Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

THROUGHPUT CONTROL AND GIVEAWAY MINIMIZATION OF A POULTRY PRODUCT BATCHER

Kay Peeters

Department of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
Eindhoven, 5600 MB, THE NETHERLANDS

Ivo Adan
Tugce Martagan

Department of Industrial Engineering
Eindhoven University of Technology
PO Box 513
Eindhoven, 5600 MB, THE NETHERLANDS

ABSTRACT

Poultry plants have to operate under challenging deadlines due to recent developments in the industry. They have expressed the need for improved control over their batching process to deal with these changes. In particular, they wish to achieve a target throughput to ensure deadlines are made, while minimizing the giveaway. This paper proposes an algorithm that is capable of controlling such a poultry product batcher, and aims to achieve a target throughput and minimize the giveaway. Using a simulation study we find that both objectives can be achieved under a wide range of settings using the proposed algorithm.

1 INTRODUCTION

The poultry industry comprises the supply chain that manages the growth, processing and sales of poultry products. Farmers grow broilers and deliver them to poultry processing plants where they are processed into a variety of final products (e.g. fillets, legs, or wings). Poultry processing plants use specialized equipment supplied by equipment manufacturers to divide broilers into parts and package them. A processing plant using high-end equipment is typically capable of processing over 200.000 broilers daily over a 16 hour period. End-products are subsequently delivered to customers of the plant, such as fast-food chains and supermarkets. In this paper we consider the perspective of such a poultry processing plant that faces demand from a supermarket chain. Due to increasingly stringent production requirements and smaller profit margins, poultry processing plants have expressed the need for improved control over their production process to remain profitable.

The poultry industry has undergone a number of transformations as outlined in Manning and Baines (2004). Globalization has led to the consolidation of food retailers and food service companies also known as business clusters. Business clusters engage in the practice of contract farming, where they agree to purchase much of the output of a processing plant as long as it is produced according to their requirements. In particular, these contracts dictate: 1) a daily distribution center delivery window during which the order must be delivered, and 2) that plants are paid a fixed price per produced batch under the condition that it contains a given target weight at minimum. For example, one may find packages of fillets, also referred to as batches, in the supermarket containing at least $600g$ worth of weight. Additional weight in a batch, or giveaway, is not paid for. Furthermore, missing a deadline could mean that a business cluster will seek their business elsewhere, resulting in a massive loss of revenue. While orders from the open market may not impose deadlines or batch target weights, they are also worth less per batch. As a result, it is of critical importance for poultry processing plants to supply to business clusters and obey their contract obligations through control of their production process.

Batchers are machines used to produce batches of products for retail customers. Oftentimes, a machine called a grader is used for this purpose, which is shown in Figure 1. Knowing the size of an order and its delivery window, plants specify a due date at which the order must be completed. Such information can be directly translated to a required throughput in products per time unit at a grader. In this paper we consider the following setting. A grader produces batches for a business cluster with a strict deadline and a given target weight. Products are supplied via a conveyor belt and the machine is allowed to pass or reject items to a downstream process that produces orders for the open market in order to reduce giveaway, as long as the deadline is met. That is, rejected items are batched or processed into products for other customers. However, the downstream process is outside of the scope of this paper. A schematic overview of the grader is shown in Figure 2. Unfortunately, the objectives of minimizing giveaway and meeting a deadline can be contradicting. If we reject items in a clever way it may be possible to reduce giveaway, but it comes at the cost of throughput. Due to decreasing profit margins, minimizing the giveaway has become of vital importance.

Figure 1: Grader.

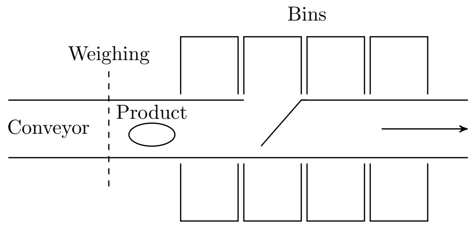

Figure 2: Schematic overview of a grader.

There are algorithms capable of controlling a batcher in such a setting. However, these algorithms are only capable of controlling the fraction of items allocated to a grader in combination with minimizing the giveaway. Depending on the items allocated by such an algorithm, the throughput may deviate from the target throughput and often requires adjustments during a production run. The objective of this paper is to investigate whether it is possible to design a grader algorithm that minimizes the giveaway and achieves a target throughput. To this end we propose an algorithm that aims to achieve both objectives simultaneously. This leads us to our main research question: does the proposed algorithm for controlling the grader both yield low giveaway by rejecting products and achieve the target throughput? This question will be answered by means of an extensive simulation study. In particular, we study the convergence of the throughput, and sensitivity of the performance of the algorithm for different target weights and target throughputs.

The paper is organized as follows. In Section 2, literature relevant to this problem and the proposed algorithm is discussed. Next, Section 3 introduces the model describing the grader, item weights, and performance measures. Section 4 discusses algorithms that are typically used in a grader, and the proposed extension to setting a target throughput. Simulation experiments are presented in Section 5, and our conclusions and recommendations for future work are summarized in Section 6.

