serve as evidence of the value of robots to everyone throughout Amazon, ensuring the success of the 2012 Kiva acquisition and encouraging further investment in (and adoption of) robotics. Since the introduction of robots, Amazon has added more than 300,000 full-time jobs globally (Amazon 2018), including jobs necessitated by the use of robots, such as positions in IT, technicians to service and maintain the robots, and even new career paths such as flow control specialists (Amazon 2019). Furthermore, the AR FCs often employ more people than non-AR FCs because of the higher volume of inventory managed in these buildings as well as the variety of jobs supporting these robotic systems.

Conclusions

This work is a clear illustration of the synergy between technology and OR. The new technology—robotic drives and pods, together with the software system that controlled the system robots—can improve the efficiency of FCs but only up to a certain point. In order to sustain improvements in operational efficiency as the operations scaled up, OR was necessary to model and optimize for key performance metrics using advanced algorithms so that the system was tailored to the strengths of the new technology. The robotic picking system was able to perform closer to its full potential when paired with algorithms designed using OR techniques to effectively utilize its strengths. OR was used not only in the picking algorithm, but also in the preliminary study done to reveal the opportunity of improvement and ability to capture it. This gave Amazon the confidence to devote more resources to redesigning the picking algorithm. In sum, by enabling new technology to scale effectively, OR has played a key role in ushering in a new era of fulfillment technology, enabling humans and robots to work together efficiently.

Acknowledgments

We acknowledge the leadership, technology, and business teams from Amazon involved in this work without which the research would not have had the impact it has had. We also thank Editor-in-Chief Michael F. Gorman as well as the anonymous associate editor and reviewers for their extremely helpful suggestions to significantly improve the paper during the review process. We are grateful for the support and encouragement of John Milne and Steve Graves, who provided feedback on earlier versions of the paper.

Appendix A. Details of the Picking Module

In this appendix, we describe the picking module in more technical detail. We begin by describing a formulation of the problem that we ideally want to solve. Because this formulation turns out to be intractable at the required scale, we then describe the decomposition of the problem into phases.

The goal of the picking module is to select the pick schedule that maximizes the overall pile-on and keeps the pile-on consistently high in each time period (say, within 80% of the overall pile-on), subject to the various operational constraints. We start with some notation:

- K: set of SKUs
- Z: set of zones
- T: set of time periods in time horizon
- P_z: set of pods in zone z with $P = \cup_{z \in Z} P_z$
- I: set of process paths
- D_i: set of due times for process path i up to a suitable due time $\bar{d}_i$
- J_i: set of shipments in process path i with $J = \cup_{i \in I} J_i$
- $J_i(d)$: set of shipments in J_i with due time at most d
- a_{pk0}: initial units of inventory for SKU k on pod p
- b_{jk0}: initial units of SKU k for shipment j in backlog
- c_{zt}: available zone capacity in zone z in period t
- $[\underline{f}_{it}, \bar{f}_{it}]$: allowed target rate range for process path i in period t with nominal pick rate f_{it}
- $\underline{h}_{idt}$: pick throughput allocations for process path i and due time d in period t
- g_{it}: partial picking allowance for process path i in period T
- $\bar{d}_i$: latest due time for process path i

We can assume that most, if not all, of the backlog has to be picked by the end of the time horizon (approximately a day or so). Therefore, in place of maximizing pile-on, we may reasonably minimize the number of times a pod is selected in a period. Similarly, we may constrain the number of pods selected in each time period to be at least 80% of the average number of pods selected in each time period, weighted by $\sum_{i \in I} f_{it}$. The variables in the model are as follows:

- x_{pt}: binary variable indicating whether pod p is selected in period t
- w_{jt}: binary variable indicating whether shipment j is assigned to the pick window in period t
- y_{pkt}: number of units of SKU k to be picked from pod p in period t
- s_{jkt}: number of units of SKU k not picked for shipment j by period t if j is selected into the pick window in period t (zero otherwise)
- a_{pkt}: units of inventory for SKU k on pod p in period t
- b_{jkt}: units of SKU k for shipment j in backlog in period t

The formulation can then be written as

$$\min \sum_{p \in P,\, t \in T} x_{pt} \tag{A.1}$$

subject to (A.2)

$$\sum_{p \in P} x_{pt} \geq 0.8 \frac{\sum_{i \in I} f_{it}}{\sum_{i \in I,\, \tau \in T} f_{i\tau}} \sum_{p \in P,\, \tau \in T} x_{p\tau} \qquad \forall t \in T, \tag{A.3}$$

$$\sum_{j \in J} b_{jkt} w_{jt} - \sum_{j \in J} s_{jkt} = \sum_{p \in P} y_{pkt} \qquad \forall k \in K, t \in T, \tag{A.4}$$

$$\sum_{k \in K} s_{jkt} \leq \left(\sum_{k \in K} b_{jkt} - 1 \right) w_{jt} \qquad \forall j \in J, t \in T, \tag{A.5}$$

$$\sum_{j \in J} s_{jkt} \leq \sum_{p \in P} a_{pkt} \qquad \forall k \in K, t \in T, \tag{A.6}$$

$$y_{pkt} \leq a_{pkt} x_{pt} \qquad \forall k \in K, p \in P, t \in T, \tag{A.7}$$

