Velocity-based Storage Assignment in Semi-automated Storage Systems

July 2016, revised January 2018

Rong Yuan[†] • Tolga Cezik[‡] • Stephen C. Graves[†]

[†] *A. P. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139-4307, rongyuan@mit.edu, sgraves@mit.edu*

[‡] *Amazon.com, Seattle, Washington 98109, cezikm@amazon.com*

Abstract

Our research focuses on the storage decision in a semi-automated storage system, where the inventory is stored on mobile storage pods. In a typical system, each storage pod carries a mixture of items, and the inventory of each item is spread over multiple storage pods. These pods are transported by robotic drives to stationary stations on the boundary of the storage zone where associates conduct pick or stow operations. The storage decision is to decide to which storage location within the storage zone to return a pod upon the completion of a pick or stow operation. The storage decision has a direct impact on the total travel time, and hence the workload of the robotic drives. We develop a fluid model to analyze the performance of velocity-based storage policies. We characterize the maximum possible improvement from applying a velocity-based storage policy in comparison to the random storage policy. We show that class-based storage with two or three classes can achieve most of the potential benefits and that these benefits increase with greater variation in the pod velocities. To validate the findings, we build a discrete-time simulator with real industry data. We observe an 8% to 10% reduction in the travel distance with a 2-class or 3-class storage policy, depending on the parameter settings. From a sensitivity analysis we establish the robustness of the class-based storage policies as they continue to perform well under a broad range of warehouse settings including different zoning strategies, resource utilization and space utilization levels.

Key words: velocity-based storage, class-based storage, semi-automated storage system

1 Introduction

Online retailers are investing in automation systems in their order fulfillment centers in order to meet higher demand and greater service expectations. With the introduction of new technology comes new operational problems. In this paper we report on our research on some of these operational problems for a semi-automated storage system.

Online retailers rely on fulfillment centers (FCs) to receive and store their inventory, from which they fulfill customer orders. The typical operations of an FC entail receiving inventory from vendors, stowing this inventory into the FC's storage space, picking items from the inventory to fulfill customer orders, and then packaging these items for shipment to the customer.

We consider an FC that operates with a goods-to-person model for both stowing and picking. The inventory in the FC is stored on mobile storage pods. A typical pod has around 100 cubic feet of storage space. Each pod consists of shelves of various heights and depths that allows it to carry a large number of units, across different stock keeping units (sku's), depending on the size of the units. These pods are stored in a storage field that consists of a grid of storage locations. Each pod can be stored in any location in the grid. In Figure 1 we provide a representative example of a storage field.

The FC associates conduct pick or stow operations at static stations that are contiguous to the storage field. Each station is dedicated to either picking or stowing. To pick (stow) a unit, a robotic drive must first travel along the grid aisles to the storage pod that has been selected for picking (stowing). The robotic drive then carries the storage pod to a pick (stow) station on the boundary of the storage field. The pod and drive may have to queue at the station behind other pods that are waiting to be picked (stowed). When the pod reaches the station, the associate picks (stows) the selected units from (to) the pod. The drive then will return the pod to some open storage location in the storage field. As is evident from Figure 1, pod locations in the storage field vary significantly in terms of their accessibility and their distance from the pick and stow stations. See D'Andrea and Wurman (2008) and Enright and Wurman (2011) for more on the operational details and challenges of these systems, which they term mobile fulfillment systems.

This is a goods-to-person system in that the inventory (goods) is mobile, while the pick and stow operators are stationary. We term this a semi-automated storage system, reflecting the fact that the pod storage and retrieval operations are automated, yet the picking and stowing processes are still manual. In a semi-automated storage system, there are three major operational decisions: (1) the *stowage* *decision* that determines on what pods to store the inventory that is received by the FC; (2) the *picking decision* that decides from what pods to pick the inventory that is required to meet each customer order; and (3) the *storage decision* that selects the location in the storage field to return a pod upon the completion of a pick or stow operation. In this paper we focus on the storage decision, with an objective to minimize the total travel distance of the robotic drives in the system. To minimize the travel time, the general idea is to follow a velocity-based policy; that is, we place the popular (or higher velocity) items closer to the stations. For a semi-automated storage system, this notion is complicated for at least two reasons. First, each storage pod will hold a large set of different items. And each item will typically have its inventory stored on several different pods. We need a way to predict the velocity of each pod, as a measure of the popularity of the items it contains. Second, as there are multiple stations located on the boundary of the storage field, we need a way to determine the desirability (or closeness) of each storage location. In the following we provide models for analyzing and implementing a velocity-based storage policy in this context; we also present results from testing of these policies, showing the effectiveness and potential impact.

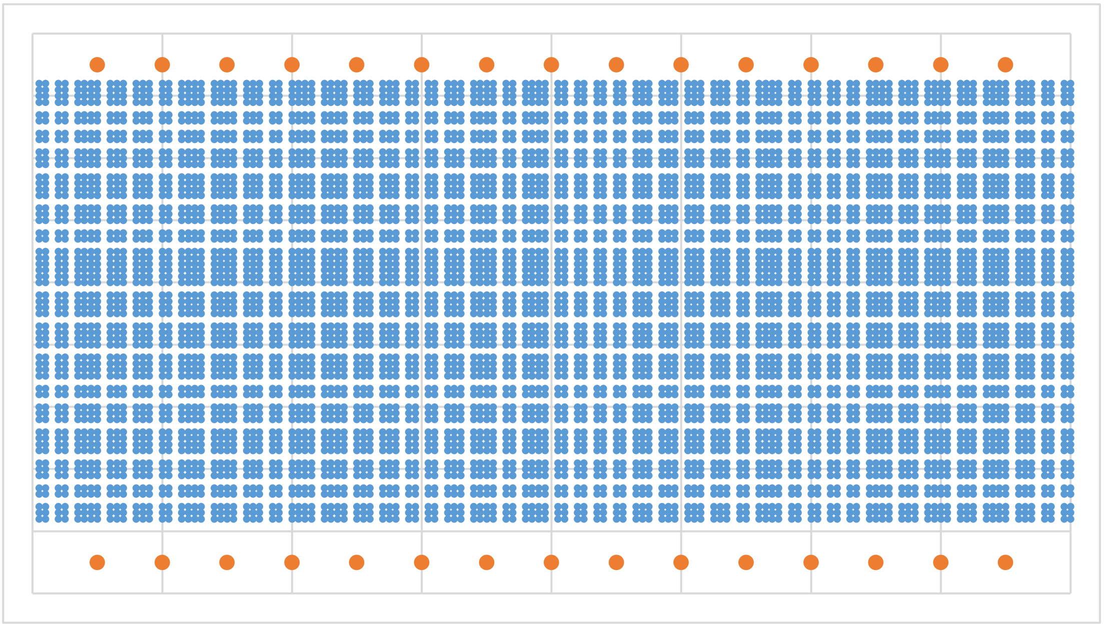


Figure 1: A Representative Grid Storage Field (small blue dots are storage locations, larger orange dots on boundary are stations for picking and stowing)

The rest of the paper is organized as follows. In Section 2 we review the related literature. In Section 3 we show the benefits of adopting a velocity-based storage policy in the semi-automated storage system with a fluid model. In Section 4 we verify these benefits with a simulation model with real data. We finally conclude our study and discuss future research opportunities in Section 5.

2 Literature Review

There is an extensive literature on general storage decisions in warehouses since the 1960's, especially in the context of an automated storage and retrieval system (AS/RS). Gagliardi et al. (2012), Bartholdi and Hackman (2011), Roodbergen et al. (2009), Gu et el. (2007), De Koster et al. (2007), and Rouwenhorst et al. (2000) provide reviews of this literature on storage decisions, as well as on other related issues in warehouse control and management.

There are in general four storage policies that have been examined in the literature, namely full-turnover-based storage, random storage, closest-open-location storage, and class-based storage. The full-turnover-based storage policy assigns the storage location based on the turnover (or velocity) of an item, usually represented by the demand rate. The implementation of the full-turnover-based storage policy will typically entail dedicated storage, in which a dedicated storage location is assigned to each item or sku. The main advantage of this storage policy is to save travel distance as the popular items are closer to the retrieval points. A disadvantage of this storage policy is that, if implemented as a dedicated storage policy, the space utilization is likely to be low; the amount of space dedicated to each item needs to accommodate the maximum inventory level of the item.

In contrast, random storage assigns a randomly-chosen storage location to each item. This can result in a high space utilization, as all storage locations are available to all storage units. But travel distance will increase, compared to a full-turnover-based storage policy. Closest-open-location storage assigns the closest open location to an arriving unit. This policy is widely adopted in practice as it minimizes the immediate travel distance. However, when the space utilization is high, a closest-open-location policy will perform similarly to random storage (e.g., see Hausman et al. (1976)).

Hausman et al. (1976) introduce a class-based storage policy for an AS/RS as a way of achieving the travel-time savings from velocity-based storage along with a high level of space utilization. The class-based storage policy divides the inventory units into classes according to their velocities or turnover rates; similarly, the storage area is divided into corresponding storage zones, based on distance. Each class of units is assigned to the corresponding storage zone. Within each zone the storage is random. In fact, the full-turnover-based storage can be viewed as a special case of class-based storage where the number of classes equals the number of items.

The literature confirms that velocity-based storage can outperform random storage in terms of the travel distance. Furthermore, in a variety of warehouse settings, most of the benefits from class-based storage can be achieved with a limited number of classes, i.e., with two or three classes. (Hausman et al. 1976, Graves et al. 1977, Rosenblatt and Eynan 1989, Guenov and Raeside 1992, Eynan and Rosenblatt 1994, Kouvelis and Papanicolaou 1995, Thonemann and Brandeau 1998, Van den Berg and Gademann 2000, De Koster et al. 2008).

Our work investigates the storage policies in the context of a semi-automated storage system. Our problem differs significantly from an AS/RS in several aspects. First, in the semi-automated storage system, the travel distance is based on the Manhattan distance, which is a consequence of the physical configuration of the grid storage field. This is different from the "square in time" measure used in the AS/RS literature, where an automated crane can move horizontally and vertically at the same time.

Second, similar to an AS/RS, we can place a storage unit (pod) at any storage location and a storage location can only store one pod. However, typically it is assumed that the AS/RS stores pallets that contain a single sku. A pod in the semi-automated storage system, however, may contain hundreds of different sku's, with the number of units of each sku varying from one up to several dozen. Furthermore, the inventory of each sku may be spread over several pods. This creates a great challenge in determining the velocity of a pod, which is a key input to any velocity-based policy.

Finally, the semi-automated storage system will have several stations for picking and stowing that are located around the storage field. An AS/RS will typically have one or two input/output locations at the end of each aisle. As a consequence of this difference, more care is needed in deciding which storage locations are desirable, in terms of travel distances.

In an effort to stimulate research, Enright and Wurman (2011) provide an overview of the resource allocation problems that arise in the operation of mobile fulfillment systems; the pod storage decision is one of the problems that is highlighted. Lamballais et al. (2017a, 2017b) appear to be the first researchers to model a mobile fulfillment system. They have developed a set of queueing models to determine the performance of the system as it depends on the number of robotic drives, the floor layout of the fulfillment center, and the volume and nature of the orders. Weidinger et al. (2016) consider the storage assignment problem for a semi-automated storage system, as described in this paper. They formulate the problem as a mixed-integer optimization under the assumption that the schedule of pod visits at the picking stations are known over the immediate planning horizon. The paper demonstrates the effectiveness of both exact and heuristic solutions strategies with an objective of minimizing robot travel time. Our work differs from Weidinger et al. (2016) in that in our context we do not have a schedule of future pod visits to the picking stations. As a consequence we examine dynamic storage policies that will rely on forecasts of when the pod will next return to the picking station.

3 Model

In a semi-automated storage system, the storage decision needs to be made upon the completion of a pick or stow operation; at this time the pod can be returned to any open storage location in the storage area. The storage decision has a direct impact on the total travel time of the pods, as one can try to store popular pods closer to the stations. Pod travel time is relevant because it determines the workload for the robotic drives; as a consequence, reducing the pod travel time reduces the number of drives that are required for a given system throughput rate (Enright and Wurman 2011; Yuan 2016; Lamballais et al. 2017a).

We develop a fluid model to evaluate the potential benefits from applying the velocity-based storage policies in a semi-automated storage system. We use the model to compute the expected travel distance for the robotic drives to complete the retrieval and storage of the pods. We associate with each pod a velocity measure that is an estimate of the expected number of trips (or tours) to a picking station in the next time period. Each storage policy assigns the storage locations to pods based on their velocity measures. We compute the expected travel distance for a storage policy based on multiplying the pod velocities by the storages distances from the location assignments.