2 LITERATURE

In literature, the challenge of allocating products to bins in order to minimize the giveaway is known as the online K-bounded bin-covering problem. Online refers to the items that arrive one-by-one and have to be allocated as they arrive, and K refers to the fixed number of available bins. The bin-covering problem is characterized by items that have to be allocated to bins, where each bin has a fixed capacity and the goal is to minimize the giveaway. In our case rejection of items is also considered.

Packing problems such as bin covering have received ample attention in literature. Bin covering is often considered in a stylized setting with uniformly distributed item weights, where the goal is to minimize the giveaway while all items are known and can be re-ordered (offline), and an unlimited number of bins can be used (unbounded). The focus of these studies are often bounds on algorithm performance relative to the optimal packing strategy (e.g Assmann et al. (1984)). Our problem has the following additional properties: 1) items arrive online, 2) the number of available bins is bounded, 3) items can be rejected, and 4) there is a throughput constraint. Csirik et al. (2001) introduce the Sum-of-Squares (SS) algorithm for bin covering that is online and bounded. Asgeirsson and Stein (2006) and Asgeirsson and Stein (2009) use Markov chains to predict giveaway and use it for controlling the online bounded problem in their Min-Bin-State-Cost algorithm. Similarly, Ásgeirsson (2014) introduces a prospect function to predict the cost for full bins and use it in their Prospect Algorithm to control the online and bounded problem. He also briefly discusses a modification of the prospect algorithm to allow for the rejection of items. Peeters et al. (2017) introduce an index policy for allocating items to multiple batchers. The policy is shown to have a similar structure as the prospect algorithm, as they both predict the expected loss at a given bin level (index) and use it to allocate products.

The bin-covering problem with rejection is studied in a number of papers. They consider the case where each item has a given rejection value, and each bin filled has a fixed value. That is, for each bin filled and each item rejected a reward is acquired and the goal is to maximize the total reward. However, these papers place no constraint on the throughput and often partition the item weights and use some arbitrarily large bound on the number of bins to achieve good performance (e.g. Dósa and He (2006), Correa and Epstein (2008)). Instead, we continue the line of work of Asgeirsson and Stein (2006), Asgeirsson and Stein (2009), Ásgeirsson (2014), and Peeters et al. (2017), which all use a type of index function to predict the costs of full bins in a comparable realistic setting. They permit the use of arbitrary weight distributions, use a fixed bound on the number of bins, and study the performance of different algorithms in simulation studies. However, none of these papers consider a target throughput.

In our study, we will use index policies as a basis for allocating and rejecting items. The scientific contribution of this paper will be the introduction of an algorithm for the online K-bounded bin-covering problem with throughput constraint, capable of performing under any given number of bins and item weight distribution. An extensive simulation study is performed to illustrate the behavior, and evaluate the performance of this algorithm. In Section 3 the model for the grader with rejection is introduced, which is used in Section 4 to introduce the heuristic developed for this problem.

3 MODEL

In this section we introduce the model that is used to describe the batching process in a grader. Section 3.1 outlines the production process and provides a detailed description of the grader. Next, Section 3.2 introduces a formal model of this batcher, characterizes the arriving products and introduces relevant performance measures.

3.1 Production Process

Before products reach a grader a number of production steps are performed. Flocks of broilers are delivered to a processing plant in trucks after which a flock is placed in the *primary process*. Firstly, the broilers are stunned and slaughtered. Secondly, the broilers are hanged in shackles by their feet, defeathered, eviscerated and cooled. After cooling down, broilers are moved to the *secondary process*. In this process, chickens are cut up into different parts and processed to end-products. In the secondary process broilers pass a number of production steps. First, they are divided in a cut-up line before moving to processing lines. Within the processing lines, products such as chicken legs, wings, and fillets are produced. These products are then transported to different batchers using conveyor belts. Lastly, batches of products are packaged and stored before being transported to the customer. A detailed description of the production process can be found in Peeters et al. (2017).

The product batcher considered in this paper is a grader. A schematic overview of this batcher can be found in Figure 2. Some time before arriving to the grader, broilers are placed in the cooling process that takes up to two hours. Since the broilers are weighed before they are placed in the cooling process and it takes long for them to reach the grader, we assume their weight distribution is known and stationary. Due to the nature of intermediate processing steps, the order of arrival of products to the grader can be mixed up. Products are weighed on the fly on a conveyor belt, and weighed accurately to the nearest gram. Furthermore, the distance from the weighing scale to the grader is small. Therefore the only information known to the grader is the weight of the next item, the weight distribution, and the contents of its bins. Based on this information an algorithm then allocates products to one of the bins. When an allocated product passes its designated bin, a flap is opened to guide the product into the holder. Bins are instantaneously emptied as soon as the batch it holds has reached or surpassed its target weight. Next, the information of this section is more formally defined in Section 3.2.