$$\sum_{p \in P_z,\, k \in K} y_{pkt} \leq c_{zt} \qquad \forall z \in Z, t \in T, \tag{A.8}$$

$$\sum_{j \in J_i} s_{jkt} \leq g_{it} \qquad \forall i \in I, t \in T, \tag{A.9}$$

$$\underline{f}_{it} \leq \sum_{k \in K,\, j \in J_i} \left(b_{jkt} w_{jt} - s_{jkt} \right) \leq \bar{f}_{it} \qquad \forall i \in I, t \in T, \tag{A.10}$$

$$\underline{h}_{idt} \leq \sum_{k \in K,\, j \in J_i(d)} \left(b_{jkt} w_{jt} - s_{jkt} \right) \qquad \forall i \in I, d \in D_i, t \in T, \tag{A.11}$$

$$a_{pkt} = a_{pk,t-1} - y_{pk,t-1} \qquad \forall k \in K, p \in P, t \in T, \text{with } y_{pk0} = 0, \tag{A.12}$$

$$b_{jkt} = b_{jk,t-1}(1 - w_{j,t-1}) + s_{jk,t-1} \quad \forall j \in J, k \in K, t \in T, \text{with } w_{j0} = s_{jk0} = 0, \tag{A.13}$$

$$0 \le y_{pkt} \le a_{pkt} \quad \forall k \in K, p \in P, t \in T, \tag{A.14}$$

$$0 \le s_{jkt} \le b_{jkt} \quad \forall k \in K, j \in J, t \in T, \tag{A.15}$$

$$w_{jt} \in \{0,1\} \quad \forall j \in J, t \in T, \tag{A.16}$$

$$x_{pt} \in \{0,1\} \quad \forall p \in P, t \in T. \tag{A.17}$$

Unfortunately, this formulation proves too large to be practical at the required scale because of the large number of pods and shipments that need to be considered. Therefore, we propose a decomposition of the problem that aims to solve the same problem over a reduced search space. As previously illustrated in Figure 5, each phase successively reduces the search space as follows: phase 1 restricts the pick backlog to that which should be picked over the next day or so; phase 2 structures the reduced backlog from phase 1 into nested subsets and selects nested subsets of pods to cover these nested backlogs; phase 3 considers the smallest pod set from phase 2 (approximately corresponding to pods that are planned to be used in the next few hours) and selects shipments that are fulfilled by units picked from these pods, producing a provisional pick schedule for the next 15 minutes or so; finally, phase 4 uses the provisional pick schedule from phase 3 to constrain the search space as it solves a modification of the original problem.

We are now ready to discuss the phases in more detail. Phase 1, the feasibility phase, considers the backlog and the desired number of picks in order to determine a suitable restriction of the backlog that still allows a feasible number of picks by process path and storage zone. It does this by solving an MIQP to try to balance each process path and zone, minimizing the sum of squared shortages from their targets on a percentage (i.e., proportional to their target quantities) basis. Each process path is initially restricted to units with due times before some threshold. If any process path does not have sufficient units in the reduced backlog, its threshold is moved later and the MIQP is resolved. This is repeated until all process paths have sufficient units to meet their target rates or have no more available units in the backlog. The additional notation used for the MIQP is as follows:

- J^P: set of shipments that are currently partially picked
- J_i^B: set of shipments in backlog for process path i with due time at most $\bar{d}_i$ with $J^B = \cup_{i \in I} J_i^B$
- a_{pk}: units of inventory for SKU k on pod p at start of current time period
- b_{jk}: units of SKU k for shipment j in backlog at start of current time period
- c_z: available zone capacity in zone z during current time period
- f_i: target rate for process path i during current time period
- g_i: partial picking allowance for process path i
- $\bar{d}_i$: latest due time for process path i

The variables in the MIQP are as follows:

- w_j: binary variable indicating whether shipment j is selected into the reduced pick backlog
- y_{pk}: number of units of SKU k to be picked from pod p
- u_j: binary variable indicating whether shipment j is partially but not completely picked
- s_{jk}: number of units of SKU k not picked for shipment j
- r_i: target rate down-scaling factor for process path i
- q_z: zone capacity downscaling factor for zone z

The MIQP can then be written as

$$\min \sum_{i \in K} (1 - r_i)^2 + \sum_{z \in Z} (1 - q_z)^2 \tag{A.18}$$

subject to (A.19)

$$\sum_{j \in J^B} b_{jk} w_j - \sum_{j \in J^B} s_{jk} = \sum_{p \in P} y_{pk} \quad \forall k \in K, \tag{A.20}$$

$$\sum_{k \in K} s_{jk} \le \left(\sum_{j \in K} b_{jk} - 1 \right) u_j \quad \forall j \in J^B, \tag{A.21}$$

$$\sum_{j \in J_i^B} u_j \le g_i \quad \forall i \in I, \tag{A.22}$$

$$\sum_{k \in K, p \in P_z} y_{pk} = c_z q_z \quad \forall z \in Z, \tag{A.23}$$

$$\sum_{k \in K, j \in J_i^B} b_{jk} w_j - \sum_{j \in J_i^B} s_{jk} = f_i r_i \quad \forall i \in I, \tag{A.24}$$