In this section, we first list the key assumptions and describe the preliminary settings for the fluid model. We evaluate the velocity-based storage policy for a random stowage policy and then for a velocity-based stowage policy. We finally provide numerical examples for the analytical results.

3.1 Model Assumptions

In this section we present and discuss the assumptions for the model.

A1. *Fluidity of stock keeping units.* For analytical tractability we assume the size of the assortment is infinite. In reality, a storage system stocks on the order of 10^5 to 10^6 items or sku's. We index the items in descending order of expected demand on the continuum from 0 to 1.

A2. *Exponential demand for each item.* The demand of each item is the number of units required for picking in a specified fixed period of time, e.g., the next twelve hours. For each item we model its demand as a continuous random variable with an exponential distribution. This is a simplifying assumption that has some justification as it captures the long tail and high variability of the demand.

For instance, we consider the actual daily demand of over 120,000 items for a seven-day week for a typical storage field for a large retailer. We only include items for which the average daily demand is at least one unit per day. We order the items by the coefficient of variation of their daily demand and plot this ordering in Figure 2. We observe that the ratios are in the range of 0.5 to 1.5 for the majority of the items, which is reasonably consistent with the assumption that demand over a short period of time is exponentially distributed.

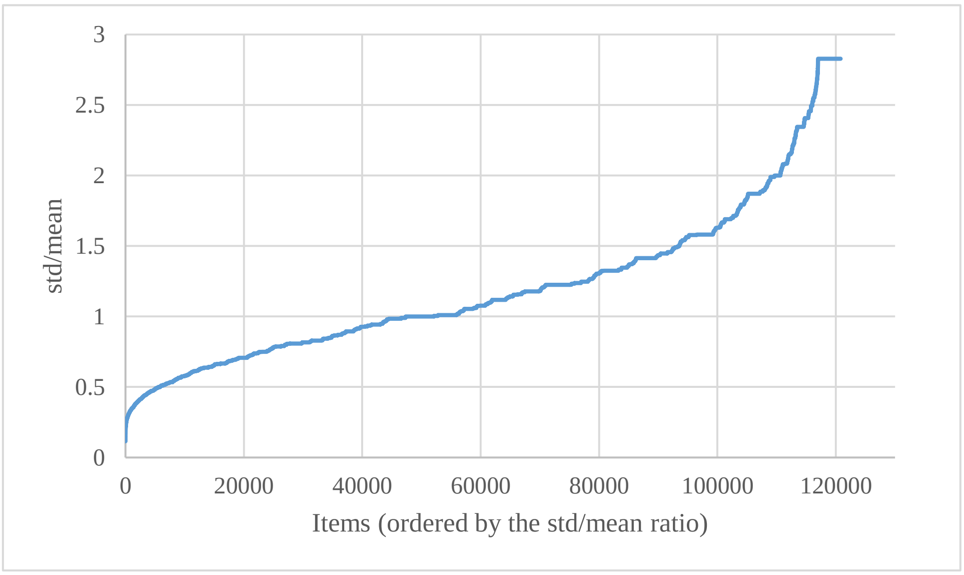


Figure 2: The Ratio of Standard Deviation to Mean of the Daily Demand of Over 120,000 Items in a Week

A3. *Ranked Inventory.* For each item we assume that we can fully rank (or order) its inventory, such that the higher ranked inventory will be picked first. Thus, if the demand for item i equals x units, then the x-highest ranked units of inventory will be picked to meet the demand. This assumption allows mathematical tractability for our pod velocity measure. Thonemann and Brandeau (1998) make an analogous assumption, in terms of pallet demand, in their extension of Hausman et al. (1976) to a stochastic environment.

A4. *Universal stock out rate for all items.* We set the inventory for each item so that its probability of stock-out within the next time period equals the parameter α. As we will see later, this is consistent with the inventory, expressed in terms of days of supply, being the same for all items.

A5. *Same unit size for all items, fixed pod size.* We assume that all pods have the same storage capacity and that all items have the same unit cubic size.

A6. *Number of picks independent of the pod's velocity measure.* We assume that when a pod is called to a pick station, the number of picks from it is independent of its velocity measure. As justification, there is not a high correlation between the actual number of units picked and the pod's velocity. The number of picks from a pod is usually limited by a variety of operational constraints such as the time window for picking, the due time of the orders, and the status of the orders being picked.

A7. *Linear travel distance.* We model the travel distance from the storage locations, ordered by distance, to the picking stations by a linear function: that is, the distance from the *x*-percentile closest location is given by $d(x) = hx$ for constant h and $0 \leq x \leq 1$. This is a simplifying assumption that has some justification based on the actual travel times for a typical storage field. For instance, Figure 1 shows a typical layout of the grid storage field with 30 picking stations, and with dimensions of 160 by 70 storage locations (including aisles). For this configuration, we evaluate the travel distance from each storage location to the closest pick station. In order to mimic the real operations, we assume one-way traffic for each vertical aisle and we add a constant travel time for any two-deep storage locations (i.e., any storage location that is not contiguous to an aisle). We plot the sorted travel distances of the storage locations for this configuration in Figure 3, and observe that the assumption of the linear travel distance is fairly reasonable.

To assess the robustness of the assumption, we repeated this evaluation for a wide range of configurations that are representative of actual storage operations. We varied the length of the field from 100 to 200 storage locations, and the width of the field from 50 to 100 storage locations. We locate stow/pick stations uniformly along each long side of the storage field, where the number of stations on each side is between 10 and 20. We populated the storage field with rectangular storage cells that are of dimension (x, y) for $2 \leq x \leq 4, 2 \leq y \leq 6,$ and $x \leq y$, integer. For instance a (3, 4) storage cell is a 3-column, 4-row arrangement of 12 storage locations. There is an aisle with width of one storage location separating the cells. For a large set of randomly generated configurations, we determined the ranked travel distances similar to Figure 3, and found that a linear model was a reasonable fit.

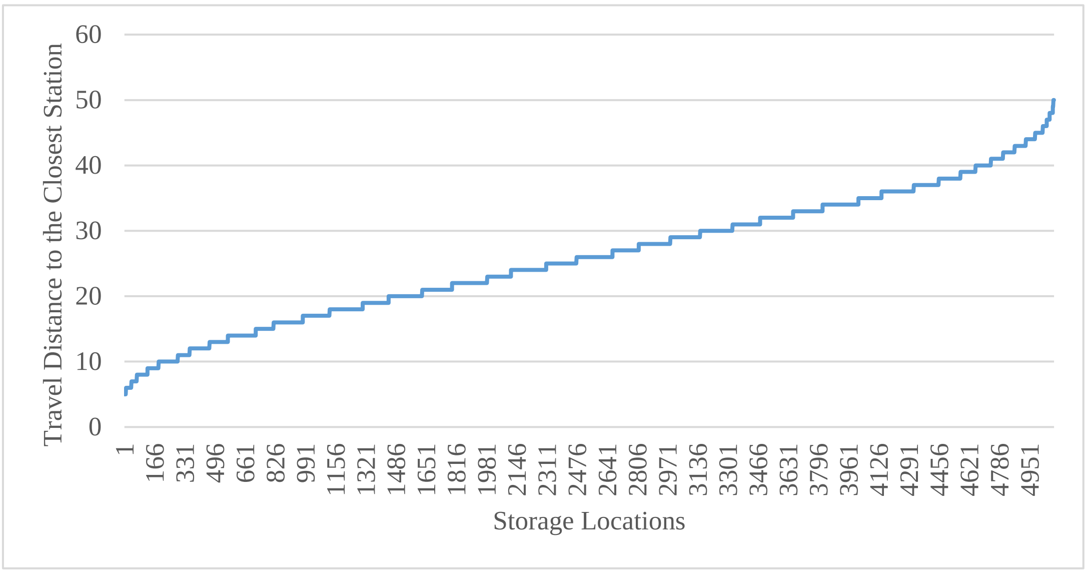


Figure 3: Travel Distances from Storage Locations to the Closest Stations; Length of Storage Field = 160, Width of Storage Field = 70, Number of Stations = 30

A8. *High space utilization.* We assume that there are the same number of storage locations and pods in the system. In effect, if the number of pods J is less than the number of storage locations, we assume that we will just utilize the J closest storage locations.

A9. *Ignore stow operations.* We do not model stow operations, by which the pods get replenished with inventory. In reality, pods will also travel to stow stations and receive inventory, and then return to the storage area. The prime justification is to avoid undue complication. There is greater discretion associated with the stowage decisions in terms of deciding when a pod should go to a stow station, what items should be assigned to the pod, and whether or not the visit to a stow station can be combined with a visit to a pick station. We view consideration of stow operations as a topic for future research.

A10. *Ignore pick-up trip.* Each pick tour is comprised of three travel legs for the robotic drive: the first leg is when the drive brings the pod to the station (forward trip); the second leg is when the drive returns the pod to a location (return trip); the final leg is when the drive moves empty to another location to pick up another pod (pick-up trip) and then starts its next tour. In our model, we do not consider the pick-up trip. We expect the distance of the pick-up trips to depend not just on the storage policy but also on the level of drive utilization and the real-time pod selection and prioritization scheme. For instance, if the drives were not heavily utilized, then a drive is likely to idle for a while after a return trip before it is assigned to a forward trip; and at any time the assignment algorithm would presumably try to minimize the length of the pick-up trips.

3.2 Model Preliminaries

By assumption *A2* we define $v_i(x)$ as the probability that demand for item *i* exceeds *x*, given by:

$$v_i(x) = e^{-x/V_i} \tag{1}$$

where V_i is the demand rate for item *i* in some period of time (say half a day). Without loss of generality, we set the integral of V_i to be 1, i.e.

$$\int_0^1 V_i di = 1. \tag{2}$$

That is, we scale the demand so that the expected total demand across all items in the next time period is set to 1. By assumption *A4*, we define Q_i as the inventory quantity for item *i*:

$$Q_i = \int_0^\infty 1(v_i(x) \geq \alpha)dx = -V_i \ln \alpha. \tag{3}$$

Thus, the stock-out probability in the next period for each item equals α, and the total system inventory Q_{inv} is

$$Q_{inv} = \int_0^1 Q_i di = -\ln \alpha. \tag{4}$$

We note from (3) that the ratio $Q_i/V_i = -\ln \alpha$ for all items. This ratio can be interpreted as the "days of cover" for the inventory; that is, the inventory for each item covers the same number of periods of demand. For instance, if the stock-out probability were 0.05, then we hold inventory to cover 3.00 periods of demand.

We define *C* to be the fixed space capacity of a pod and *J* to be the number of pods in the system. By assumption *A5*, we have the following relationship:

$$J = \frac{Q_{inv}}{C} = -\frac{\ln \alpha}{C}. \tag{5}$$

3.3 Evaluation of Storage Policies under Random Stowage

In this section, we assume that the arriving inventory is stowed randomly to the pods; we refer to this as random stowage. We note that random stowage is descriptive of the actual practice in the fulfillment center we consider. Our objective is to evaluate the expected total travel distance for pick tours for the different storage policies.

We define D_i to be the random variable for the number of picks of item *i* in the next time period. Given the above assumptions on demand and on the inventory, we have:

$$\begin{aligned}\Pr\left[D_i > x\right] &= v_i\left(x\right) = e^{-x/V_i} \ \ for \ \ 0 \le x < Q_i \\ \Pr\left[D_i > x\right] &= 0 \ \ for \ \ x \ge Q_i.\end{aligned} \tag{6}$$

We can then find $E[D_i]$ and $Var(D_i)$ as follows:

$$E\left[D_i\right] = V_i\left(1-\alpha\right) \tag{7}$$

$$Var\left(D_i\right) = V_i^2\left(1+2\alpha\ln\alpha-\alpha^2\right). \tag{8}$$

We can use (7) and (8) to characterize the number of picks from a pod. Let U_j^R be a random variable that denotes the expected number of picks for pod *j* under random stowage; we term U_j^R to be the velocity of pod *j* under random stowage. To calculate the mean and variance of U_j^R, we invoke an additional assumption that the random variables $U_j^R, j=1,2,\dots,J$ are independent and identically distributed. We consider this as a reasonable assumption because there are thousands of pods in the system; each pod contains hundreds of items and the inventory units are randomly stowed across the pods by assumption. Thus, the expectation of U_j^R is the same for each *j*, and is $\frac{1}{J}$ of the expected system picks across all items. By the independence assumption, the same is true for the variance of U_j^R. We can then calculate the mean and variance of U_j^R in the following:

$$E[U_j^R] = \frac{\int_0^1 E[D_i]di}{J} = \frac{1-\alpha}{-\ln\alpha}C \tag{9}$$

$$Var(U_j^R) = \frac{\int_0^1 Var(D_i)di}{J} = \frac{v\left(1+2\alpha\ln\alpha-\alpha^2\right)}{-\ln\alpha}C \tag{10}$$

where we define $v = \int_0^1 V_i^2 di$.