3.2 Grader Model

In this section a formal model of the grader is introduced. Items arrive online: they are weighed and processed one-by-one. The probability that the next arriving item weighs w is $p(w)$. Since we are dealing with real products, there is some positive lower and upper bound on the weights observed, $w_{\min}$ and $w_{\max}$ respectively. The set of observed item weights is denoted by $\mathscr{W} = \{w_{\min}, w_{\min}+1, \dots, w_{\max}\}$. The grader has a fixed number of bins K, which are emptied as soon as a the bin contains at least the target weight B worth of weight. Bins are labeled 1 through K. The corresponding set of bins is given as $\mathscr{K} = \{1, 2, \dots, K\}$. The weight levels that a bin can contain is denoted by $\mathscr{V} = \{0, 1, \dots, B-1\}$. The weight in bin k is written as $v_k \in \mathscr{V}$ for all bins $k \in \mathscr{K}$. The giveaway function $g(x)$ in Equation (1) is used to calculate the giveaway. The function is only positive if its argument exceeds the target weight B, and is 0 otherwise.

$$g(x) = \begin{cases} 0, & x \le B, \\ x - B, & x > B. \end{cases} \tag{1}$$

Items can be allocated to a bin or rejected. The grader is allowed to reject items under the condition that a long-term average target throughput is met. Following the paper of Peeters et al. (2017), we use the realistic assumption that weights are Normally distributed $\mathscr{N}(\mu, \sigma)$ with mean μ and standard deviation σ. The average inter-arrival time between products is denoted by t. Recall that plants will set a target throughput based on a contractual delivery window. Suppose an order of Q batches with target weight B has to be finished in T units of time, and that it is the only order that will be processed on a given grader. The average target throughput in weight batched per time is $\frac{Q \cdot B}{T}$, and an average of $\frac{\mu}{t}$ weight per time unit will be processed. This translates into a target throughput of $q = \frac{Q \cdot B \cdot t}{T \cdot \mu}$ weight per gram. The goal is now to pack or reject items such that the long-term average giveaway is minimized, under the condition that the long-term average throughput is equal to the target throughput q.

Next, we introduce the (fraction of) long-term average throughput weight, giveaway and rejected weight. Together they serve as performance indicators of a grader algorithm. An item is called *processed* when it has been rejected or allocated to a bin. Rejected weight is the weight that has been rejected by the grader. Allocated weight is weight that has been allocated to a grader. Within allocated weight we make a distinction between throughput weight and giveaway. Throughput weight is weight that has contributed to the batch, and giveaway is the weight by which batches have been overfilled. Together, batched weight, giveaway, and rejected weight sum to the processed weight.

Suppose that we are processing items on a grader using a given allocation policy and have processed a total of N items. After N items, the total throughput weight is denoted as w_N^b, the total giveaway is denoted as w_N^g and the total rejected weight is denoted as w_N^r. Now we observe item number $N+1$ with weight $w \in \mathscr{W}$. If the item is rejected, the total rejected weight becomes $w_{N+1}^r = w_N^r + w$, the total batched weight becomes $w_{N+1}^b = w_N^b$, and the total giveaway becomes $w_{N+1}^g = w_N^g$. On the other hand, if the item is allocated to a bin $k \in \mathscr{K}$: the total rejected weight becomes $w_{N+1}^r = w_N^r$, the total batched weight becomes

$w^b_{N+1} = w^b_N + (w - g(v_k + w))$, and the total giveaway becomes $w^g_{N+1} = w^g_N + g(v_k + w)$. Hence:

$$w^b_{N+1} = \begin{cases} w^b_N + (w - g(v_k + w)), \\ w^b_N, \end{cases} \quad w^g_{N+1} = \begin{cases} w^g_N + g(v_k + w), \\ w^g_N, \end{cases} \quad w^r_{N+1} = \begin{cases} w^r_N, & \text{if accepted} \\ w^r_N + w, & \text{if rejected.} \end{cases} \tag{2}$$

Let w_N denote the total weight of the first N items. Then we have

$$w_N = w^b_N + w^g_N + w^r_N. \tag{3}$$

Similarly, we can define the fraction of throughput $\bar{w}^b_N = w^b_N / w_N$, giveaway $\bar{w}^g_N = w^g_N / w_N$, and weight rejected $\bar{w}^r_N = w^r_N / w_N$ after N items. Combining these definitions with Equation (3) yields Equation (4).

$$1 = \frac{w^b_N}{w_N} + \frac{w^g_N}{w_N} + \frac{w^r_N}{w_N} = \bar{w}^b_N + \bar{w}^g_N + \bar{w}^r_N. \tag{4}$$

Furthermore, the long-term fraction of throughput weight is defined as $\bar{w}^b = \lim_{N\to\infty} w^b_N / w_N$, fraction of giveaway as $\bar{w}^g = \lim_{N\to\infty} w^g_N / w_N$, and fraction of rejected weight as $\bar{w}^r = \lim_{N\to\infty} w^r_N / w_N$, assuming that the limits exist. This results in Equation (5).

$$1 = \bar{w}^b + \bar{w}^g + \bar{w}^r \tag{5}$$

The goal is to minimize the long-term fraction of giveaway $\bar{w}^g$, under the condition that the long-term fraction of throughput weight $\bar{w}^b$ is equal to the target q. In order to treat this problem, Section 4 first introduces policies that minimize the giveaway using a grader, and then presents an extension that attempts to also achieve target q by rejecting items. The performance of the algorithm in terms of giveaway $\bar{w}^g$ and throughput $\bar{w}^b$ is later analyzed in a numerical study in Section 5.