$$0 \le y_{pk} \le a_{pk} \quad \forall k \in K, p \in P, \tag{A.25}$$

$$\sum_{p \in P} y_{pk} + \sum_{j \in J^B} s_{jk} \le \sum_{p \in P} a_{pk} \quad \forall k \in K, \tag{A.26}$$

$$0 \le s_{jk} \le b_{jk} \quad \forall k \in K, j \in J^B, \tag{A.27}$$

$$0 \le r_i \le 1 \quad \forall i \in I, \tag{A.28}$$

$$0 \le q_z \le 1 \quad \forall z \in Z, \tag{A.29}$$

$$w_j = 1 \quad \forall j \in J^P, \tag{A.30}$$

$$w_j \in \{0,1\} \quad \forall j \in J^B, \tag{A.31}$$

$$x_p \in \{0,1\} \quad \forall p \in P. \tag{A.32}$$

This phase returns a reduced pick backlog $\bar{B}$ indicated by the values of w_j in the MIQP solution, and only shipments from $\bar{B}$ are considered in subsequent phases. Furthermore, it returns zone capacity utilization rates α_z (given by the value of q_z in the solution) that we can realistically manage to achieve.

In phase 2, the pod-selection phase, we find a collection of pods to cover the items that need to be picked from $\bar{B}$, approximately satisfying the output of the feasibility phase. The goal is to have a small subset of pods for subsequent phases. An initial set cover problem using pods to cover units is solved using an MIP over the reduced backlog resulting from the feasibility phase. The primary objective of the set covering is to minimize the number of pods selected, and a secondary goal is to come close to the zone utilizations of the previous phase. A sequence of set cover problems is then solved, each generating a set of pods to cover the items that need to be picked with due time before a given threshold. This due-time threshold is brought forward in time with each subsequent problem solved, and the set of pods is restricted to be a subset of the solution from the previous problem in the sequence, thus producing a sequence of nested subsets of pods. The initial set cover problem uses the following additional notation:

- $\hat{P}$: set of pods already in current pick window or en route to pick station
- $J_{i\bar{B}}$: set of shipments in reduced backlog for process path i with due time at most $\bar{d}_i$ with $J^{\bar{B}} = \left(\cup_{i \in I} J_i^{\bar{B}} \right) \cup J^P$

• μ_z: penalty per unit of workload deficit in zone z

The variables in the initial set cover problem are as follows:

• x_p: binary variable indicating whether pod p is selected in the pod cover

• y_{zk}: units of SKU k to be picked from zone z

• l_z: units of workload deficit in zone z

The initial set cover problem can then be written as

$$\min \sum_{p\in P} x_p + \sum_{z\in Z} \mu_z l_z \tag{A.33}$$

subject to (A.34)

$$\sum_{p\in P_z} a_{pk}x_p \geq y_{zk} \qquad \forall z \in Z, k \in K, \tag{A.35}$$

$$\sum_{z\in Z} y_{zk} = \sum_{j\in J^{\bar{B}}} b_{jk} \qquad \forall k \in K, \tag{A.36}$$

$$\sum_{k\in K} y_{zk} + l_z \geq \alpha_z \sum_{k\in K,\, j\in J^{\bar{B}}} b_{jk} \qquad \forall z \in Z, \tag{A.37}$$

$$l_z \geq 0 \qquad \forall z \in Z, \tag{A.38}$$

$$y_{zk} \geq 0 \qquad \forall z \in Z, k \in K, \tag{A.39}$$

$$x_p = 1 \qquad \forall p \in \hat{P}, \tag{A.40}$$

$$x_p \in \{0,1\} \qquad \forall p \in P. \tag{A.41}$$

This returns a pod cover P^* (given by the values of x_p in the solution) that holds sufficient inventory for the shipments in $\bar{B}$. Next, we solve a sequence of set cover problems, each time considering a smaller subset of the reduced backlog (based on due times) and constraining the feasible set of pods to be a subset of the solution to the previous set cover problem, starting with P^*. The notation used in each set cover problem in the sequence is as follows:

• d: the latest due time considered in this set cover problem; we move d earlier each time we proceed to the next problem in the sequence

• P_d^{next}: set of pods that was the output of the previous set cover problem in the sequence (the first problem has $P_d^{\text{next}} = P^*$)

• $J^{\bar{B}}(d)$: set of shipments in reduced backlog with due time at most d

There is only one variable in each set cover problem in the sequence: x_p, the indicator variable for whether pod $p \in P^*$ is selected in the pod cover. Each problem can be formulated as follows:

$$\min \sum_{z\in Z,\, p\in P_z} \frac{1}{\alpha_z} x_p \tag{A.42}$$

subject to (A.43)

$$\sum_{p\in P} a_{pk}x_p \geq \sum_{j\in J^{\bar{B}}(d)} b_{jk} \qquad \forall k \in K, \tag{A.44}$$

$$x_p = 1 \qquad \forall p \in \hat{P}, \tag{A.45}$$

$$x_p = 0 \qquad \forall p \notin P_d^{\text{next}}, \tag{A.46}$$

$$x_p \in \{0,1\} \qquad \forall p \in P. \tag{A.47}$$

The output of this phase is a nested sequence of pod sets P_d^* that are planned to be picked to fulfill shipments from the reduced backlog at varying time horizons. Thus, if the smallest due time considered in the sequence of set cover problems is $d = \underline{d}$, then we have $P_{\underline{d}}^*$, the set of pods that we plan to use in the near future (say, in the next few hours) to cover $J^{\bar{B}}(\underline{d})$, the subset of $\bar{B}$ that we plan to pick in the next few hours.

