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.

A HYBRID GENETIC ALGORITHM FOR THE K-BOUNDED SEMI-ONLINE BIN COVERING PROBLEM IN BATCHING MACHINES

Bert H.H. Hundscheid
Kay Peeters

Department of Mechanical Engineering
Eindhoven University of Technology
Groene Loper 15
Eindhoven, 5600 MB, THE NETHERLANDS

Jelle Adan
Tugce Martagan
Ivo J.B.F. Adan

Department of Industrial Engineering
Eindhoven University of Technology
Groene Loper 3
Eindhoven, 5600 MB, THE NETHERLANDS

ABSTRACT

The semi-online bin covering problem is a NP-hard problem that occurs in a batching processes in a high-end poultry processing line. The objective is to form batches of items with minimal giveaway, which is the difference between the target and realized batch weight. The items in this process are allocated in the order of arrival, and the weight of the first set of items is assumed to be known. We develop a novel hybrid genetic algorithm, combining a genetic algorithm and several local search methods. Simulation experiments based on real-world data are performed to gain managerial insights. These simulations suggest that the proposed algorithm produces high quality solutions within a reasonable time limit.

1 INTRODUCTION

In the poultry processing industry broilers are processed to salable items, such as broilers and chicken fillets. Marel Poultry is a company that develops high-end production lines for processing chickens, capable of processing up to 15.000 broilers per hour. Using Marel Poultrys specialized equipment, poultry processing plants produce and package batches of poultry products for distribution to customers. The component in a processing line that creates the batches is called a batcher, of which a schematic overview is shown in Figure 1. The batcher moves items from the processing line into fixed-weight batches. Customers require that each finished batch contains at least a given target weight, for which they pay a fixed price. Consequently, additional weight over the target weight, called giveaway, is not paid by the customer. The objective of the batcher is to minimize the giveaway per batch in order to increase the number of sold batches. As a result, by reducing giveaway, the turnover of a poultry processing plant will increase.

The batcher considered in this paper uses several bins to store uncompleted batches (in Figure 1 there are four bins). Arriving items are placed in bins in order of arrival. The weight of a number of the next arriving items (the horizon) is known. Figure 2a shows how the horizon looks in the batcher. An algorithm allocates items to bins using the weight information of products in the horizon, with the goal of minimizing the giveaway. Figure 2b shows an item that is allocated to a bin. The production line moves at a constant speed, such that the time between arriving products is constant.

The long-term average giveaway per batch depends on the item assignments, the number of bins, the length of the horizon, the target weight per batch, the item weight distribution and the available calculation time to compute an assignment. The number of bins and horizon length determine the footprint of the batching process. The weight distribution depends on the flock weight distribution and the equipment used to process the broilers. The calculation time per product allocation is limited by the inter-arrival time of products. Hence, we must make sure that the calculation time used by the algorithm does not exceed the permitted calculation time.

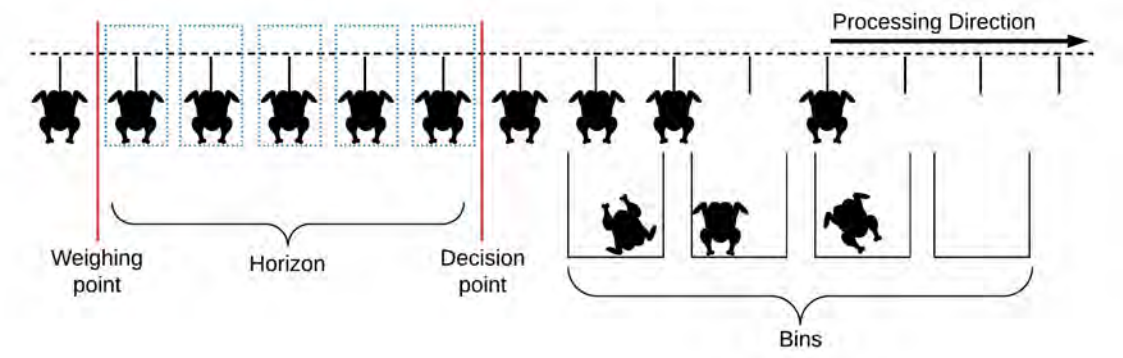


Figure 1: Schematic representation of a batcher from Marel Poultry.