4 INDEX POLICIES WITH REJECTION

Index policies have been successfully used in bin-covering problems in a practical setting (e.g. Ásgeirsson (2014) and Peeters et al. (2017)). In this section we will formally describe such index policies and motivate their use in our specific setting. Lastly, an extension to index policies that makes it possible to reject items in a desirable way will be discussed.

4.1 Index Policies

Index policies are algorithms that calculate an index value for all of its available options, and choose the option with the highest index. For some problems an index policy is even provably optimal (e.g. Weitzman (1979)). For the bin-covering problem an index $\ell(v)$ can be defined that denotes the expected loss at some bin level $v \in \mathscr{V}$ and given target weight B. The procedure for finding these indexes is to first define a loss function $f(v)$ for full bin levels ($v \geq B$), and then recursively calculate index values for non-full bin levels ($v < B$) using properties of the weight distribution. This procedure is expressed by Equation (6).

$$\ell(v) = \begin{cases} f(v), & v \geq B, \\ \sum_{w\in\mathscr{W}} p(w)\ell(v+w), & v = B-1, B-2, \ldots, 1. \end{cases} \tag{6}$$

An example of a loss function $f(v)$ is using the giveaway of a bin. The motivation for using this type of index is that once a bin is filled we know its loss exactly, but this loss has to be estimated for non-full bin levels. The expected loss for non-full bin levels is calculated under the assumption that each level sees the full weight distribution. That is, we assume $p(w)$ does not depend on the bin level v. Note that the expected loss is exact when the grader comprises a single bin. Otherwise, the weight distribution observed per bin level will depend on the policy used and other bin levels. An index policy can now be defined, which uses the indexes to determine to which bin an item should be allocated. Since each index indicates the expected loss at a bin level, the goal will be to find a policy that maximizes the reduction of the expected loss in some way. In this paper we consider a policy that allocates each item to the bin with the largest reduction of expected loss $k^*(v_1,\dots,v_K,w)$, as presented in Equation (7). For convenience we also introduce the value of the largest reduction of expected loss $\ell^*(v_1,\dots,v_K,w)$ in Equation (8), which is later used to determine whether an item should be rejected or not.

$$k^*(v_1,\dots,v_K,w) = \underset{k\in\mathscr{K}}{\arg\max}\{\ell(v_k)-\ell(v_k+w)\}, \tag{7}$$

$$\ell^*(v_1,\dots,v_K,w) = \max_{k\in\mathscr{K}}\{\ell(v_k)-\ell(v_k+w)\}. \tag{8}$$

Papers have used different loss functions and policies. Ásgeirsson (2014) indicates that loss functions that discount higher levels of giveaway achieve comparatively good results. In our paper we use the loss function $f(v)$ for full bins ($v \geq B$) as presented in Equation (9).

$$f(v) = (v-B)^{\alpha} \tag{9}$$

Here, α is a nonnegative constant to be chosen such that the policy achieves a minimal long-term average giveaway. The motivation for this form of the loss function is that it encompasses: the trivial Next-Fit algorithm ($\alpha = 0$) that allocates all items to a single bin, the Index Policy ($\alpha = 1$) that was presented in Peeters et al. (2017) and did not discount the giveaway, and can discount giveaway ($0 < \alpha < 1$). The index policy with the loss function in Equation (9) that allocates items to bins using Equation (7) will be referred to as an Index Policy with Power function (IPP). Application of the IPP is outlined in Algorithm 1.

Algorithm 1 The *Index Policy with Power function (IPP)* for the K-bounded on-line bin-covering problem consists of the following steps. Input an α for the loss function in Equation (9). Recursively calculate the indexes $\ell(v)$ for all bin levels $v \in \mathscr{V}$ using Equation (6), given the target weight B.
While items arrive do:

1. When an item with weight $w \in \mathscr{W}$ arrives, determine the policy-optimal bin $k^*(v_1,\dots,v_K,w)$ using Equation (7). Go to step 2.
2. Allocate the item to bin $k^*(v_1,\dots,v_K,w)$. Empty the bin if $v_{k^*(v_1,\dots,v_K,w)} \geq B$. Return to step 1.

Studying the IPP in simulation experiments we have empirically observed that: 1) the optimal value of α lies between 0 and 1, and 2) the giveaway is convex in α between $\alpha = 0$ and $\alpha = 1$. Assuming 1) and 2), we propose a bisection method for finding a suitable α. Initialize α at 0.5 and estimate the giveaway $\bar{w}^g$ through simulation. Step 1) is to consider $\alpha \pm \Delta_1$ with $\Delta_1 = 2^{-2}$. If either value yields lower giveaway, update α. Step 2) is to consider $\alpha \pm \Delta_2$ with $\Delta_2 = 2^{-3}$, and update α^* if it improves upon the giveaway. The procedure is repeated until the step size $\Delta_A = 2^{-(A+1)}$ in step A is sufficiently small. Algorithm 1 in combination with this bisection method to determine the optimal α can now be used to find an index policy that yields low giveaway. Next, in Section 4.2 an extension to this approach is introduced that allows the rejection of items in order to the reduce the giveaway, while guaranteeing a target throughput.