In phase 3, the provisional schedule creation phase, we select shipments from $J^{\bar{B}}(\underline{d})$ and match them to pods from $P_{\underline{d}}^*$. The goal is to select a pick window and schedule that limits the number of times that a selected pod needs to be used again in a future time period. To do so, suppose that, for each SKU k from each shipment j, we plan to pick y_{jk} units, and these units are to be picked from pods P^{selected}. We define the *deficiency* π_k of a SKU k to be the remaining units of k required in the reduced backlog $\bar{B}$ minus the available inventory of pods that have not yet been selected:

$$\pi_k = \sum_{j\in \bar{B}} \left(b_{jk} - y_{jk}\right) - \sum_{p\in P^* \setminus P^{\text{selected}}} a_{pk}. \tag{A.48}$$

This can be interpreted as the number of units of this SKU that should be picked from pods selected so as to avoid being forced to pick that SKU from one of these pods again in the future (we may assume that the total number of units for any SKU in the backlog does not exceed available inventory). A positive deficiency implies that a previously selected pod has to be selected again at some point in the future. Therefore, we should aim to pick units from pods in such a way as to reduce the deficiency to zero or less. At the same time, we also want to avoid picking more than the deficiency to avoid being overly greedy and having a high pile-on now at the expense of future pile-on. Therefore, if we are able to have $\pi_k(t) = 0\ \forall k, t$, then we can avoid greediness in the pick schedule by picking just the right amount from each pod so that the pod is no longer needed in the rest of the schedule, helping to achieve a stable pile-on over time.

All shipments are given three scores: based on how much they reduce the deficiency, whether they can be picked from an independent pod (pods that cover whole shipments only—these are desirable pods, but we also want to avoid selecting too many of them now and leaving too few for future periods), and what fraction of the shipment has been picked previously. A greedy algorithm is run to select a collection of pods that cover the most preferred shipments, and the shipments are ranked according to the appropriate score, depending on which score is most relevant at the time based on the state of the backlog and which pods and shipments the algorithm has previously selected. One complication is to ensure that this greedy approach respects the workload allocated to each zone, but for simplicity, we omit how this is handled by the algorithm. The output of this phase is a provisional pick schedule: the set of shipments J^{prov} with units in the pick window (units planned to be picked over the next 15 minutes or so), and the set of pods P^{prov} that covers these units.

In phase 4, the final schedule creation phase, we search in the neighborhood of the solution from the previous phase to determine a final output of order shipments to be picked next and from which pod each item in those order shipments should be picked. We augment the pod collection P^{prov} from the provisional schedule creation phase with the smallest set of pods from P^* that allows us to complete all partially picked shipments in J^{prov}, giving us more freedom to choose which subset of shipments to only partially pick and which to completely pick. Then, we solve an MIP to select pods from this augmented collection of pods P^{M^*} and shipments from the backlog to minimize total deficiency. This is essentially the full problem we want to solve in the picking module, but in this fourth phase, we select from among the smaller set of pods determined by the previous phases. The additional notation used in the phase 4 MIP is as follows:

- $P_z^{M^*}$: considered set of pods in zone z with $P^{M^*} = \cup_{z \in Z} P_z^{M^*}$ determined as described
- J_i^{prov}: subset of J^{prov} in process path i
- J_i^M: set of shipments in process path i considered in phase 4 with $J_i^M = J_i^{\text{prov}} \cup J_i^{\bar{B}}, J^M = \cup_{i \in I} J_i^M$
- $J_i^M(d)$: set of shipments in J_i^M with due time at most d

The variables in the model are as follows:

- x_p: binary variable indicating whether pod p is selected
- w_j: binary variable indicating whether shipment j is selected into the pick window
- y_{pk}: number of units of SKU k to be picked from pod p
- s_{jk}: number of units of SKU k not picked for shipment j
- π_k: deficiency of SKU k

The formulation can then be written as

$$\min \sum_{k \in K} \pi_k \tag{A.49}$$

subject to (A.50)

$$\sum_{j \in J^M} b_{jk} w_j - \sum_{j \in J^M} s_{jk} = \sum_{p \in P^{M^*}} y_{pk} \qquad \forall k \in K, \tag{A.51}$$

$$\sum_{k \in K} s_{jk} \le \left(\sum_{k \in K} b_{jk} - 1 \right) w_j \qquad \forall j \in J^M, \tag{A.52}$$

$$\sum_{j \in J^M} s_{jk} \le \sum_{p \in P^{M^*}} a_{pk} \qquad \forall k \in K, \tag{A.53}$$

$$y_{pk} \le a_{pk} x_p \qquad \forall k \in K, p \in P^{M^*}, \tag{A.54}$$

$$\sum_{p \in P_z^{M^*}} y_{pk} \le c_z \qquad \forall z \in Z, \tag{A.55}$$

$$\sum_{j \in J_i^M} s_{jk} \le g_i \qquad \forall i \in I, \tag{A.56}$$

$$\underline{f}_i \le \sum_{k \in K, j \in J_i^M} (b_{jk} w_j - s_{jk}) \le \bar{f}_i \qquad \forall i \in I, \tag{A.57}$$