(a) The items in the horizon.

(b) The allocation of an item to a bin.

Figure 2: The real batching processing line from Marel Poultry (Marel 2019).

In this study a Hybrid Genetic Algorithm (HGA) is proposed that uses knowledge of the items in the horizon to minimize the giveaway of the finished batches. A novel aspect of the the HGA is that part of a solution is recycled in following item allocations, and that it learns from past decisions by using Learning Automata (LA). A simulation study using real data from Marel Poultry is performed to get insight in the performance of the HGA for this batching processing line. In the simulation study we provide managerial insights by varying the number of bins, the bin target weights and different item weight distributions. Furthermore, we show the maximal achievable throughput for different HGA parameters. Although this research is conducted in close collaboration with Marel Poultry, the results could be implemented on food batching processing lines with similar characteristics. For instance, the batching of cheese, fruit, vegetable, fish or other meat products.

The remaining part of this paper is organized as follows. In Section 2 a literature review links the batching problem to previous studies and gives background information about the HGA. Section 3 introduces the model of the batching process. Section 4 describes the developed HGA. Section 5 presents the simulation results of the HGA using real-world data. Here, a benchmark problem is introduced for a small batching

problem and a larger batching problem is tested to gain managerial insights. Finally, Section 6 summarizes our findings and identifies opportunities for further research.

2 LITERATURE REVIEW

The batching problem considered in this paper is referred to as the K-bounded semi-online bin covering problem. In a bin covering problem items of given sizes are assigned to bins with a given target weight. The objective is to assign these items such that the number of filled bins is maximized (Csirik et al. 1991). K-bounded refers to the maximum number of unfinished bins that can be used at the same time. Offline problems consider the setting where the weight of all items is known in advance, and the order of arrival may be changed. In online problems, only the weight of the next arriving item is known. In this paper we consider the semi-online case, where the size of the first few items (the horizon) is known (Coffman et al. 1996).

Assmann et al. (1984) introduces the bin covering problem and proves that this problem is NP-hard. Peeters et al. (2017) and Asgeirsson and Stein (2006) introduce algorithms for the online bin-covering problem. Few studies have considered the semi-online algorithms. Ásgeirsson (2014) introduces an allocation strategy for a batching machine with a horizon, called the Multiple Subset Sum Algorithm (MSS). Grove (1995) studies a semi-online bin packing problem with a small lookahead, and assumes that a limited number of items can be stored before allocation.

Inspired by Darwin's theory of evolution, Holland (1992) introduced Genetic Algorithms (GAs), which can be used to solve mathematical and engineering optimization problems. His student Goldberg and Holland (1988) introduced a genetic algorithm that forms the basis for the algorithm proposed in this paper. GAs are powerful algorithms that utilize multiple concurrent search points and do not require the derivative of an objective function (Haupt and Haupt 2004). Moreover, GAs are capable in finding a global optimum by jumping out a local optimum in complex problems. The main disadvantage of GAs is that once the region of optimal solutions has been identified, the convergence towards a better solution slows down (Javadi et al. 2005). By equipping GAs with local search (LS) methods, the GA converges towards an optimal solution more quickly (Hamzaçebi 2008).

To date, there have been no studies that apply a HGA to the K-bounded semi-online bin covering problem. However, some research has been performed on HGAs for bin packing problems. For example, Falkenauer (1996) proposes an offline grouping genetic algorithm enhanced with a local optimization from Martello and Toth (1990), which employs a dominance criterion to reduce the search space.