With assumption *A6*, the expected number of pod visits to the pick stations is proportional to U_j^R. We can then determine the expected total travel distance for a given storage policy by

$$E\left[d\right] = E\left[\rho\sum_{j=1}^{J} U_j^R d_{s(j)}\right] \tag{11}$$

where $s(j)$ denotes the storage location for pod *j*, $d_{s(j)}$ denotes the travel distance from the location of pod j to the closest station and ρ is a proportionality constant for the expected number of trips per pick. The storage policy determines $s(j)$ for each pod, and for the velocity-based storage policies, this assignment will depend on the velocity of pod j, namely the realization of U_j^R.

With *random storage*, the pod location is chosen randomly, and does not depend on the pod velocities. Without loss of generality, we can then assign pods to storage locations by any arbitrary rule, e.g., $s(j) = j$. We can then evaluate the expected total travel distance for the random storage policy under random stowage by (11):

$$E\left[d_{random}\right] = E\left[\rho\sum_{j=1}^{J} U_j^R hj\right] = \frac{1}{2}\rho h(1-\alpha)(J+1) \tag{12}$$

where we use (9) and $d_j = h \times j$ from *A7*.

We use a simulation study to numerically evaluate the pod travel distance for the velocity-based storage policies under random stowage. We assume each U_j^R follows a Gamma distribution with the mean and variance defined by formula (9) and (10). We assume that we observe the velocity on each pod, and then use this to assign the pods to storage locations, based on the storage policy. Thus, for each iteration of the simulation, we generate a realization of the pod velocities, under the assumption that U_j^R follows a Gamma distribution; we then assign the pods to the storage locations according to the pod velocities and storage policy.

The *full-velocity storage policy* is an idealization that we use as a benchmark. It assigns the pod with the highest velocity to the closest storage location, and so on. We sort the pods by their velocities in a descending order as $U_{(1)}^R, U_{(2)}^R \ldots U_{(J)}^R$ where $U_{(j)}^R \geq U_{(j+1)}^R$. We then use $\sum_{j=1}^{J} U_{(j)}^R hj$ to evaluate the expected total travel distance under the full-velocity storage policy for each realization.

In practice, it is unrealistic to implement the full-velocity storage policy as the system needs to exactly match the rankings of the pod with the rankings of the storage locations. This prevents the system to fully utilize its space capacity. The *class-based storage policy* aims at providing most of the benefits of the full-velocity storage policy while achieving high space utilization.

Here we consider a 2-class and a 3-class storage policy. For the 2-class storage policy, we define the closest $J_1 = \lfloor bJ \rfloor$ storage locations as zone 1 (fast zone) and the remaining $J - J_1$ storage locations as zone 2 (slow zone) where b is the break point. We then place the J_1 highest velocity pods $U^R_{(1)}, U^R_{(2)} \cdots U^R_{(J_1)}$ in class 1, and the remaining pods $U^R_{(J_1+1)}, \cdots U^R_{(J)}$ in class 2. The class 1 (2) pods are randomly stored in zone 1 (2). We can evaluate the travel distance for each class as done for the random assignment policy. For the 3-class storage policy we evaluate the expected total travel distance in a similar way, but now with three pod classes and three storage zones, and with break points $b_1, b_2;\ 0 < b_1 < b_2 < 1$.

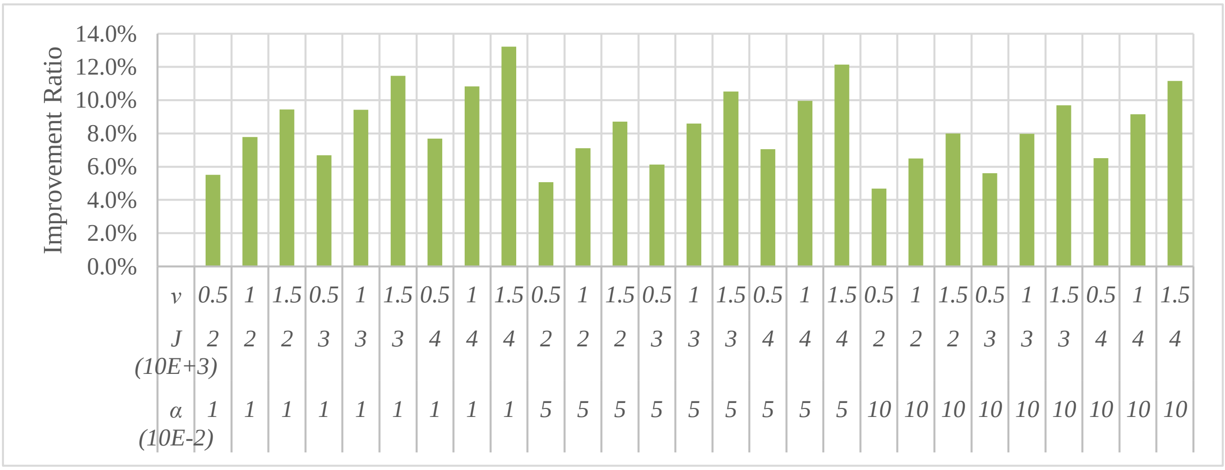
We have simulated these velocity-based storage policies for various test cases that are parameterized by the stock-out rate α, the number of storage pods J, and the second moment of the demand rates ν. For each case, we repeat the simulation for 100 iterations, and record the mean and standard deviation of the travel distance. For these tests, we set the break points: $b = 0.5$ for the 2-class storage policy, and $b_1 = 0.3, b_2 = 0.7$ for the 3-class storage policy. In Table 1 we report the improvement ratios for the full-velocity and class-based storage policies for an illustrative case: $\alpha = 0.05$, $J = 3000$, and $\nu = 1.0$. The improvement ratio is the percentage savings in travel distance relative to the random storage policy. We note that the class-based storage policy captures a significant fraction of the benefits provided by the full-velocity storage policy.

Policy	**Mean**	**Std**
Full-velocity	8.7%	0.3%
2-class	6.1%	0.3%
3-class	7.4%	0.3%

Table 1: Improvement Ratios for $\alpha = 0.05$, $J = 3000$, and $\nu = 1.0$

In Figure 4 we report the improvement ratios for the full-velocity storage policy under different parameter settings. We find that the improvement ratio decreases in α, and increases in J and ν. As explanation, the relative performance of a velocity-based storage policy improves when there is more variation across the pod velocities: when there is more variation, there is more travel savings possible by storing high-velocity pods in the closest locations, low-velocity pods in the furthest locations. The variation in pod velocities increases when there are more pods (greater J), when there is more inventory (smaller α), and when there is greater demand variation across the items (greater ν.)

The improvement ratios for the 2-class and 3-class storage policies show the same behavior as for the full-velocity storage policy in Figure 4. Indeed, we observe that the class-based policies perform quite consistently across all test cases: the 2-class and 3-class storage policies capture around 71% and 86% of the improvement ratio of the full-velocity storage policy, respectively, in each test case.



v	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5
J (10E+3)	2	2	2	3	3	3	4	4	4	2	2	2	3	3	3	4	4	4	2	2	2	3	3	3	4	4	4
α (10E-2)	1	1	1	1	1	1	1	1	1	5	5	5	5	5	5	5	5	5	10	10	10	10	10	10	10	10	10

Figure 4: Improvement Ratios of Applying Full-velocity Storage under Random Stowage

3.4 Evaluate Storage Policies under Velocity-based Stowage

In the prior section we assume that units are stowed randomly across the pods. In this section we consider a different stowage policy, whereby units are stowed to the pods based on their velocities. For example, we might stow high (low) velocity units to create high (low) velocity pods; if so, then it may be possible for even more benefit from a velocity-based storage policy.

The velocity-based stowage refers to an idealized situation in which we can strictly rank and stow units by their velocities. In particular, we invoke assumption *A3*, namely that we can order the inventory according to its likelihood of being demanded, as given by the $v_i(x)$ functions. We then assume that we can assign the inventory to the pods according to the velocity measure. We define thresholds $\{y_j, j=1,2,...,J\}$, with $y_1 = 1$ to demark the boundaries between the pods; that is, pod *j* contains all inventory that has a velocity v_i value between y_{j+1} and y_j and thus the pod holds all units whose probability of being picked is between y_{j+1} and y_j, where $y_{j+1} < y_j$. We determine the thresholds so that the inventory assigned to each pod matches its storage capacity. Hence, we determine the thresholds from solving the following equations:

$$\int_0^1 \int_0^{Q_i} 1\left(v_i(x) \geq y_{j+1}\right) dxdi - \int_0^1 \int_0^{Q_i} 1\left(v_i(x) \geq y_j\right) dxdi = C, \quad j=1,2,...,J \tag{13}$$

with $y_1 = 1$. The first integral on the left side gives the total inventory with a velocity value greater than y_{j+1}; the second integral is the total inventory with velocity greater than y_j. The difference is the inventory that would ideally be stowed on pod *j*; and the equation sets the inventory quantity to the pod capacity.

We note that

$$\int_0^1 \int_0^{Q_i} 1\left(v_i(x) \geq y_j\right) dxdi = -\ln\left(y_j\right).$$

We can then re-express (13) as

$$-\ln\left(y_{j+1}\right) + \ln\left(y_j\right) = C, \quad j = 1, 2, ..., J.$$

Given the boundary condition $y_1 = 1$, we have an explicit solution for (13), given by:

$$y_j = e^{-(j-1)C}, \quad j = 1, ..., J+1. \tag{14}$$

Intuitively, we store *C* units that have the velocities between 1 and e^{-C} on pod 1, and the following *C* units that have the velocities between e^{-C} and e^{-2C} on pod 2, and so on. The last pod *J* contains *C* units that have velocities between $e^{-(J-1)C}$ and e^{-JC}. We note that $e^{-JC} = \alpha$ which confirms that this partition of the inventory accounts for all units.

We now use this specification of the pods to determine the expected picks on pod *j* for velocity-based stowage, denoted by U_j^V. We can calculate U_j^V as the integral of the expected picks over all items from pod *j*:

$$U_j^V = \int_0^1 \int_0^{Q_i} 1\left(y_{j+1} \leq v_i\left(x\right) \leq y_j\right) v_i\left(x\right) dxdi = \alpha^{\frac{j-1}{J}} - \alpha^{\frac{j}{J}}. \tag{15}$$

We note that by summing these values we obtain the expected total picks in the next time period; that is, $\sum_{j=1}^{J} U_j^V = 1 - \alpha$. We also observe that these pod velocities decrease geometrically.

3.4.1 Full-velocity Storage Policy under Velocity-based Stowage

We now consider the full-velocity storage policy to the specification of pods in (15); that is, we assume that pod *j* is stored in the *j*th closest location. This is an idealization and will act as a benchmark. We can calculate the expected total travel distance as

$$E\left[d_{Velocity}\right] = \rho \sum_{j=1}^{J} U_j^V hj = \rho h \left(\frac{1-\alpha}{1-\alpha^{1/J}} - J\alpha \right). \tag{16}$$

We measure the improvement relative to the base case (12) of a random storage policy under random stowage. We find the improvement ratio to be

$$R_{Velocity} = \frac{E\left[d_{Random}\right]-E\left[d_{Velocity}\right]}{E\left[d_{Random}\right]} = \frac{2}{J+1}\left(\frac{J-1}{1-\alpha}-\frac{\alpha^{1/J}-\alpha}{\left(1-\alpha^{1/J}\right)\left(1-\alpha\right)}-\frac{J-1}{2}\right). \tag{17}$$

We show in the appendix that the expression (17) is positive for any $\alpha \in (0,1)$ and $J > 0$, and that it decreases in α for $\alpha \in (0,1)$. To get further insight, we take the limit of (17) as $J \to \infty$:

$$R_{Velocity}\Big|_{J\to\infty} = \frac{1+\alpha}{1-\alpha}+\frac{2}{\ln\alpha}. \tag{18}$$

We finally note that the improvement ratio does not depend on the skewness of the demand rates (v.) This is because with the velocity-based stowage policy, the full-velocity storage policy directly takes advantage of the velocity difference among the inventory units. The skewness of the demand rates does not provide additional benefits to the storage decision.