$$\underline{h}_{id} \le \sum_{k \in K, j \in J_i^M(d)} (b_{jk} w_j - s_{jk}) \qquad \forall i \in I, d \in D_i, \tag{A.58}$$

$$\pi_k \ge \sum_{j \in J^M} b_{jk}(1 - w_j) - \sum_{p \in P^{M^*}} a_{pk}(1 - x_p) + \sum_{j \in J^M} s_{jk} \qquad \forall k \in K, \tag{A.59}$$

$$\pi_k \ge 0 \qquad \forall k \in K, \tag{A.60}$$

$$0 \le y_{pk} \le a_{pk} \qquad \forall k \in K, p \in P^{M^*}, \tag{A.61}$$

$$0 \le s_{jk} \le b_{jk} \qquad \forall k \in K, j \in J^M, \tag{A.62}$$

$$w_j \in \{0,1\} \qquad \forall j \in J^M, \tag{A.63}$$

$$x_p \in \{0,1\} \qquad \forall p \in P^{M^*}. \tag{A.64}$$

The output of this phase is then the final pick schedule: the set of shipments with units in the pick window, the set of pods from which these units should be picked, and the number of units of each SKU to be picked from each pod for each shipment.

Appendix B. Pile-on and Cycle-Time Models

A description of both the pile-on and cycle-time models is given in this appendix. Using these two models in conjunction allows us to quantitatively estimate the trade-off between pile-on and cycle time when changing the pick-window size and update frequency. Here, we measure pick-window size by the length of time it takes to completely pick everything in the pick window given a constant pick rate.

B.1. Pile-on Model

The pile-on model estimates the pile-on that can be achieved given pick-window size and update frequency, number of pods, size of backlog, size of backlog by priority levels (based on due time), and required picks by priority levels. It begins with an initial pile-on estimate and then, assuming that all units are uniformly randomly distributed among each of the pods, computes the expected number of pods needed to cover enough units to replenish the pick window. This gives an expected pile-on for the entire pick window, which we use as the pile-on estimate for the next iteration. We iterate until the pile-on converges, and we take the limit as the final expected pile-on.

In this model, we assume that we have a pick-window size of n minutes, planned to be picked at a uniform rate of η units per minute. Thus, the pick window contains $N = n\eta$ units. After $N\alpha$ units have been picked, the pick window is replenished up to size N, selecting from an eligible backlog of size B located in γ pods. We assume that some proportion B_α of the backlog is new and could potentially be already covered by the pods previously selected to cover the $N(1-\alpha)$ units remaining in the pick window. We assume that all units are uniformly randomly distributed among the pods. We initialize the system with the remaining $N(1-\alpha)$ units covered by $\gamma_{N(1-\alpha)}$ pods, and we want to understand the expected pile-on in such a system. In practice, the model needs to handle the additional complication that varying due times in the backlog reduces the degree of freedom to select units, but for simplicity, we omit that part of the model.

First, we define a few functions. Suppose we have χ units that are uniformly randomly distributed among γ pods, and we want to compute $F(\chi, \rho, \gamma)$, the expected number of these units covered by a fixed set of ρ pods. Then, we have $F(\chi, \rho, \gamma) = \chi\rho/\gamma$.

Next, suppose that we have to pick χ units from a selection of β units that are uniformly randomly distributed among γ pods, and we want to estimate the probability $\phi_\xi(\chi, \beta, \gamma)$ that ξ of these γ pods are sufficient to cover χ units. The number of units covered by a given subset of ξ pods can be modeled as a $\text{Bin}(\beta, \xi/\gamma)$ random variable, which can be approximated as a $N(\beta\xi/\gamma, \beta\xi(\gamma - \xi)/\gamma^2)$ random variable. The probability that this given subset of ξ pods covers the sufficient number of units is then approximated by

$$\psi_\xi(\chi, \beta, \gamma) = \mathbb{P}\left[Z \ge \frac{\chi - \beta\frac{\xi}{\gamma}}{\sqrt{\beta\frac{\xi}{\gamma}\frac{\gamma - \xi}{\gamma}}} \right],$$

where Z is the standard normal variable. But there are $\binom{\gamma}{\xi}$ possible subsets of ξ pods, so if we make the approximation that whether each subset covers the required units is independent of each other, then we have $\phi_\xi(\chi, \beta, \gamma) = 1 - (1 - \psi_\xi(\chi, \beta, \gamma))^{\binom{\gamma}{\xi}}$. Computationally, we further approximate $\binom{\gamma}{\xi}$ by $(\gamma e/\xi)^\xi / (\sqrt{2\pi\xi})$.

Finally, given χ units required from β units uniformly randomly distributed among γ pods, the expected number of pods $G(\chi,\beta,\gamma)$ required to cover these units can be approximated by $G(\chi,\beta,\gamma) = \min(\sum_{\xi=0}^{\gamma}(1-\phi_\xi(\chi,\beta,\gamma)),\chi)$, using the complementary cumulative distribution function formula for the expectation of a discrete random variable and the fact that no more than χ pods are needed to cover χ units.

Now we are ready to consider the expected pile-on.

1. At the time of replenishing the pick window, we are left with $N(1-\alpha)$ units in the pick window covered by $\gamma_{N(1-\alpha)}$ pods.