We conclude that the semi-online bin covering has received little attention, and that HGAs have not been applied to this problem. In order to fill this gap we introduce a HGA for the K-bounded semi-online bin covering problem where only the items in horizon are known, K bins need to be filled and a new (empty) bin becomes available again once a bin has been filled. In a numerical study the HGA implementation will be compared to other algorithms, subject to the calculation time constraint.

3 THE MODEL

In this section the model that represents the batching process will be introduced. The batching process consists of a weighing point, horizon, and batcher (see Figure 1). The items passing through the process undergo a number of steps. Firstly, the items pass the weighing point where the items are weighted. Secondly, the weighted items move into the horizon. The horizon consists of the number of slots between the weighing point and the decision point, where each slot can hold a single item. Thirdly, each time an item passes the decision point a decision is made to which bin to allocate it. Afterwards, this first item leaves the horizon and is placed in one of the bins. Simultaneously, each item in the horizon moves to the next slot and a new item enters the horizon. This allocation cycle repeats for each allocation decision. When the target weight of a bin is reached, the giveaway is calculated and the bin is emptied immediately.

The weighing scale measures the item weight in positive discrete values. The minimum possible weight is denoted by $w_{\min}$ and the maximal possible weight by $w_{\max}$. The set of possible weights is denoted by $\mathscr{W} = \{w_{\min}, w_{\min}+1, \ldots, w_{\max}\}$. The probability that an item has weight w is given by $p(w)$ for all $w \in \mathscr{W}$. Furthermore, the horizon has a constant number of slots N, and it is assumed that all slots contain an item. The items are labeled by their position in the horizon, where the slot nearest to the decision point is labeled 1. Let $\mathscr{N} = \{1,2,\ldots,N\}$ be the set of slots in the horizon. The weight of an item in a specific slot in the horizon is denoted by w_n where $w_n \in \mathscr{W}$ for $n \in \mathscr{N}$. In addition, the batcher has a fixed number of K bins, and the bins are labeled from 1 to K. Let $\mathscr{K} = \{1,2,\ldots,K\}$ denote the set of bins.

When filling a bin $k \in \mathscr{K}$, the bin weight increases till it reaches or surpasses its target weight B. The total weight of items in a bin is within the set $\mathscr{V} = \{0,1,\ldots,B-1\}$. The weight currently in a bin is denoted by v_k, with $v_k \in \mathscr{V}$ for all $k \in \mathscr{K}$. Let $\mathscr{X} = \{X_1, X_2, \ldots, X_N\}$ be the bin assignment for all products in the horizon. Here, each index X_n represents an allocation bin $k \in \mathscr{K}$ for item $n \in \mathscr{N}$. The bin allocation decision is determined by the HGA. Every item that needs to be allocated has its own item identification number for tracking the items in the batching processing line. The first item is numbered 1 and the last item C. Let $\mathscr{C} = \{1,2,\ldots,C\}$ be the set of all items that need to be allocated. An item has an item identification number c where $c \in \mathscr{C}$. At any moment in time the items with identification number c till $c+N-1$ are in the horizon. The item in the first slot in the horizon ($n=1$) is moved to its assigned bin X_1. The weight of this item w_1 is added to the weight v_k of its assigned bin. The assigned bin is emptied if v_k is equal or higher than its target weight B. After the allocation takes place, all items in a slot $n \in \mathscr{N}$ in the horizon move to the next slot $n-1$. Simultaneously, a new item with weight w_N arrives in the last slot of the horizon ($n=N$).

The HGA aims to assign the items in the horizon into the bins such that the total giveaway of the filled batches with the items $n \in \mathscr{N}$ is minimized. The number of possible assignments of N items in the horizon to the K bins equals K^N. The computational time will explode if all possible assignments are evaluated, especially for instances with a large number of items in the horizon or a large number of bins. For these instances the limited available computational time makes a complete enumeration approach unfeasible. In the HGA, the computational time is kept low by only generating a fraction of all possible assignments. These assignments are efficiently generated such that they contain the optimal, or close-to-optimal solution with regards to the objective of this problem.