3.4.2 Class-based Storage under Velocity-based Stowage

In practice, the full-velocity storage policy is not realistic. Hence, we again consider the more realistic class-based storage policy with velocity-based stowage.

We first consider a 2-class storage policy where the storage area is divided into a fast and a slow storage zone, and the pods are categorized into fast-moving and slow-moving classes. We again let *b* be the break point that divides the two zones, i.e. *b* represents the percentage of the storage space for zone 1. Then zone 1 contains the $J_1 = \lfloor bJ \rfloor$ storage locations that are closest to the stations and zone 2 contains the remaining $J - J_1$ storage locations. We then place in class 1 the J_1 highest velocity pods, namely pods $j = 1, 2, \ldots J_1$. The remaining $J - J_1$ pods go in class 2.

We determine the expected travel distance for this policy by first computing the travel distance for each class of pods. Within class 1, the storage of pods is random; hence the expected travel distance is given by:

$$E\left[d_{2class}^{1}\right] = \frac{h\left(J_1+1\right)}{2}\rho\sum_{j=1}^{J_1}U_j^V = \frac{h\left(J_1+1\right)}{2}\rho\left(1-\alpha^{\frac{J_1}{J}}\right)$$

where $h\left(J_1+1\right)/2$ is the average travel distance to the J_1 storage locations in zone 1. For class 2, the average travel distance is $h\left(J+J_1+1\right)/2$. We can then use this to find the expected travel distance for the pods in class 2:

$$E\left[d_{2class}^2\right]=\frac{h\left(J+J_1+1\right)}{2}\rho\sum_{j=J_1+1}^{J}U_j^V=\frac{h\left(J+J_1+1\right)}{2}\rho\left(\alpha^{\frac{J_1}{J}}-\alpha\right).$$

We now combine these two expressions to get the expected total travel distance for the 2-class system:

$$\begin{aligned}E\left[d_{2class}\right]&=E\left[d_{2class}^1\right]+E\left[d_{2class}^2\right]\\&=\frac{h\left(J_1+1\right)}{2}\rho\left(1-\alpha^{\frac{J_1}{J}}\right)+\frac{h\left(J+J_1+1\right)}{2}\rho\left(\alpha^{\frac{J_1}{J}}-\alpha\right).\end{aligned} \tag{19}$$

We calculate the improvement ratio for the 2-class storage policy under the velocity-based stowage against the base case where random storage and random stowage is assumed:

$$R_{2class}=\frac{E\left[d_{Random}\right]-E\left[d_{2class}\right]}{E\left[d_{Random}\right]}=\frac{J}{J+1}\left(\frac{1-\alpha^{\frac{J_1}{J}}}{1-\alpha}-\frac{J_1}{J}\right). \tag{20}$$

When *J* goes to infinity and holding the breakpoint *b* fixed, we have

$$R_{2class}\big|_{J\to\infty}=\frac{1-\alpha^b}{1-\alpha}-b. \tag{21}$$

We show in the appendix that the improvement ratio in (21) is always positive for the 2-class storage policy. Furthermore, the optimal break point is achieved when

$$b^*=\log_\alpha\left(\frac{1-\alpha}{-\ln\alpha}\right). \tag{22}$$

For the 3-class storage policy, let b_1 and b_2 be the break points that divide the storage locations into three zones where zone 1 contains $J_1=\lfloor bJ\rfloor$ closest storage locations, zone 2 contains the $J_2=\lfloor(b_2-b_1)J\rfloor$ next closest storage locations, and zone 3 contains the remaining $J-J_1-J_2$ storage locations. Similar to the 2-class case, we associate the J_1 highest velocity pods with class 1, the next J_2-J_1 pods with class 2, and the remaining pods with class 3. We construct the expected travel distance for the 3-class storage policy in the same way as done for the 2-class policy: we find the expected travel for each class of pods, using the average travel distance for the assigned zone:

$$\begin{aligned}E\left[d_{3class}^1\right]&=h\frac{J_1+1}{2}\rho\sum_{j=1}^{J_1}U_j^V=h\frac{J_1+1}{2}\rho\left(1-\alpha^{\frac{J_1}{J}}\right)\\E\left[d_{3class}^2\right]&=h\frac{J_1+J_2+1}{2}\rho\sum_{j=J_1+1}^{J_2}U_j^V=h\frac{J_1+J_2+1}{2}\rho\left(\alpha^{\frac{J_1}{J}}-\alpha^{\frac{J_2}{J}}\right)\\E\left[d_{3class}^3\right]&=h\frac{J_2+J+1}{2}\rho\sum_{j=J_2+1}^{J}U_j^V=h\frac{J_2+J+1}{2}\rho\left(\alpha^{\frac{J_2}{J}}-\alpha\right).\end{aligned} \tag{23}$$

The expected total travel distance for the 3-class system is then the sum of the above:

$$E\left[d_{3class}\right]=\frac{h\left(J_1+1\right)}{2}\rho\left(1-\alpha^{\frac{J_1}{J}}\right)+\frac{h\left(J_1+J_2+1\right)}{2}\rho\left(\alpha^{\frac{J_1}{J}}-\alpha^{\frac{J_2}{J}}\right)+\frac{h\left(J_2+J+1\right)}{2}\rho\left(\alpha^{\frac{J_2}{J}}-\alpha\right).\quad(24)$$

We find the improvement ratio relative to the base case to be

$$R_{3class}=\frac{E\left[d_{Random}\right]-E\left[d_{3class}\right]}{E\left[d_{Random}\right]}=\frac{J\left(1-\alpha^{\frac{J_2}{J}}\right)-J_1\left(1-\alpha^{\frac{J_2}{J}}\right)-J_2\left(\alpha^{\frac{J_1}{J}}-\alpha\right)}{\left(1-\alpha\right)\left(J+1\right)}.\quad(25)$$

If we let J go to infinity, the improvement ratio becomes

$$R_{3class}\Big|_{J\to\infty}=\frac{\left(1-b_1\right)\left(1-\alpha^{b_2}\right)+b_2\left(\alpha-\alpha^{b_1}\right)}{1-\alpha}.\quad(26)$$

Finally, to obtain the optimal setting for the break points, we take the derivative of (26) with respect to each of the break points; this gives the following two equations to solve:

$$\begin{aligned}\alpha^{b_2}-1&=b_2\alpha^{b_1}\ln\alpha\\ \alpha-\alpha^{b_1}&=\left(1-b_1\right)\alpha^{b_2}\ln\alpha.\end{aligned}\quad(27)$$

By taking the second derivatives, we can verify that the improvement ratio is concave in b_1 and b_2 and hence by solving the system of equations (27) we can obtain the global maximum solution. We solve (27) with a simple grid search method.

This analysis can be extended for more than three classes; however, the benefits decline rapidly after three classes as will be shown in the next section.

3.5 Numerical Example and Sensitivity Analysis

In this section we will use numerical examples to quantify the improvement from the velocity-based storage policies with velocity-based stowage relative to the benchmark of random storage with random stowage. For these numerical results, we use the optimal break points for the 2-class and 3-class storage policies, given by (22) and (27).

We first show how the improvement ratio depends on the number of pods J for a fixed stock-out rate. In Figure 5, we show that the improvement ratio converges very quickly to the limits, for stock-out rate $\alpha = 0.1$. This convergence behavior is the same for other stock-out rates. The number of pods in a typical semi-automated storage system is on the order of thousands; hence, we can safely use the asymptotic analytical result to evaluate the improvement ratios.

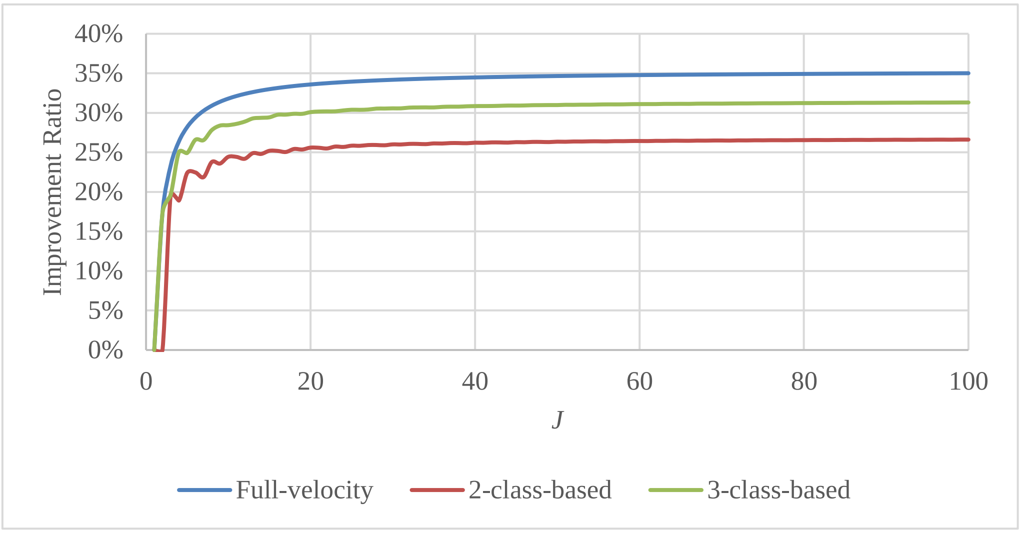


Figure 5: Convergence of the Improvement Ratios to the Limits ($\alpha = 0.1$)

We next examine how the improvement ratio depends on the stock-out rate α. We plot the asymptotic improvement ratios of the velocity-based policies as functions of α in Figure 6. We observe that the improvement ratio decreases in the stock-out rate. Furthermore, we observe that the 2-class storage policy consistently captures 76%-78% of the benefits of the full-velocity storage policy; the 3-class storage policy consistently captures 89%-91%. Interestingly, Hausman, et al. (1976) have a very similar finding: they find that the 2-class and 3-class storage policies capture approximately 70% and 85% of the full-turnover-based storage policy in the context of an AS/RS.

The optimal break points for the class-based storage policies are increasing in the stock-out rate. When $\alpha = 0.01$, the optimal break points are $b = 0.33$ and $b_1 = 0.20, b_2 = 0.49$. When $\alpha = 0.2$, the optimal break points are $b = 0.43$ and $b_1 = 0.28, b_2 = 0.60$. An explanation of this is that when there is more inventory in the system (smaller α), we need to assign more space to the slow-moving items which implies smaller fast-moving zones (smaller b values).

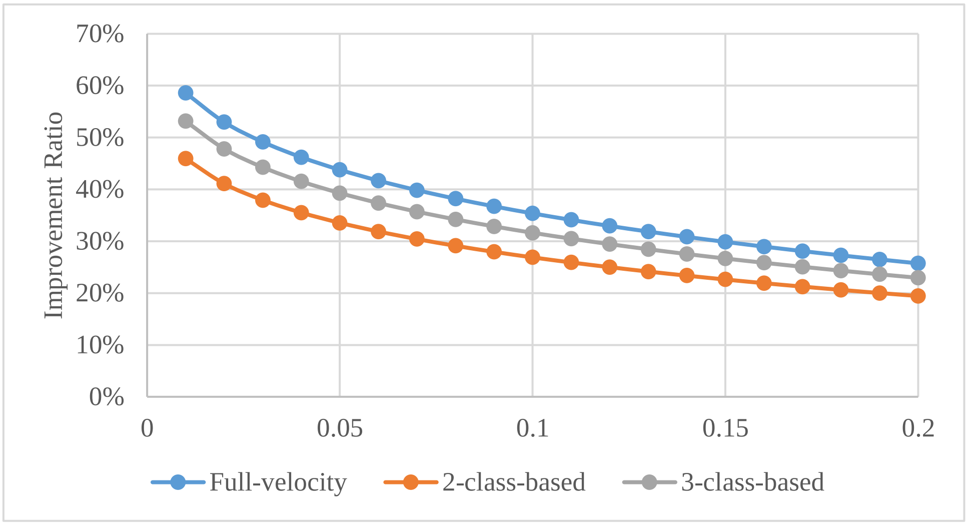


Figure 6: Improvement ratios for full-velocity and the optimal 2-class and 3-class storage policies under different α when J goes to infinity

Finally, we compare the performance of the storage policies under random and velocity-based stowage with the illustrative example from Table 1, namely $\alpha = 0.05$, $J = 3000$, and v=1.0. For this comparison, we use the corresponding optimal break points for the 2-class and 3-class policy, namely b=0.38, and $b_1 = 0.24$, $b_2 = 0.55$. We observe from Table 1 that with random stowage, the full-velocity and class-based storage policies provide an improvement on travel distance of 6%-9%. With velocity-based stowage, we find in Table 2 that the savings can be much greater, from 33% to 43%. In both cases, the 2-class and 3-class storage policies capture most of the benefit from the full-velocity storage policy. The performance difference between velocity-based and random stowage is also noteworthy: it suggests the potential from smarter stowage policies that would attempt to create more variation across the pods in terms of their pick velocities.