2. We consider how many pods we need to cover the required $N\alpha$ units needed to replenish the pick window.

3. $\text{NewUnitsCovered} = \min(F(BB_\alpha,\gamma_{N(1-\alpha)},\gamma),N\alpha)$ of the backlog are units that are new to the backlog and already covered by the $\gamma_{N(1-\alpha)}$ pods.

4. This leaves $\text{RemainingUnits} = N\alpha - \text{NewUnitsCovered}$ to be covered.

5. This requires $\text{AdditionalPods} = G(\text{RemainingUnits}, B - \text{NewUnitsCovered}, \gamma - \gamma_{N(1-\alpha)})$ additional pods.

6. Therefore, we have expected pile-on

$$\pi = \frac{N}{\gamma_{N(1-\alpha)} + \text{AdditionalPods}}.$$

7. Iterating and replacing $\gamma_{N(1-\alpha)}$ by $N(1-\alpha)/\pi$ in each iteration, we converge to an expected pile-on.

In practice, accounting for varying due times complicates the way we compute π in each iteration.

B.2. Cycle-Time Model

The cycle-time model estimates the cycle time by order shipment size (number of items in the order shipment) given pick-window size and update frequency. The model assumes the following dynamics. The pick window starts with the number of items that can be picked over the pick-window size given a fixed constant pick rate. Items are picked according to this pick rate with the next item to be picked chosen uniformly at random from the remaining items in the pick window and independent of past picks. When the time comes for the pick window to be updated (after a constant time has passed based on the update frequency), unpicked items in the pick window remain in the pick window, and new items are added to bring it up to the original size. Items belonging to the same order shipment always enter the pick window at the same time. Based on these dynamics, we derive a closed-form expression for the expected cycle time for any order shipment size.

More concretely, we define the target size of the pick window to be the time taken to complete picking, n minutes. According to the dynamics described, the pick time of a unit is independently and uniformly distributed over the target window size, but if it is not picked in the first $n\alpha$ minutes, in which $\alpha \in (0,1]$ is the window update frequency, then the window is replenished up to its target size and the n minute period is reset. Therefore, each item is picked in the Xth $n\alpha$-minute period, where $X \sim Geo(\alpha)$, and the pick time of each item is $(X-1)n\alpha + Y$, where $Y \sim U[0,n\alpha]$ is independent of X.

Let $Z^m_{(u)}$ be the time of the uth pick of a shipment of size m. We wish to compute the expected time between the $(n-1)$ th and nth pick for a shipment of size m given by

$$\lambda^m_u = \mathbb{E}\left[Z^m_{(u)} - Z^m_{(u-1)}\right]. \tag{B.1}$$

We note that

$$\lambda^m_u = \lambda^{m-1}_{u-1} \qquad \forall m \ge 2, 3 \le u \le m, \tag{B.2}$$

so it suffices to compute λ^m_2 for all m. Conditioning on how many of the m items are picked in the first $n\alpha$-minute period, we get the following:

$$\lambda^m_2 = (1-\alpha)^m\lambda^m_2 + m\alpha(1-\alpha)^{m-1}n\alpha\mathbb{E}[W] + \sum_{v=2}^{m}\binom{m}{v}\alpha^v(1-\alpha)^{m-v}\mu^{v,m}_2, \tag{B.3}$$

where W is the number of periods separating the first and second pick and

$$\mu^{v,m}_2 = \mathbb{E}\left[Z^m_{(2)} - Z^m_{(1)} \mid Z^m_{(v)} \le n\alpha < Z^m_{(v+1)}\right] \tag{B.4}$$

is the expected time between the first two picks of a shipment of size m given that exactly v of them are picked in the first $n\alpha$-minute period.

Using the memoryless property of the geometric distribution, $W \sim Geo(1-(1-\alpha)^{m-1})$, so

$$\mathbb{E}[W] = \frac{1}{1-(1-\alpha)^{m-1}}. \tag{B.5}$$

Using the fact that the uth order statistic of v uniformly distributed random variables follows a $\text{Beta}(u, v+1-u)$ distribution, we have

$$\mu^{v,m}_2 = n\alpha\frac{2}{v+1} - n\alpha\frac{1}{v+1} = n\alpha\frac{1}{v+1}. \tag{B.6}$$

Now, we have

$$\lambda^m_2 = \frac{n\alpha}{1-(1-\alpha)^m}\left[\frac{q\alpha(1-\alpha)^{m-1}}{1-(1-\alpha)^{m-1}} + \sum_{v=2}^{m}\binom{m}{v}\alpha^v(1-\alpha)^{m-v}\frac{1}{v+1}\right]. \tag{B.7}$$

Because

$$\sum_{v=0}^{m+1}\binom{m+1}{v}\alpha^v(1-\alpha)^{m+1-v} = 1, \tag{B.8}$$

we can also write this as

$$\lambda^m_2 = \frac{n\alpha}{1-(1-\alpha)^m}\left[\frac{m\alpha(1-\alpha)^{m-1}}{1-(1-\alpha)^{m-1}} + \frac{1-(1-\alpha)^{m+1}}{\alpha(m+1)} - (1-\alpha)^m - \frac{m}{2}\alpha(1-\alpha)^{m-1}\right]. \tag{B.9}$$

This gives a closed-form solution. From this, we can obtain the expected time between any two consecutive picks $\lambda^m_u = \lambda^{m-u+2}_2$ and total cycle time $\Gamma^m = \sum_{u=2}^{m}\lambda^m_u$.