4.2 Rejection Threshold

Index policies use the expected loss for a given bin level to control a grader. While it is only an estimation of the true loss, it still serves as an indication of the quality of an allocation and allows a direct comparison between the quality of different allocations. For convenience, we use the shorthand notation k^* to denote the policy-optimal bin from Equation (7), and ℓ^* from Equation (8) to denote the policy-optimal value for given bin levels and item weight in the following.

Ásgeirsson (2014) introduced an extension to index policies that considers the rejection of items. If the largest reduction of the expected loss ℓ^* is above a threshold value R, the item is accepted, and rejected otherwise. In a simulation study he shows that by changing the threshold value, the rate of acceptance and giveaway change as well. Unfortunately, his approach would require extensive tuning to find a combination of loss function and rejection threshold that achieves the desired throughput, if it even exists. We expand this idea to a dynamic threshold to improve the level of controllability. The proposed method updates the value of the threshold after each processed item, which is described in the remainder of this section.

Since the threshold R is no longer constant, we let R_N denote the value of the threshold after N items have been processed. This means that item $N+1$ is rejected if $\ell^* < R_N$, or allocated to bin k^* otherwise. A method for updating the threshold is now proposed in Equation (10). The idea is to increase the threshold proportional to the throughput weight, and decrease the threshold proportional to the giveaway and rejected weight. The former is scaled with a constant C that is yet to be determined. We conjecture that by deriving C by considering the throughput requirement, the threshold is self-stabilizing. That is, if we are below the target throughput the threshold is decreased and becomes less strict, and if we are above the target throughput the threshold is increased and becomes more strict.

$$R_{N+1} = \begin{cases} R_N - w, & \text{if rejected,} \\ R_N - g(v_{k^*}+w) + C\cdot(w - g(v_{k^*}+w)), & \text{if accepted.} \end{cases} \tag{10}$$

Given the value of the threshold for the first item R_0, its evolution can be described as follows. If the first item with weight w_1 is rejected: $R_1 = R_0 - w_1$. If the item is accepted: $R_1 = R_0 - \cdot g(v_{k^*}+w) + C\cdot(w_1 - g(v_{k^*}+w_1))$. Similarly, if the second item with weight $w_2 \in \mathscr{W}$ is accepted: $R_2 = R_1 - g(v_{k^*}+w_2) + C\cdot(w_2 - g(v_{k^*}+w_2))$, and $R_2 = R_1 - w_2$ if it is rejected. It follows from Equations (2) and (10) that after N items, the threshold has been increased by a total of $C\cdot w_N^b$, and decreased by $w_N^g + w_N^r$. Hence, the value of R_N can be expressed as follows.

$$R_N = R_0 - (w_N^g + w_N^r) + C\cdot w_N^b.$$

Using Equation (3), this relation can be rewritten to Equation (11).

$$R_N = R_0 - (w_N - w_N^b) + C\cdot w_N^b. \tag{11}$$

The total change in threshold value after N items is $\Delta R_N = R_N - R_0$, so

$$\Delta R_N = R_N - R_0 = -(w_N - w_N^b) + C\cdot w_N^b. \tag{12}$$

Next, the threshold displacement per processed gram is defined to be $\Delta\bar{R}_N = \Delta R_N / w_N$. Combining Equations (5) and (12), we then find Equation (13).

$$\Delta\bar{R}_N = \frac{\Delta R_N}{w_N} = -\frac{(w_N - w_N^b)}{w_N} + C\frac{w_N^b}{w_N} = -(1-\bar{w}_N^b) + C\cdot\bar{w}_N^b. \tag{13}$$

Since we are interested in the long-term average performance of a policy, we also define $\Delta\bar{R} = \lim_{N\to\infty}(\Delta\bar{R}_N)$ to be the long-term average threshold displacement per processed gram, resulting in Equation (14).

$$\Delta\bar{R} = -(1-\bar{w}^b) + C\cdot\bar{w}^b. \tag{14}$$

Furthermore, if we aim to reach a long-term target throughput fraction $0 < q < 1$, a necessary condition for $\Delta\bar{R}$ is that it must approach 0 in the limit. Otherwise if $\Delta\bar{R} > 0$, than $R_N \to \infty$. Similarly, if $\Delta\bar{R} < 0$, than $R_N \to -\infty$. As a result, all items would be rejected or accepted respectively. Hence, we impose $\Delta\bar{R} = 0$ yielding from Equation (14) that

$$C = \left(\frac{1}{\bar{w}^b} - 1\right). \tag{15}$$

Note that the long-term average throughput weight per gram $\bar{w}^b$ follows from the policy used and is not known beforehand. Instead, $\bar{w}^b$ in Equation (15) is replaced by the target throughput q. To satisfy Equation (15) we take:

$$C = \left(\frac{1}{q} - 1\right). \tag{16}$$

We conjecture that under an index policy with a threshold, updated according to Equation (10), the throughput converges to the target q. To support this claim consider again Equation (13). By substituting Equation (16), we obtain:

$$\Delta\bar{R}_N = -(1-\bar{w}^b_N) + \left(\frac{1}{q} - 1\right)\bar{w}^b_N = \left(\frac{\bar{w}^b_N}{q} - 1\right). \tag{17}$$