Improvement Ratios	**Random Stowage**	**Velocity-based Stowage**
Random storage	-	0%
Full-velocity storage	8.6%	43.3%
2-Class storage	5.9%	33.2%
3-Class-storage	7.3%	38.9%

Table 2: Comparison of the Improvement Ratios for an Example where $\alpha = 0.05$, $J = 3000$, v=1.0, b=0.38, $b_1 = 0.24$, $b_2 = 0.55$

4 Simulation

In this section, we report on a simulation study to validate the findings from the fluid model. The primary intent of the simulation is to examine the effectiveness of a class-based storage policy in a large scale fulfillment center with real industrial data, in comparison to the actual storage policy. In particular we use the simulation to test a practical way to determine the pod velocities. We also perform sensitivity analyses on the impact of both different zoning strategies and the space utilization levels. Our goal is to identify the range of conditions and parameters for which the class-based storage policies remains effective.

We first describe the context and data for the simulation. We then review the assumptions and the settings for the simulation. We next present the base case results from the simulation for a 2-class and 3-class storage policies, followed by a sensitivity analysis and some conclusions.

4.1 Simulation Context and Data

The simulation study is based on one storage zone in a semi-automated storage system, very similar to Figure 1. We obtained detailed data on the operations of this storage zone for a ten-day period. The data is representative of the scale in a facility. This storage zone initially holds more than 500,000 sku's and over 2,000,000 inventory units. The storage zone fulfills more than 50,000 units per day, and receives and stows about the same amount of inventory units over this ten-day period. There are more than 5000 pods in storage. A typical pod holds 300-500 units of inventory that are spread over 100 to 200 sku's. The inventory of each sku can be stored on several pods, as shown in Figure 7 for a typical inventory snapshot.

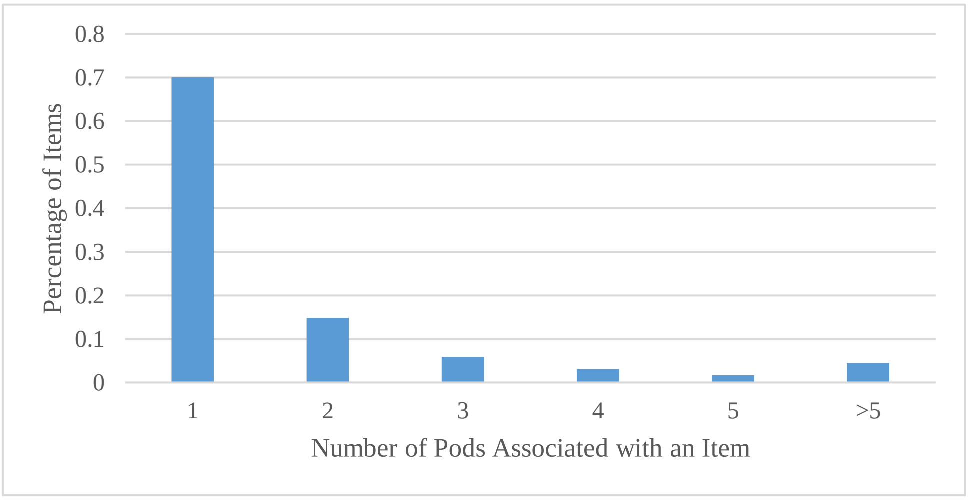


Figure 7: Histogram of the Number of Pods Associated with an Item for a Typical Inventory Snapshot

A central control system assigns (or “drops”) customer orders to the storage zone continuously over the day. The assigned orders (or requests) join a backlog of requests to be picked; associated with each request is a due time that corresponds to a shipment time. The picking algorithm needs to make sure that those due times are met and that the aggregate picking requests are within the picking capacity for the storage zone.

A large percentage of the actual orders are for multiple sku’s, so called multi-item orders. In this system, unlike that described in Enright and Wurman (2011) and Lamballais et al. (2017a), the assembly of a multi-item order does not occur at the pick stations. Rather each unit of a multi-item order can be picked by any station. After picking the units get conveyed to a sortation stage, at which the orders are assembled and boxed. In order for sortation to operate efficiently, the operating system will pick all of the units for an order within a tight time window. But the individual items in each multi-item order can (and will) be picked at different stations.

4.2 Simulation Framework

We have designed the simulation to model the pod travel associated with picking events. We set the picking decisions to be exactly the same as in the actual data from the ten-day period. We then make the storage decisions, based on whatever storage policy is being tested. We thus are able to compare the performance of the different storage policies, effectively with all else being equal.

We describe the simulation in terms of the underlying assumptions. We then explain how we set the velocity measure for each pod and how we implement class-based storage. We provide an overview of the simulation logic and the parameter settings for our simulation experiments.

4.2.1 Assumptions

A1. *Stow Operations.* We do not simulate the pod movements that are associated with the stowing transactions. We do, however, need to assume in the simulation that stowage will happen, to replenish the inventory in the storage zone. To do this, we will simply increment the inventory counts on the pods according to the actual stowing record, at (roughly) the actual time. For simplicity, we assume the inventory is stowed at the beginning of each hour. With this assumption, we only consider the storage decision following the picking operations. We argue that for the class-based storage policies, as long as we follow the same logic for returning the pods back to the storage zones for the stowing operation as for the picking operation, it should have a similar effect on the drive travel times.[1]

A2. *Specification of Picks*. We assume the picking decisions are given, based on the actual pick records. At the beginning of each hour we use these pick records to identify which picks were done from which pods in the next few hours. We then construct two pod-trip sets at the beginning of each hour: an urgent trip set which is the set of pod trips with picks that are due within the next four hours; and a normal trip set which contains the remaining trips with less urgent picks.

A3. *Assignment of Pods to Stations*. Within the simulation, at every time instant each pick station has a virtual queue of pods that have been assigned to the station for picking but have yet to be picked. These pods are either in transit to the station, or in queue at the station. We control the size of the virtual queue with a threshold parameter. Whenever the station's virtual queue drops below the threshold, the pick station requests (or pulls) a new pod from a pod-trip set. These pulls are made first from the urgent trip set, and then from the normal trip set; within the chosen trip set, the simulation assigns the pod that is closest to the station. This assumption is based on the current practice in the storage system. The intent is to assure that each station has work and is never starved. The threshold parameter is an input to the system. When the pod is at the pick station, we execute the picks according to the pick records.

A4. *Deterministic travel times.* The pod travel times are deterministic and determined by the distance between the origin and destination, according to a Manhattan metric. As such, we ignore any impact on travel time due to congestion, and ignore the fact that some storage locations are less accessible.

A5. *Deterministic unit pick time.* The unit pick time at the pick station is deterministic. This is a reasonable assumption as the variation of picking an item is small in practice.

A6. *Constant picking capacity*. We assume there is a fixed number of active picking stations over the course of the simulation. In practice, the number of active stations can vary across work shifts and there are break times in a day. Keeping the number of stations constant simplifies the simulation. As our focus is on the storage policies, we expect that varying the picking capacity will have no effect on the performance of a storage policy.

A7. *Forward and Return Trip Only.* Similar to assumption *A10* for the model development, we

[1] In the appendix we relax this assumption and modify the simulation to account for the travel associated with the stow trips. The main findings with respect to the effectiveness of class-based storage policies remain the same.

consider only the forward and return trips in the simulation. We do not model the pick-up trips. In effect we assume there are sufficient drives so that a drive is available whenever a station pulls a pod.

4.2.2 Pod Velocity Measures

To implement a class-based storage policy, we need some measure of the velocity of each pod. Ideally this measure would predict whether or not we pick from the pod in the near future. We consider two factors to develop the velocity measure, namely the future demand and the inventory distribution of the items. For the first factor, we model the future demand in our simulation by the existing order backlog; that is, our prediction as to whether a pod is likely to be requested or not is based on whether it contains units that can satisfy any orders in the current backlog. Typically the order backlog provides a good estimate of what will be picked in the near future, i.e., in the next 12 to 24 hours. We can supplement the order backlog with a forecast of new orders, if that can be done reliably. The second factor, the inventory distribution of the items, depends on the number of pods that hold each item; essentially a unit of inventory for an item contributes less to the velocity of a particular pod if the item is also stowed on many other pods.

For specifying the velocity measure, we assume that each unit of an item in the system has the same likelihood of getting picked to meet a unit demand for the item. Based on this assumption, we define at any time the velocity of pod *j* as the expected picks from the pod in the next *t* time periods:

$$V_j = \sum_{i \in I(j)} \min\left(\frac{a_{ij}}{\sum_k a_{ik}} \sum_{u \le t} b_{iu}, a_{ij} \right) \tag{28}$$

where b_{iu} is the current demand backlog of item *i* that will be due at time *u*; a_{ij} is the current inventory of item *i* on pod *j*; and *I(j)* is the set of items contained in pod *j*. The first term in the minimization is the expected fraction of demand to be allocated to the pod in the next *t* time periods. If this were greater than the available inventory on pod *j*, then we only allocate a_{ij} picks to the pod, to ensure we do not over-count the velocity of an item. The pod velocity is the sum of the expected picks over all of the items on the pod.

4.2.3 Storage Policies

We use the simulation to evaluate the class-based storage policy and the closest-open-location storage policy. The closest-open-location storage policy returns the pod to the closest available storage location relative to the pick station, and is a proxy for the storage policy used in current practice.

For the class-based storage policy, when a pod completes a pick operation, we assign the pod to a class according to its velocity given in (28). For a 2-class system, we assign the pod to either class 1 or 2, depending on whether the pod velocity is higher or lower than a velocity threshold; for a 3-class system, the assignment is similar but now with two thresholds. We update these thresholds every minute within the simulation, so as to best match the distribution of pod velocities with the available storage space. We discuss the details of how we make these adjustments in the section on Parameter Setting.

In theory, we specify the velocity zones so that the storage locations in the higher-velocity zone are closer to the pick stations, compared to the storage locations in a lower-velocity zone. However, since we have many pick stations that are distributed around the boundary of the storage field (see Figure 1), this specification is ill-defined. Hence we need to explain how we implement class-based storage in the simulation.

We associate a distance to each storage location, where that distance is the travel distance to the closest station. We then determine the velocity zones based on a ranking of the locations by these distances. For instance, if velocity zone 1 has size N_1, then the N_1 highest-ranked locations are assigned to this zone. If the next zone has size N_2, then the next N_2 ranked locations go into this zone, and so on.

We demonstrate this zoning strategy in a simple two-station example in Figure 8. The stations are the black boxes at the bottom of each figure. Each cell is a storage location and the number within the cell is the distance to the nearest station. The colors demark the zones.

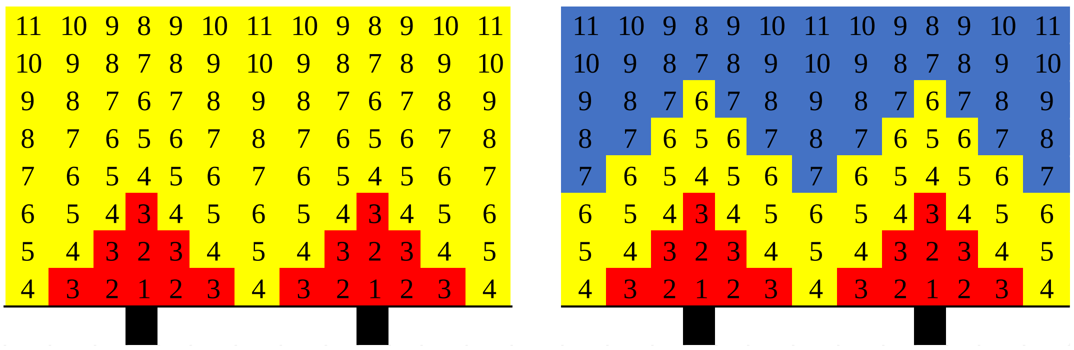


Figure 8: Illustrations for the 2-class and 3-class velocity zones for a two-station system

We fix the velocity zones at the start of the simulation. Thus, it may not always be possible to store a pod in its desired zone. For a 2-class system we use the following assignment rules:

- We always assign a class 1 pod to the closest available storage location in the entire storage field from the current pick station. Hence we may store the pod in velocity zone 2, if the closest available location in velocity zone 2 is closer than the closest available location in velocity zone 1.