4 THE HYBRID GENETIC ALGORITHM

In a GA, a population of individuals evolves through a series of steps towards a better solution for an optimization problem. An overview of these steps is shown in Figure 3. In the first step, individuals receive numerical values that represent a candidate solution to the optimization problem (Mitchell 1995). Each candidate solution is encoded as an array of randomly generated parameter values. In the second step, the individuals are ranked and selected. The individuals are ranked according to their fitness, which corresponds to the objective value of the optimization problem. The fittest individuals, called parents, are selected to generate a new population since they have a higher chance to evolve towards a better solution (McCall 2005). Less fit individuals are removed from the population in this step. Thirdly, the cross-over and mutation steps are applied. Cross-over chooses individuals as parents to generate a new population of individuals, called children. Afterwards, these children are mutated to maintain diversity in the population. The evolution of a population is an iterative process where the population in each iteration is referred to as a generation. The GA terminates in the last step if G_{max} generations have been generated. In addition to the GA, several LS methods can be used in each generation to speed up the evolution towards an better solution. In this HGA context, LS_{max} LS steps are applied in each generation. In the next subsections, a detailed description of the HGA designed for the semi-online bin covering problem is presented.

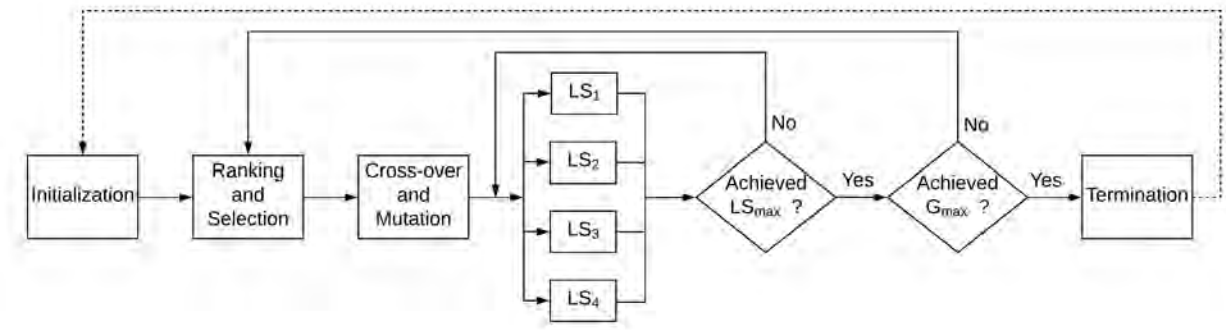


Figure 3: Flow chart of HGA, where LS_1, LS_2, LS_3, and LS_4, represent different local search methods. $LS_{\max}$ and $G_{\max}$ represent the maximum number of local search executions and the maximum number of generations, respectively.

4.1 Representation

In HGA, an individual in the population represents the bin assignment $\mathscr{X} = \{X_1, X_2, \ldots, X_N\}$ where $X_n \in \mathscr{X}$ denotes the parameter value. The number of individuals in a population is given by I. The number of parameter values in an individual is equal to the number of slots in the horizon (N).

4.2 Fitness

The fitness function F is the value of the objective function of the semi-online bin covering problem and represents the quality of a solution for a given individual. The fitness function is taken to be the average giveaway per weight of filled bins, if the solution for an individual were implemented. Implementing the individual with the lowest fitness value is expected to result in a low average giveaway per filled bin.