Clearly, $\Delta\bar{R}_N > 0$ if we have processed N items and $\bar{w}^b_N > q$. In this case the threshold increases, and less items will be accepted. Similarly, $\Delta\bar{R}_N < 0$ if $\bar{w}^b_N < q$. In this case the threshold decreases, and more items will be accepted. Algorithm 1 is now easily modified to include the rejection of items and updates of the threshold to obtain Algorithm 2, which will be referred to as the IPP with Target throughput q (IPPT(q)).

Algorithm 2 The *Index Policy with Power function with Target throughput q (IPPT(q))* algorithm for the K-bounded on-line bin-covering problem with target throughput consists of the following steps. Input α for the loss function in Equation (9). Recursively calculate the indexes $\ell(v)$ for all bin levels $v \in \mathscr{V}$ using Equation (6), given the target weight B. Input a target throughput q and set C according to Equation (16). Initialize the threshold value R_0. Set $N = 0$.
While items arrive do:

1. When an item with weight $w \in \mathscr{W}$ arrives, determine the policy-optimal value ℓ^* using Equation (8). If $\ell^* \geq R_N$ go to step 2. Otherwise go to step 3.
2. The item will be accepted. Calculate R_{N+1} according to Equation (10). Determine the policy-optimal bin k^* using Equation (7). Allocate the item to bin k^*. Empty the bin if $v_{k^*} \geq B$. Set $N := N+1$. Return to step 1.
3. The item will be rejected. Calculate R_{N+1} according to Equation (10). Set $N := N+1$. Return to step 1.

end.

The performance of this algorithm will be studied in Section 5 in a numerical study.

5 NUMERICAL RESULTS

In this section the research questions posed in the introduction will be answered using the simulation model introduced in Section 1. Does using the IPPT(q) algorithm both yield low giveaway and achieve the target throughput q? In order to answer this question, the behavior of the IPPT(q) algorithm introduced in Section 4.2 is illustrated through a simulation study performed in Matlab. In this study we first considered convergence to a target throughput, and how convergence was affected by the threshold R_N. Lastly, the giveaway for different target throughputs and target weights was simulated, to illustrate the impact of reducing the target throughput on the giveaway.

5.1 Evolution of the Dynamic Threshold

In this section we study the convergence of the throughput to a target q using the IPPT(q) algorithm. Let $f(x|\mu,\sigma^2)$ denote the probability density of a Normal distribution, which is discretized in the following manner. An item with weight w is seen with probability $p(w) = f(w|\mu,\sigma^2)/\sum_{x=w_{\min}}^{w_{\max}} f(x|\mu,\sigma^2)$. We used

$\mu = 100$ and $\sigma = 15$, and took $w_{\min} = 1$ and $w_{\max} = 199$ in the experiment. The grader used $K = 8$ bins, the target weight $B = 350$, and target throughput $q = 0.5$. These comprise a setting representative of practice. The initial threshold value $R_0 = 0$. We used Algorithm 2 for allocating products in the grader, and selected the parameter value of the algorithm in $A = 9$ steps. That is, a total of 19 parameter values were evaluated using Algorithm 2 and the one with minimal $\bar{w}_N^g$ at the end of each simulation run was chosen. The order size was chosen to be fixed at 10.000 batches, corresponding to a large customer order thus yielding representative algorithm performance. That is, we generated items until 10.000 batches were completed for each combination of C and α. Firstly, it was investigated how $\bar{w}_N^b$ changed per item N. The results for the first $N = 100$ items using targets $q = 0.25$, 0.5, and 0.75 have been plotted in Figure 3.

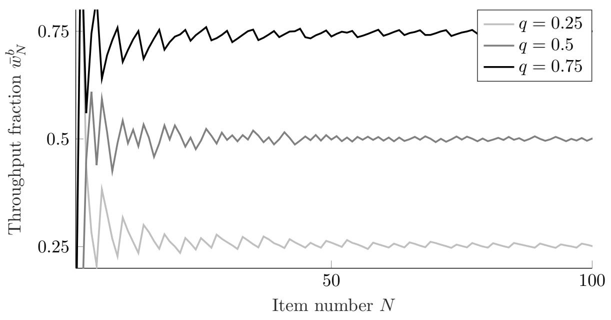

Figure 3: Throughput after N items.