- We assign a class 2 pod to the closest storage location in velocity zone 2 that is open. If velocity zone 2 is full, we assign the pod to the closest available storage location in zone 1.

The assignment rules for a 3-class system are similar but slightly more complicated:

- We always assign a class 1 pod to the closest available storage location in the storage field.
- We assign a class 2 pod to the closest available storage location in zone 2 that is within a given distance threshold from the pick station. The intent is to avoid assigning the pod to a very distant zone 2 storage location. If this is not possible, we then assign the pod to the closest available storage location considering both zone 2 and zone 3. If both zones are completely full, we assign the pod to the closest available storage location in zone 1.
- We assign a class 3 pod to the closest available storage location in zone 3. If this zone is full, we assign the pod to the closest available storage location in the storage field.

4.2.4 Parameter Settings

We indicate the parameter settings for the base case in the simulation tests as follows:

Size of Velocity Zone. In the base case, we set the break points such that for the 2-class system, approximately 40% of the storage locations are associated with zone 1; for the 3-class system, approximately 20% of the storage locations are associated with zone 1, and 30% of the storage locations are associated with zone 2. We show the storage locations and their corresponding storage zones for 2-class system and 3-class system in Figure 9 and 10 respectively. We note that there are some empty space (shown as blank in the figures) in the storage field that cannot be used for storage.

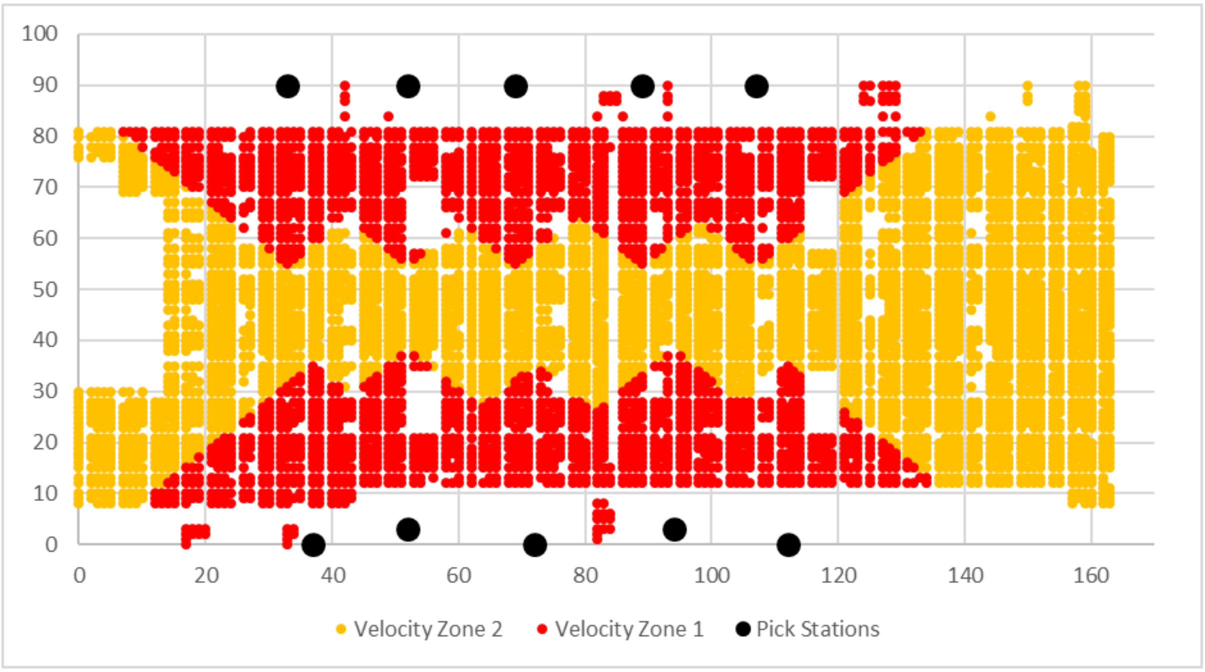


Figure 9: Storage and Station Locations by Velocity Zones for the 2-class System, 40% Zone 1 (red) and 60% Zone 2 (yellow)

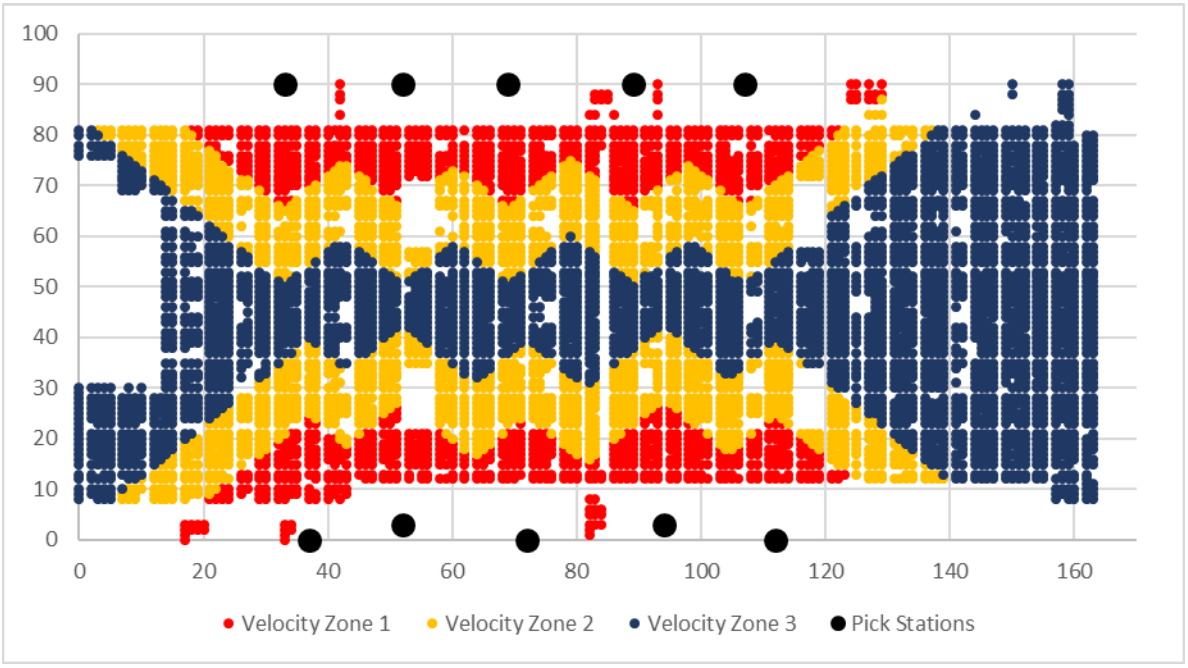


Figure 10: Storage and Station Locations by Velocity Zones for the 3-class System, 20% Zone 1 (red), 30% Zone 2 (yellow), and 50% Zone 3 (blue)

Resource Utilization. In the base case we set 10 pick stations around the storage zone, as shown in Figures 9 and 10. This provides a constant picking capacity of 3600 units per hour where each pick station can pick 6 units per minute.

Space Utilization. The space utilization is 99% in the base case. That is, the ratio of storage pods to storage locations is 0.99. The actual space utilization is much lower (around 95%) as many pods are at a pick station or in transit at any point in time.

Velocity Thresholds. For the 2-class policy let N_i denote the number of storage locations in zone *i*, *i* = 1, 2; let $E_i(t)$ denote the number of empty storage locations at time *t* in zone *i*, *i* = 1, 2. At time *t*, we set the threshold percentile:

$$TP = (1-\gamma)\frac{N_1}{N_1+N_2} + \gamma\frac{E_1(t)}{E_1(t)+E_2(t)} \tag{29}$$

where γ is the parameter $(0<\gamma<1)$ to adjust the threshold according to the distribution of the empty storage locations. We then rank the entire population of the pods in a descending order of their velocities and use the velocity of the pod at the *TP* quantile to be the velocity threshold. That is, *TP* percent of the pods have velocities greater than the velocity threshold (regardless of their storage location). We then use this velocity threshold to decide whether a pod, which has just completed picking and is to return to the storage field, is class 1 or class 2 at that point in time.

The intent of this method is to adjust dynamically the threshold based on the current storage occupancy. When there is a higher fraction of empty storage locations in zone 1 than the target ratio (i.e., $\frac{N_1}{N_1+N_2}$), then we increase *TP*, which results in assigning more pods to zone 1. As a result, the number of empty storage locations in zone 1 will decrease. We interpret γ as a balancing parameter that controls how aggressively we try to keep the fraction of empty storage locations in zone 1 close to the target ratio. We set γ to be 0.5 in the base case. For the 3-class policy, we use two equations of the same form as (29), one for zone 1 and the other for zone 2. For both equations, we set $\gamma = 0.5$.

Virtual Queue Size. We set the virtual queue threshold to 10 for each pick station, limiting the maximum number of pods that can be waiting at a station or in transit to the station.

Initiation of Simulation. At the start of the simulation we assign each pod to a storage location. We do this by random assignment, as we did not have information on the actual starting locations.

4.2.5 Simulation Logic

We provide here an overview of the architecture and logic of the simulation.

Step 0: Load information related to the inventory pods, storage locations, work stations, and current order backlogs

At the beginning of each *hour*

Step 1: Update pod velocity

Step 2: Update the pod-trip sets. We add a pod trip to the urgent trip set if the pod fulfills pick requests with due times within the next four hours; we add a pod trip to the normal trip set if it fulfills pick requests with due times within the next four to 24 hours.

Step 3: Replenish the pods instantly with the inventory that was received and stowed in that hour

At the beginning of each *minute*

Step 4: Update velocity thresholds

Step 5: Update backlog with new demand arrivals

At each *second*

Step 6: Update pod status for the pods that are in a forward trip headed to a pick station, or in queue or being picked at the station, or in a return trip to the storage location.

Step 7: At each station, continue pick activity, as indicated on the pick records

Step 8: At each station, if there is a slot open in its virtual queue, determine the closest pod in either the urgent or normal pod set to send to the pick station

Step 9: At each station, if a pod completes its pick operation, then make storage decision; the simulation will reevaluate the velocity of the pod and decide to which storage location that it will be returned.

4.3 Simulation Results

In this section, we report the simulation results for the closest-open-location storage policy, the 2-class and 3-class storage policy for the base case. The key performance measure is the total distance traveled by the storage pods and drives. We simulate ten days of activity, and report the travel distance from day 2 through day 9. We do not report the travel times on the first and last days, to avoid any start-up or end-of-horizon effects.

In Table 3, we report the normalized travel distance for each storage policy. The results are normalized by dividing each measure by the total forward-trip distance for the closest-open-location storage policy. The 2-class-based policy achieves an 8.9% reduction on travel distance; this increases to 9.8% for the 3-class policy.

	Closest-open-location	**2-class**	**3-class**
Forward Trip Distance	1.00	0.94	0.93
Return Trip Distance	1.04	0.93	0.91
Total Travel Distance	2.04	1.86	1.84
Improvement Ratio		8.9%	9.8%

Table 3: Performance Comparison for the Closest-open-location, the 2-class, and the 3-class Storage Policy under the Base Case Scenario (10 Stations, 1% Empty Space, $\alpha = 0.5$)

To get some insight into the improvement from the class-based policies, we report on some other measures from the simulations. We first show the changes of the velocity thresholds over the course of the simulation for the 2-class in Figure 11 (the results for the 3-class are very similar). We recall that the velocity of a pod estimates the expected number of units to be picked from the pod in the next few time periods. There are two drivers for the variation in the velocity threshold. One is the variation in the size of the demand backlog. When there are more units waiting to be picked, the pods' velocities will increase and the velocity thresholds also increase accordingly. The other driver is the adjustment mechanism given by (29). We adjust the velocity thresholds at the beginning of every minute in order to maintain the balance of the empty storage locations across the velocity zones.