References

Amazon (2018) Bots by the numbers: Facts and figures about robotics at Amazon. Accessed February 6, 2020, https://blog.aboutamazon.com/innovation/bots-by-the-numbers-facts-and-figures-about-robotics-at-amazon.

Amazon (2019) New robots, new jobs. Accessed January 24, 2022, https://www.aboutamazon.com/news/operations/new-robots-new-jobs.

Amazon (2020) What robots do (and don't do) at Amazon fulfillment centers. Accessed February 6, 2020, https://www.aboutamazon.co.uk/amazon-fulfilment/what-robots-do-and-dont-do-at-amazon-fulfilment-centres.

Azadeh K, De Koster R, Roy D (2019) Robotized and automated warehouse systems: Review and recent developments. *Transportation Sci.* 53(4):917–945.

Boysen N, Briskorn D, Emde S (2017) Parts-to-picker based order processing in a rack-moving mobile robots environment. *Eur. J. Oper. Res.* 262(2):550–562.

Boysen N, de Koster R, Weidinger F (2019) Warehousing in the e-commerce era: A survey. *Eur. J. Oper. Res.* 277(2):396–411.

Cezik T, Graves SC, Liu A (2021) Velocity-based stowage policy for semi-automated fulfillment system. Preprint, submitted February 18, https://dx.doi.org/10.2139/ssrn.3784731.

De Koster R, Le Duc T, Roodbergen KJ (2007) Design and control of warehouse order picking: A literature review. *Eur. J. Oper. Res.* 182(2):481–501.

Enright J, Wurman PR (2011) Optimization and coordinated autonomy in mobile fulfillment systems. *Proc. 25th AAAI Conf. Artificial Intell.* (ACM, New York), 33–38.

Gagliardi JP, Renaud J, Ruiz A (2012) Models for automated storage and retrieval systems: A literature review. *Internat. J. Production Res.* 50(24):7110–7125.

Gallien J, Weber T (2010) To wave or not to wave? Order release policies for warehouses with an automated sorter. *Manufacturing Service Oper. Management* 12(4):642–662.

Gharehgozli A, Zaerpour N (2020) Robot scheduling for pod retrieval in a robotic mobile fulfillment system. *Transportation Res. Part E Logist. Transportation Rev.* 142:102087.

Lamballais T, Roy D, De Koster M (2017) Estimating performance in a robotic mobile fulfillment system. *Eur. J. Oper. Res.* 256(3): 976–990.

Lamballais T, Roy D, De Koster R (2019) Inventory allocation in robotic mobile fulfillment systems. *IISE Trans.* 52(1):1–22.

Merschformann M, Lamballais T, de Koster M, Suhl L (2019) Decision rules for robotic mobile fulfillment systems. *Oper. Res. Perspect.* 6(C):100128.

Qin H, Xiao J, Ge D, Xin L, Goa J, He S, Hu H, Carlsson JG (2022) JD.com: Operations research algorithms drive intelligent warehouse robots to work. *INFORMS J. Appl. Anal.* 52(1):42–55.

Qiu L, Hsu WJ, Huang SY, Wang H (2002) Scheduling and routing algorithms for AGVS: A survey. *Internat. J. Production Res.* 40(3):745–760.

Rimélé A, Gamache M, Gendreau M, Grangier P, Rousseau LM (2021) Robotic mobile fulfillment systems: A mathematical modelling framework for e-commerce applications. *Internat. J. Production Res.* 60(11):3589–3605.

Roodbergen KJ, Vis I (2009) A survey of literature on automated storage and retrieval systems. *Eur. J. Oper. Res.* 194(2):343–362.

Valle CA, Beasley JE (2021) Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. *Comput. Oper. Res.* 125:105090.

van Gils T, Ramaekers K, Caris A, de Koster RB (2018) Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *Eur. J. Oper. Res.* 267(1):1–15.

Vis I (2006) Survey of research in the design and control of automated guided vehicle systems. *Eur. J. Oper. Res.* 170(3):677–709.

Wang Z, Sheu JB, Teo CP, Xue G (2021) Robot scheduling for mobile-rack warehouses: Human–robot coordinated order picking systems. *Production Oper. Management* 31(1):98–116.

Weidinger F, Boysen N, Briskorn D (2018) Storage assignment with rack-moving mobile robots in Kiva warehouses. *Transportation Sci.* 52(6):1479–1495.

Wurman PR, D'Andrea R, Mountz M (2008) Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29(1):9.

Yuan R, Cezik T, Graves SC (2018) Stowage decisions in multi-zone storage systems. *Internat. J. Production Res.* 56(1–2):333–343.

Yuan R, Cezik T, Graves SC (2019) Velocity-based storage assignment in semi-automated storage systems. *Production Oper. Management* 28(2):354–373.

Zou B, Gong YY, Xu X, Yuan Z (2017) Assignment rules in robotic mobile fulfilment systems for online retailers. *Internat. J. Production Res.* 55(20):6175–6192.

Verification Letter

Jeetu Mirchandani, Director, Software Development, Amazon Fulfillment Technology, Amazon.com, 410 Terry Avenue North, Seattle, Washington 98109, writes:

"I write this letter in support of the submission titled 'Algorithm for Robotic Picking in Amazon Fulfillment Centers Enables Humans and Robots to Work Together Effectively' to the *INFORMS Journal on Applied Analytics*.