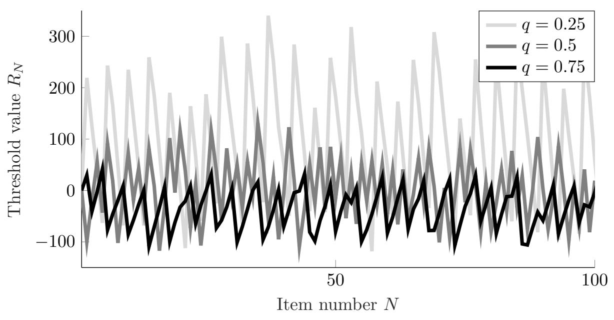

Figure 4: Value of threshold after N items.

One can observe in Figure 3 that for all chosen values of q the throughput approached the target rapidly. In fact, less than 100 items were needed to achieve this behavior. Figure 4 shows how this result was achieved by updating the threshold values using Equation (10). The threshold was increased in reaction to accepted weight, and decreased in reaction to rejected weight. Moreover, the average threshold value was lower for larger values of q, as more weight needed to be accepted on average. Most importantly, these figures suggest that the IPPT(q) algorithm will indeed approach the target throughput for sufficiently large N. In the following, all results shown have a maximum deviation of 0.1% from the target throughput q. Having verified the convergence of the algorithm to the target throughput, we are now ready to analyze its performance under a wider range of settings.

5.2 Giveaway Performance

In Section 5.1 it was shown that it was, in fact, possible to both minimize the giveaway and achieve a target throughput. In this section we will illustrate the performance of the dynamic threshold policy in terms of giveaway for different target throughputs. In particular, we focused on the effect of the throughput targets on the giveaway. We considered two target weights $B = 300$ and 350, which are values often seen in practice. We first determined the throughput for these target weights using IPP. The throughput values using IPP will be referred to as baseline. Next, q was varied from 10% through 90% of the baseline throughput, in steps of 10%, and the IPPT(q) was tested with these values. Each experiment was repeated 10 times in order to obtain confidence intervals (CI). The results are shown in Figure 5.

Due to their small size, the CI are not displayed in Figure 5. Specifically, all 95% CI were within 2.5% of the mean, indicating that a run length of 10.000 filled bins was sufficient to achieve narrow intervals. The figure shows that the more weight was rejected, the lower the giveaway became. Furthermore, it can be seen that the giveaway reduction was approximately constant at each q, showing that rejecting more items reduced the giveaway. However, there was a difference between the target weights. Using the target weight of $B = 350$ yielded a higher giveaway at first, but experienced a larger absolute reduction for lower values of q. In conclusion, it will be harder to achieve low giveaway for some target weights, but that the absolute difference diminishes if more weight is rejected.

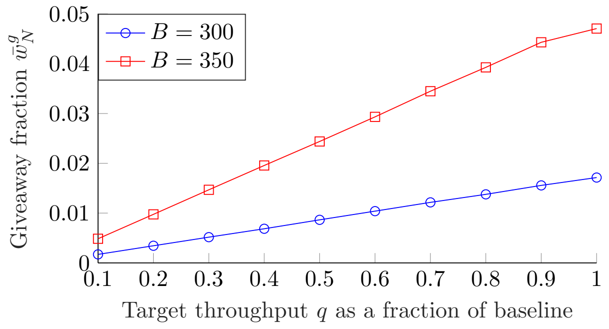

Figure 5: The giveaway for two target weights using various q.

Next, the potential giveaway reduction for different target weights is illustrated in Figure 6. Again the baseline throughput was determined using the standard IPP algorithm, which illustrated the giveaway using different target weights. In turn, we took 75% and 50% of these values as target throughputs using the IPP algorithm with rejection, to illustrate how the giveaway was affected.

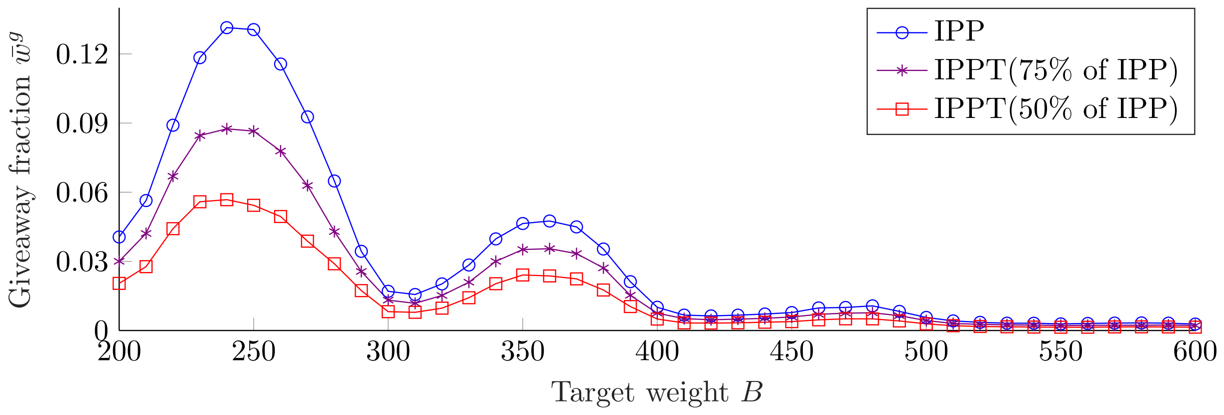

Figure 6: The giveaway fraction $\bar{w}^g$ for various B and target throughputs.