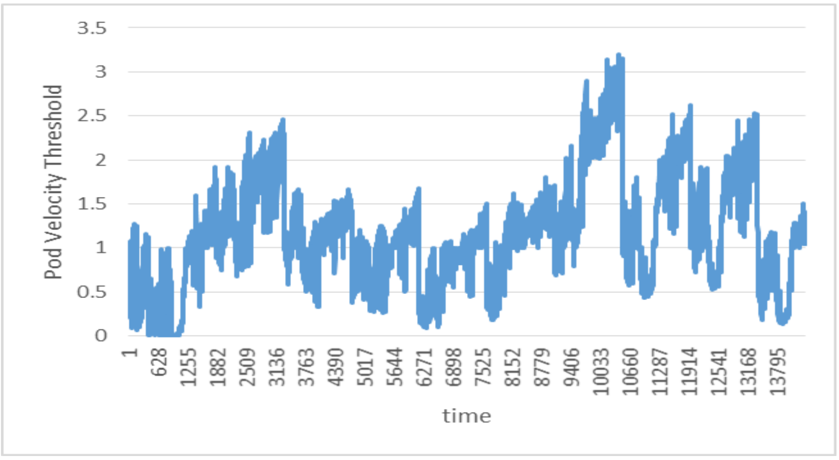


Figure 11: Velocity Threshold by Time for the 2-class System

We report the distribution of the open storage locations over time for the 2-class system in Figure 12. The figure is a time sequence of snapshots of the open storage locations for each velocity zone taken at the beginning of each minute. We observe that the system is busy for most of the time in the simulation as the number of open storage locations is equal to its upper limit for the majority of the time. (The upper limit equals the number of excess storage locations, plus the maximum virtual queue for the pick stations.) We also see that the policy tries to maintain open locations in both zones, so as to be able to return any pod to its correct zone.

In Figure 13, we plot the corresponding distribution of open locations for the closest-open-location storage policy, where we use the zone categorization from the 2-class policy. We see virtually all of the open locations would fall in zone 2; and that for much of the time there are no close open locations.

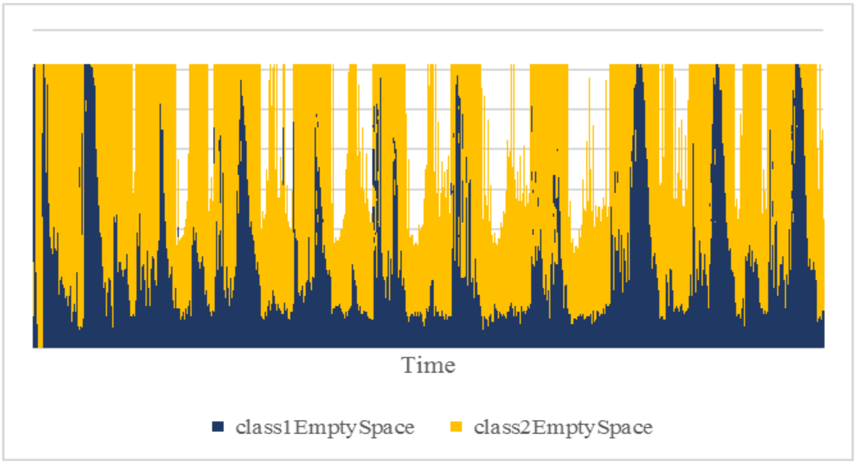


Figure 12: Distribution of the Open Storage Space by Time for the 2-class System

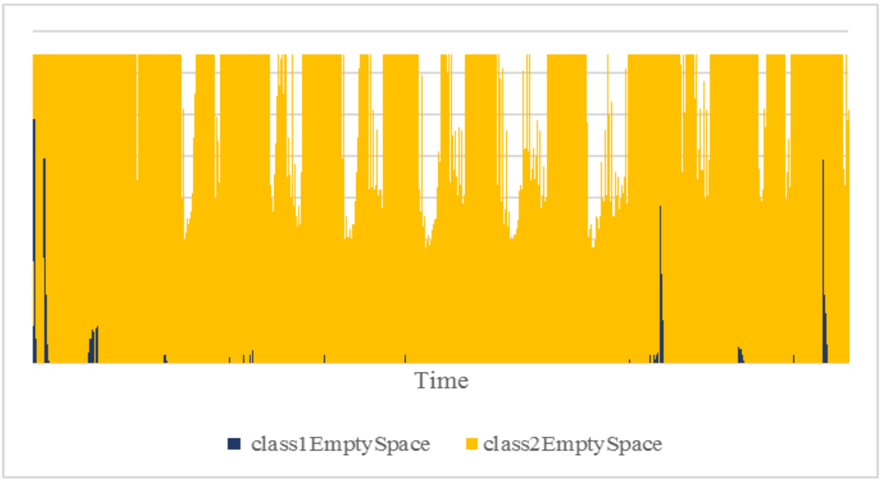


Figure 13: Distribution of the Open Storage Space by Time for the Closest-open-location Storage Policy (assume 2-class system zoning code)

We report the actual zone assignments in Table 4 and 5 for each class of pods for the class-based policies. We see that in each case, a very high percent of the pods get stored in the correct zone.

	Velocity Zone 1	**Velocity Zone 2**
Class 1 Pods	97.9%	2.1%
Class 2 Pods	4.1%	95.9%

Table 4: Storage Assignment for Each Class for the 2-class System

	Velocity Zone 1	**Velocity Zone 2**	**Velocity Zone 3**
Class 1 Pods	93.6%	6.1%	0.3%
Class 2 Pods	0.1%	95.8%	4.0%
Class 3 Pods	0.5%	6.3%	93.2%

Table 5: Storage Assignment for Each Class for the 3-class System

We also examine a snapshot of the location of pods according to their velocity classes at various points in time. For instance at the end of the first day of the simulation, we find that for the 2-class-based policy 80% of the fast pods (class 1 pods) are stored in zone 1, and 77% of the slow pods (class 2 pods) are stored in zone 2. These percentages are quite stable over the simulation, after the first-day start up.

Finally, as expected, the higher velocity storage locations turned over more quickly for the class-based policies. We define the turnover rate of a zone as the total number of pod trips from the zone divided by the number of storage locations in the zone. For the 2-class system, the turnover rate of zone 1 is around 1.6 times that of zone 2. For the 3-class system, the turnover rate of zone 1 is around 1.5 times that of zone 2, and 2.4 times the turnover rate of zone 3.

4.4 Sensitivity Analysis

We perform a sensitivity analysis with respect to three key input parameters in the simulation, namely the distance thresholds for setting the zones, the number of open storage locations, and the parameter γ for adjusting the velocity threshold. We also ran the simulation under an improved picking algorithm which attempts to maximize the number of unit picks per pod visit at the pick station. We summarize the key findings here; the details can be found in Yuan (2016). We also modified the simulation to account for the travel for stow trips; we report the results in the Appendix.

Zoning Strategy. We reran the simulation for different settings for the size of the velocity zones for the class-based policies. For the 2-class system, the improvement percentage is fairly flat as we vary the size of zone 1 from 30% to 80%. There is a marked reduction in the improvement if zone 1 were reduced below 30%. For the 3-class system, the improvement percent is also very flat as long as zone 1 was at least 20%, and the first two zones at least 50%. We conclude that the class-based storage policy is fairly robust against the zoning settings.

Space Utilization. In the base case, the ratio of pods to storage locations is set to 99%. In this test, we create additional storage locations for each velocity zone proportional to the size of the zone. We do this by converting some aisle space into storage space. We find that the improvement from class-based policies decreases in the number of open storage locations in the system. For instance, when we increase the number of excess locations to 2%, the improvement for the 2-class system drops from 8.9% to 7.6%. This occurs because there are now more open locations in zone 1 available to the closest-location policy.

Velocity Threshold. We test the class-based storage policy for various choices of the parameter γ that controls how we adjust the velocity threshold. A larger γ puts more weight on the current distribution of open locations in setting the velocity threshold. We find that the class-based storage policy is not very sensitive to the value of γ as long as it is not below 0.25.

Picking Algorithm. We test the storage policies using an improved picking algorithm that doubles the average number of units picked per pod visit at the pick station. We find that the class-based storage policy is slightly more effective with the improved picking algorithm, providing a 9.8% and 10.6% reduction on travel distance for the 2-class and 3-class storage policy respectively.

5 Conclusion and Future Research

In this paper we examine the pod storage decision for a semi-automated storage system, with a particular focus on velocity-based storage policies. The key concept underlying a velocity-based storage policy is to exploit heterogeneity in the velocities of the storage units: high-velocity storage units are stored nearby, whereas low-velocity units are stored far away. To get some insight into the potential from velocity-based storage policies, we first develop a fluid model. From this fluid model, we can make the following findings:

- If items are stowed randomly, there is a modest amount of heterogeneity in the pod velocities. Depending on the system parameters, the maximum reduction in travel time from a velocity-based storage policy is on the order of 6 to 12% compared to random storage. We find that a 2-class storage policy can achieve about 75% of the maximum theoretical improvement; a 3-class policy is even better achieving around 90% of the maximum improvement.
- If items are stowed based on their velocities, the potential from velocity-based storage is much greater, with travel time reductions of as much as 40% relative to random storage. Again, 2-class and 3-class storage can provide around 75% and 90% of this maximum improvement.

Both of these findings have important implications to practice. The first is that simple and pragmatic policies can capture most of the potential benefit from velocity-based storage in this system. There are minimal incremental benefits from considering a more complex policy with more than three storage classes. Second, we see that even with random stowage, which is quite common, there is heterogeneity in the pod velocities that can be exploited by these storage policies with meaningful impact. The third implication is the potential from smarter stowage that tries to increase the velocity heterogeneity across the pods. From the fluid model we find that travel time reductions can be increased significantly by creating fast and slow pods, etc.

The fluid model, as any model, is a theoretical construct based on a series of assumptions. To validate the findings from the fluid model, we conduct a realistic, large-scale simulation of a semi-automated storage system with the actual inventory and picking data. The key contributions from the simulation are:

- We are able to confirm the benefits from velocity-based storage. The actual storage system operates with random stowage. So the simulation results should be compared to those found from the fluid model under random stowage. We find that the simulated improvements from 2-class (8.9 %) and from 3-class (9.8 %) are consistent with the predictions from the fluid model for a system with high inventory and high demand variability, i.e., the cases in Figure 4 with settings $(\nu = 1.0 \; or \; 1.5, J = 4000, \alpha = 0.01 \; or \; 0.05)$.
- We test a practical way to set the velocity measure for a pod based on the order backlog. In the actual context, there is visibility of what skus need to be picked over the near term based on the dynamic order backlog. From this we develop a velocity measure that is indicative of how many picks might be expected from each pod in the near term.
- We develop and test a dynamic policy for adjusting the velocity thresholds for determining the break-points between different storage classes in a 2-class and 3-class policy. This is important in the dynamic setting that is simulated, as the pod velocities depend on the size of the order backlog which fluctuates over a day. By adjusting the thresholds we are able to manage the space utilization for each storage class so as to avoid blocking.
- We conduct a sensitivity analysis that finds that the improvements from the 2-class and 3-class policies are reasonably robust to how the storage zones are set, how the threshold adjustments are made, and to the level of space and worker utilization. We also simulated an improved

picking algorithm and obtained similar improvements from velocity-based storage.

As one of the first papers to consider this type of storage system, there remains many issues for further work. We discuss a few research opportunities in the following.

Pod velocity measure. The pod velocity should be an indicator that predicts the likelihood of a pod being requested. Fundamentally, the effectiveness of the class-based storage decision relies on our ability to accurately estimate the velocity of pods. For the fluid model, we assume that the units are ranked in the sequence that they will be used to satisfy the demand. For the simulation, we propose a pod velocity measure that considers both the existing backlog and the distribution of the items across the storage pods. One way to further improve the class-based storage policies is to identify more accurate pod velocity measures that account for additional information. In particular, there is possibly useful information given by the specific picking algorithm applied in the fulfillment center.

Variation of demand. To analyze the benefits from a velocity-based storage policy, we assume in our fluid model that we know the demand rate for all items in our analysis. However, in practice, demand rates may not be known and/or may vary over time; furthermore, the product assortment changes frequently with new items steadily entering the system. It is of interest to design dynamic storage policies that might account in some way for this limited knowledge on demand.

Variation of item size. For our analysis we assume that all items have identical size and consume the same amount of a capacity on a pod. However the actual variation of the item sizes can be large. One possible solution is to consider the cube-per-order index (COI) rule, first suggested by Heskett (1963, 1964), which is the ratio of the storage space requirement to the demand rate. Future research might examine how to modify velocity-based storage to account for varying unit sizes in a semi-automated storage system.