"Since its founding in 1994, Amazon.com has grown from an online bookseller to a global retail company. As its customer base grew, inventory volume increased, and the network vastly expanded. As a result, many teams at Amazon had to create solutions that would improve the overall customer experience and benefits of shopping with Amazon, such as Prime and one-day shipping.

"Fulfillment centers (FCs) serve as one of the most critical elements in the supply chain as it is here where inventory is stored and customer orders are picked, packed, and shipped. Quickly outgrowing its first FC (Jeff Bezos' garage), Amazon had to expand to larger buildings where more inventory could be stored to meet the growing customer base. In parallel, the e-commerce industry was booming, which put more pressure on Amazon to implement innovative solutions within its FCs in order to scale and continue to meet the demands of the global network. One of the most impactful changes for its fulfillment centers was the 2012 acquisition of Kiva Systems, now known as Amazon Robotics (AR), which led to the introduction of robotics.

"This robotic technology enabled a semiautomated storage system—robots and teams of associates working alongside one another to efficiently and safely fulfill customer orders. Rather than Amazon's associates going to the products to pick a customer order or stow new inventory, robots bring shelves of inventory to the associates who are either picking or stowing items. This new system brought multiple advantages to Amazon's buildings. By robotically bringing pods of inventory to pick stations, associates were able to reduce the time they spent walking from one shelf to another and redirect that time to more value-added tasks, such as problem detection and IT management. The new design of the storage system also allowed Amazon to stow 40% more inventory in its buildings due to the compact nature of how its shelving pods are filled and stored. This robotic system became extremely efficient and allowed Amazon employees to more quickly move items throughout the FCs, helping speed up delivery times for customers.

"None of this happened overnight. For the new technology to work in Amazon.com FCs, teams first had to customize the system and software to scale to the needs of the buildings and Amazon's growing network. Teams in both Seattle and Boston took on this challenge. By 2014 Amazon's first robotic FC was up and running. Initial results and associate feedback on the new process was positive. However, Amazon saw room for improvement and adjustments had to be made to deliver results at a global scale.

"In 2015, the modeling and optimization team at Amazon.com redesigned the robotic picking algorithm so that robotic drives could more efficiently bring items to associates. The new algorithms minimized the average distance the robots had to travel per item picked. Not only did the algorithms reduce travel distance, they also reduced idle time between pod interactions, another benefit to the associates. After simulating the algorithms, the teams implemented them in the software running the FCs, which has since been introduced to more than 50 robotic fulfillment centers.

"Results were far better than expected. The distance traveled by drives per unit picked was reduced by 62% without operational impact. This improvement reduced the number of drives AR FCs required by 31%, which in turn amounted to half a billion dollars in savings to Amazon. Moreover, the algorithm increased the likelihood an associate would be able to pick multiple items from one storage pod, minimizing travel time and pod turnover. The redesigned algorithm supported the overall Amazon Robotic system, which enables seamless collaboration between associates and robots, creating more jobs and career options for associates. The system's ability to scale and capture the advantages of robotic technology and pod-based storage convinced Amazon executives to make AR FCs the standard for new FCs, allowing Amazon to reduce the storage footprint compared with what would have been required if only non-AR FCs were constructed.

"The financial results spoke for themselves and served as evidence of the value of robots to everyone throughout Amazon, ensuring the success of the 2012 Kiva acquisition and encouraging further investment in robotics. Since the introduction of robots, Amazon has added more than a million new jobs globally, including jobs that were created because of the use of robots, such as positions in IT as well as roles necessary for the service and maintenance of the robots. Furthermore, the AR FCs often employ more people due to the higher volume of inventory managed in these buildings, as well as the variety of jobs supporting these robotic systems.

"With a focus on algorithmic efficiency and effectiveness, operations research has been able to shorten the overall development time necessary for wide scale deployment, while increasing the overall performance of a critical robotic system used within Amazon. The algorithmic improvements have enabled sustained improvements in operational efficiency and have played a key role in accelerating Amazon's fulfillment operations, enabling humans and robots to work together efficiently."

Russell Allgor is the chief scientist for Amazon.com, where he leads a team of mathematical modeling experts to improve the efficiency of Amazon's operations using data analysis, modeling, simulation, and optimization. Ideas and algorithms developed by Russell and his team have returned billions of dollars to Amazon's bottom line. Before joining Amazon.com, he worked in applied R&D for Bayer AG in Germany. Russell holds a PhD in chemical engineering from MIT and a BS from Princeton University.

Tolga Cezik is a senior principal research scientist at Amazon.com, where his research focuses on process design and algorithms for fulfillment center operations and transportation systems. He has previously worked as a researcher at Bell Labs, Université de Montréal, and Tilburg University. Tolga holds a PhD in operations research from Columbia University and a BS in industrial engineering and operations research from Middle East Technical University.

Daniel Chen is a research scientist in the modeling and optimization team at Amazon.com, where his research focuses on improving the effectiveness of fulfillment operations. He also holds a concurrent appointment as a research scientist at the Institute of High Performance Computing in Singapore. Daniel holds a PhD in operations research from MIT and a BA in mathematics from the University of Cambridge.