The giveaway was relatively low for target weights that were a multiple of the mean item weight, and relatively high for target weights that were in between these values. Similar behavior was previously documented by Ásgeirsson (2014). Furthermore, it can be observed that the giveaway was lower for higher target weights. At higher target weights more combinations of items can be made, providing index policies with more flexibility. By lowering the target throughput, the giveaway can be reduced by rejecting more items. When the target throughput was lowered to 75% of the baseline, a significant giveaway reduction was observed. Additional giveaway reduction could be achieved when only 50% of the baseline throughput was required. The achieved giveaway curves implies that the performance of a grader strongly depends on the item weight distribution and target weight. Situations where the target weight was approximately a multiple of the mean item weight yielded relatively low giveaway. The same was observed to hold for higher target weights. As a result, it would be desirable to reject more weight when the giveaway is high, and accept more otherwise.

These results illustrate the quality of the algorithm presented, offer companies a method to control the throughput on graders, and provide insight for practitioners when negotiating throughput targets with customers.

6 CONCLUSIONS AND FUTURE WORK

In this paper we developed and studied a control algorithm for a grader. The requirements for this algorithm are to minimize the giveaway, under the condition that some predetermined target throughput is achieved. To this end, a dynamic threshold-based strategy has been proposed, and tested in a simulation study.

Firstly, the convergence of the throughput to a throughput target q was studied, illustrating that the algorithm works as intended. Secondly, the threshold was shown to adapt to accepted and rejected weight in order to achieve the target throughput. Lastly, the effect of lower target throughputs on the giveaway was studied for different target weights. A reduction in giveaway can be achieved at the price of reduced throughput. In conclusion, the algorithm developed in this paper is successful in both reducing giveaway and achieving a target throughput.

Future work will be aimed at comparing different index policies under the proposed rejection threshold method. Secondly, it will be interesting to see if the performance of the IPPT(q) algorithm can be improved further. Lastly, it will be interesting to see if our method for rejection can be extended to bin-packing problems.

REFERENCES

Ásgeirsson, A. 2014. *On-line Algorithms for Bin-covering Problems with Known Item Distributions*. Ph. D. thesis, Georgia Institute of Technology.

Asgeirsson, E. I., and C. Stein. 2006. “Using Markov Chains to Design Algorithms for Bounded-Space On-Line Bin Cover”. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, 75–85. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics.

Asgeirsson, E. I., and C. Stein. 2009. “Bounded-Space Online Bin Cover”. *Journal of Scheduling* 12(5):461–474.

Assmann, S., D. S. Johnson, D. J. Kleitman, and J. Y.-T. Leung. 1984. “On a Dual Version of the One-Dimensional Bin Packing Problem”. *Journal of Algorithms* 5(4):502–525.

Correa, J. R., and L. Epstein. 2008. “Bin Packing with Controllable Item Sizes”. *Information and Computation* 206(8):1003–1016.

Csirik, J., D. S. Johnson, and C. Kenyon. 2001. “Better approximation algorithms for bin covering”. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, 557–566. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics.

Dósa, G., and Y. He. 2006. “Bin Packing Problems with Rejection Penalties and Their Dual Problems”. *Information and Computation* 204(5):795–815.

Manning, L., and R. Baines. 2004. “Globalisation: a Study of the Poultry-Meat Supply Chain”. *British Food Journal* 106(10/11):819–836.

Peeters, K., T. Martagan, I. Adan, and P. Cruysen. 2017. “Control and Design of the Fillet Batching Process in a Poultry Processing Plant”. In *2017 Winter Simulation Conference (WSC)*, edited by W. K. V. Chan, A. DAmbrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 3816–3827. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Weitzman, M. L. 1979. “Optimal Search for the Best Alternative”. *Econometrica: Journal of the Econometric Society*:641–654.

AUTHOR BIOGRAPHIES

KAY PEETERS is a Ph.D. student of the Dynamics & Control research group at the Eindhoven University of Technology. He holds an MSc degree in mechanical engineering from the same university. His Ph.D. project is centered around the optimization of poultry processing plants, with a focus on control and design. His research interests include discrete-event simulation of manufacturing networks, Markov processes and queuing theory, in the context of applied research. His email address is k.peeters@tue.nl.

IVO J. B. F. ADAN is a Professor of the Department of Industrial Engineering at the Eindhoven University of Technology. His current research interests are in the modeling and design of manufacturing systems, warehousing systems and transportation systems, and more specifically, in the analysis of multi-dimensional Markov processes and queuing models. His email address is i.j.b.f.adan@tue.nl.

TUGCE MARTAGAN is an Assistant Professor and Marie S. Curie Research Fellow in the Department of Industrial Engineering at Eindhoven University of Technology. She received her Ph.D. in Industrial Engineering from the University of Wisconsin-Madison. Her research interests include stochastic modeling and optimization of manufacturing systems. Her email address is t.g.martagan@tue.nl.