Velocity-based stowage policy. One way to further improve the performance of the class-based storage policy is by stowing products more intelligently to the storage pods. We have shown with the fluid model that there is a great benefit from velocity-based stowage in a theoretical framework. Future research might test or model some version of the velocity-based stowage and storage decisions together.

N-class. Further research is required to investigate the ideal number of classes for a class-based storage policy in the semi-automated storage system. This might be similar to Rosenblatt and Eynan (1989), and Van den Berg (1996), who propose recursive and dynamic programming procedures respectively to find optimal zone sizes for an N-class system for an AS/RS.

Reference

Bartholdi III, John J., and Steven T. Hackman. 2011. Warehouse & distribution science: release 0.92. *The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology*.

D'Andrea, Raffaello, and Peter Wurman. 2008. Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities. *Technologies for Practical Robot Applications, 2008. TePRA 2008. IEEE International Conference on*, pp. 80-83. IEEE.

De Koster, René, Tho Le-Duc, and Kees Jan Roodbergen. 2007. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research* 182(2), 481-501.

Enright, John, and Peter Wurman. 2011. Optimization and Coordinated Autonomy in Mobile Fulfillment Systems. *Automated action planning for autonomous mobile robots*, pp. 33-38.

Eynan, Amit, and Meir J. Rosenblatt. 1994. Establishing zones in single-command class-based rectangular AS/RS." *IIE Transactions* 26(1), 38-46.

Gagliardi, Jean-Philippe, Jacques Renaud, and Angel Ruiz. 2012. Models for automated storage and retrieval systems: a literature review. *International Journal of Production Research* 50(24), 7110-7125.

Graves, Stephen C., Warren H. Hausman, and Leroy B. Schwarz. 1977. Storage-retrieval interleaving in automatic warehousing systems. *Management Science* 23(9), 935-945

Gu, Jinxiang, Marc Goetschalckx, and Leon F. McGinnis. 2007. Research on warehouse operation: A comprehensive review. *European Journal of Operational Research* 177(1), 1-21.

Guenov, Marin, and Robert Raeside. 1992. Zone shapes in class based storage and multicommand order picking when storage/retrieval machines are used. *European Journal of Operational Research* 58(1),37-47.

Hausman, Warren H., Leroy B. Schwarz, and Stephen C. Graves. 1976. Optimal storage assignment in automatic warehousing systems. *Management Science* 22(6),629-638.

Heskett, James L. 1963. Cube-per-order index-a key to warehouse stock location. *Transportation and Distribution Management* 3(1), 27-31.

Heskett, James L. 1964. Putting the cube-per-order index to work in warehouse layout. *Transportation and Distribution Management* 4(8), 23-30.

Kouvelis, P., and V. Papanicolaou. 1995. Expected travel time and optimal boundary formulas for a two-class-based automated storage/retrieval system. *International Journal of Production Research* 33(10), 2889-2905.

Lamballais, T., D. Roy, and M. B. M. De Koster. 2017. Estimating performance in a Robotic Mobile Fulfillment System. *European Journal of Operational Research* 256(3), 976-990.

Lamballais, T., D. Roy, and M. B. M. De Koster. 2017. Inventory Allocation in Robotic Mobile Fulfillment Systems. working paper, January.

Roodbergen, Kees Jan, and Iris FA Vis. 2009. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research* 194(2), 343-362.

Rosenblatt, Meir J., and Amit Eynan. 1989. Note—Deriving the Optimal Boundaries for Class-Based Automatic Storage/Retrieval Systems. *Management Science* 35(12), 1519-1524.

Rouwenhorst, Bart, B. Reuter, V. Stockrahm, G. J. Van Houtum, R. J. Mantel, and W. H. M. Zijm. 2000. Warehouse design and control: Framework and literature review. *European Journal of Operational Research* 122(3), 515-533.

Thonemann, Ulrich W., and Margaret L. Brandeau. 1998. Note. Optimal storage assignment policies for automated storage and retrieval systems with stochastic demands. *Management Science* 44(1), 142-148.

Van den Berg, J. P. 1996. Class-based storage allocation in a single-command warehouse with space requirement constraints. *International Journal of Industrial Engineering* 3, 21-28.

Weidinger, Felix, Nils Boysen and Dirk Briskorn. 2016. Storage assignment with rack-moving mobile robots in KIVA warehouses. Friedrich-Schiller Universitat Jena, working paper, December.

Yuan, Rong. 2016. *Velocity-based Storage and Stowage Decisions in a Semi-automated Fulfillment System*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Appendix A: Characteristics of the Improvement Ratio of the Full-velocity Storage Policy

Proposition A1: The improvement ratio of the full-velocity storage policy under the velocity-based stowage policy, i.e. expression (17), is positive for any give value of $\alpha \in (0,1)$ and $J > 0$.

Proof:

We consider the two cases where J is odd and even respectively.

For J odd, we have

$$R_{Velocity} = \frac{2}{J+1}\sum_{j=1}^{J-1}\left(\frac{1-\alpha^{j/J}}{1-\alpha}-\frac{1}{2}\right)$$

$$= \frac{2}{J+1}\sum_{j=1}^{\frac{J-1}{2}}\left(\frac{\left(1-\alpha^{j/J}\right)\left(1-\alpha^{\frac{J-j}{J}}\right)}{1-\alpha}\right) > 0.$$

We note that the equality is found by adding up all the i^{th} and $(J-i)^{th}$ term within the summation.

For J even, similarly, we have

$$R_{Velocity} = \frac{2}{J+1}\sum_{j=1}^{J-1}\left(\frac{1-\alpha^{j/J}}{1-\alpha}-\frac{1}{2}\right)$$

$$= \frac{2}{J+1}\left[\sum_{j=1}^{\frac{J}{2}-1}\left(\frac{\left(1-\alpha^{j/J}\right)\left(1-\alpha^{\frac{J-j}{J}}\right)}{1-\alpha}\right)+\frac{\left(1-\alpha^{1/2}\right)^2}{2(1-\alpha)}\right] > 0 \qquad Q.E.D.$$

Proposition A2: The improvement ratio of the full-velocity storage policy under the velocity-based stowage policy decreases in α for $\alpha \in (0,1)$.

Proof:

For $x \in (0,1), a \in (0,1)$, we consider the following function

$$f(x,a) = \frac{2}{1/a+1}\left(\frac{1/a-1}{1-x}-\frac{x^a-x}{\left(1-x^a\right)(1-x)}-\frac{1/a-1}{2}\right).$$

We note that we can evaluate the improvement ratio of the full-velocity storage policy at $(\alpha, \frac{1}{J})$ of the function *f*. We want to show that the function f is decreasing in x on its defined domain.

Taking the derivative of f, we have

$$\frac{df(x)}{dx}=\frac{2}{1/a+1}\frac{x\left(1-x^a\right)^2-\left(1-x\right)^2x^a a^2}{a\left(x-1\right)^2x\left(x^a-1\right)^2}.$$

We would like to show that the above derivative of f is negative for $x\in\left(0,1\right),a\in\left(0,1\right)$. We only need to consider the term $x\left(1-x^a\right)^2-\left(1-x\right)^2x^a a^2$ as the other terms are obviously positive.

We observe that the following inequalities are equivalent:

$$\begin{aligned}
&x\left(1-x^a\right)^2-\left(1-x\right)^2x^a a^2<0\\
&\Leftrightarrow\frac{x\left(1-x^a\right)^2}{\left(1-x\right)^2x^a}<a^2\\
&\Leftrightarrow x^{\frac{1-a}{2}}\frac{1-x^a}{1-x}<a\\
&\Leftrightarrow x^{\frac{1-a}{2}}-x^{\frac{1+a}{2}}-\left(1-x\right)a<0.
\end{aligned}$$

So we only need to consider

$$g(x)=x^{\frac{1-a}{2}}-x^{\frac{1+a}{2}}-\left(1-x\right)a.$$

Taking the derivative of g, we have

$$\begin{aligned}
g'(x)&=\frac{1-a}{2}x^{-\frac{1+a}{2}}-\frac{1+a}{2}x^{-\frac{1-a}{2}}+a\\
&=p(x^{-(1-p)}-1)-\left(1-p\right)(x^{-p}-1)
\end{aligned}$$

where $p=\frac{1-a}{2}\in\left(0,\frac{1}{2}\right)$.

By observing

$$p(x^{-(1-p)}-1)-\left(1-p\right)(x^{-p}-1)>0\Leftrightarrow\frac{x^{-(1-p)}-1}{1-p}>\frac{x^{-p}-1}{p}$$

and the second inequality is always true for $p\in\left(0,\frac{1}{2}\right)$ as the function $\frac{x^{-p}-1}{p}$ is increasing in p when $p\in(0,1)$. We thus prove that $g'(x)$ is always positive for $p\in\left(0,\frac{1}{2}\right),x\in\left(0,1\right)$.

This implies that $g(x)$ is increasing for $x\in\left(0,1\right)$. Since $g(0)=-a,g(1)=0$, we know that $g(x)$ is negative for $x \in (0,1)$, which further proves that $\frac{df(x)}{dx} < 0$ for $x \in (0,1)$.

Q.E.D.

Appendix B: Simulation with Stow Trips

In Section 4 we reported on our simulation study to validate the findings from the fluid model. The primary intent of the simulation was to examine the effectiveness of a class-based storage policy in a large scale fulfillment center with real industrial data, in comparison to the actual storage policy. As such we made a number of simplifying assumptions. One assumption (A1) was to not model the stow operations within the simulation. The primary justification was to avoid the complexity required to simulate both pick and stow trips. Furthermore, we argued that the travel-time benefits from class-based storage policies should remain the same as long as we follow the same logic for returning the pods back to the storage zones for the stowing operation as for the picking operation.

In this appendix we relax this assumption and include stow trips in the simulation. To do this we make the following assumptions:

- We assume that each pod gets replenished with new inventory according to the actual stowing record, at roughly the actual time. Specifically, we aggregate the stow events within each hour and assume that for each pod there is at most a single stow trip that will occur at the start of the hour. That is, we include each stowage event in the simulation.
- For each stow trip, we assume that the pod travels from its pod storage location to the closest stow station, and then returns to the same storage location. Hence, the travel time for each stow event is the round-trip from the pod's storage location to the nearest stow station. And, we do not reevaluate the pod's velocity and storage class after the stow event.
- For the simulation we assume that the stow trip occurs instantaneously, so that we don't need to model any queuing at the stow stations or interference with other trips.
- We assume that the stow trips are not combined with a pick trip, which is consistent with the actual data.

We report below in Table A1 the simulation results, analogous to Table 3, for the base case.

	Closest-open-location	**2-class**	**3-class**
Forward Trip Distance: pick	1.00	0.94	0.93
Return Trip Distance: pick	1.04	0.93	0.91
Total Travel Distance: pick	2.04	1.86	1.84
Total Travel Distance: stow	0.14	0.13	0.12
Total Travel Distance	2.18	1.99	1.96
Improvement Ratio		8.7%	10.1%

Table A1: Performance Comparison for the Closest-open-location, the 2-class, and the 3-class Storage Policy under the Base Case Scenario (10 Stations, 1% Empty Space, $\alpha = 0.5$)

In this table the first three rows of results are the same as in Table 3, namely the normalized travel distances for the pick trips. In the fourth row we report the total travel time for the stow events, normalized by the total travel time for the forward trips for the pick events. The fifth row reports the total travel time for each storage policy, accounting for both pick trips and stow trips. We have a couple of observations.

First, the travel time for stow events is less than 10% of the travel time for pick trips. There are two reasons for this: there are many fewer stow trips than pick trips, and the travel time per trip is shorter for the stow trips.

The number of units stowed per trip is about five times the number of units picked per trip, resulting in many fewer stow than pick trips; furthermore, over the course of the week more units were picked than stowed, which also contributes to fewer stow trips.

The length of the average stow trip was roughly 85% that for a pick trip. This is primarily attributable to the simplifying assumption that for each stow trip, we could go to the nearest stow station.

The second observation is that the inclusion of the stow trips does not affect our findings, as reported in Section 4. Namely, we see that the class-based storage policies outperform the closest location policy, and can yield travel time reductions on the order of 8 to 10%. As expected both pick and stow trips benefit from the class-based storage policies. However, since over 90% of the travel is due to pick trips, most of the benefits are from these trips.