The fitness value of an individual is derived by virtually allocating the items from the horizon into the bin that corresponds to the individual. That is, the first item in the horizon will be virtually allocated, taking into account the initial bin weights v_k. Then, subsequent items are virtually allocated until a bin is filled, after which it is emptied. This process continues until all items in the horizon are processed in this manner. The number of filled bins from the items in the horizon is given by Q, and the set of filled bins is denoted by $\mathscr{Q} = \{1, 2, \ldots, Q\}$. The weight of a filled bin is given by U_q for all $q \in \mathscr{Q}$. The fitness value of the individual is determined by Equation (1).

$$F = \frac{\sum_{q=1}^{Q} \left[U_q - B\right]}{\sum_{q=1}^{Q} U_q} \tag{1}$$

4.3 Initialization

In the initialization step the initial population is generated for use with the HGA. In this paper, a distinction is made between two initialization cases. The first case will be used if the HGA has to allocate the first item ($c = 1$). In this case, all individuals are randomly generated for the initial population. The individuals in this initial population are represented as $\mathscr{X} = \{X_1^R, X_2^R, \ldots, X_N^R\}$. The superscript R of a parameter value X_n^R with $n \in \mathscr{N}$ indicates that a parameter value is randomly generated from the uniform distribution on the set of available bins. The second initialization case will be used if the HGA has to allocate items after the first item ($c > 1$). In this case, the fittest individual ($\mathscr{X} = \{X_1, X_2, \ldots, X_N\}$ where $X_n \in \mathscr{X}$) that was used to allocate item c is reused to create the first individual in the initial population. The first parameter value X_1 of this individual is removed, since the corresponding item c has been allocated. In addition, a random bin is added to the end of this individual to provide a bin allocation for the newly arrived item

in the horizon. This new individual is given by $\mathscr{X} = \{X_2, X_3, \ldots, X_N, X_{N+1}^R\}$. In addition to the partially preserved individual, a number of individuals with randomly generated parameter values is added till the population reaches its initial population size. Those individuals are represented by $\mathscr{X} = \{X_1^R, X_2^R, \ldots, X_N^R\}$

4.4 Ranking and Selection

Each time a population is evaluated, the fitness of all individuals is calculated and the individuals are ranked from lowest to highest fitness value. Then, the I_{best} individuals with the lowest fitness are selected from the ranked population. Those selected individuals, called parents, are used for generating children to breed out a new generation. The remaining individuals with a higher fitness value are removed from the population.

In addition to the above ranking and selection strategy, an elitist selection strategy is implemented. Elitism ensures that the individual with the lowest fitness in the population remains unaltered when moving to the next generation. This individual is referred to as the elitist individual. Elitist selection guarantees that the fitness between following generations cannot decrease (Baluja and Caruana 1995). In addition, this strategy saves computational time because it does not have to re-discover discarded individuals (Adan et al. 2018).

4.5 Cross-over and Mutation

The parents selected in the ranking and selection step will be used generate children using cross-over and mutation. Cross-over and mutation change the parameter values in an individual to obtain a individual with a lower fitness. Next, cross-over is explained, followed by mutation.

In the cross-over step, a one-point crossover mechanism (Bäck et al. 2018) is used. Using this mechanism, a child is generated by combining the genetic information of two randomly selected parents. The genetic information of a parent consists of all parameter values in its individual. For each pair of parents, one random cross-over point is generated. Then, the cross-over point splits the genetic information of both parents in two parts. The first part of the child consist of the genetic information of the first part from one parent. The second part of the child consist of the genetic information of the second part of the other parent. The length of a child is again equal to N. In the next generation, the selected elitist individual is copied. Then, children are generated until the same population size as the initial population size is reached. Consequently, the new population consists out of children and the elite individual. Since the new population is generated from individuals with low fitness, the average fitness is expected to improve.

When solely fit individuals are used in the cross-over step, the diversity of the genetic information in the new population decreases. Without genetic diversity, the GA can remain trapped in a local optimum. For this reason mutation is introduced. The mutation applies M mutations to each child in the generation. Each mutation changes a bin allocation for a random item in the individual into a random bin allocation. Next, the new population moves to the local search step.

4.6 Local Search

In addition to the genetic algorithm, various Local Search (LS) methods are implemented. The aim of a local search step is to try to reduce the fitness value of a selected individual. Different local search methods are used until a maximum of $LS_{\max}$ LS applications have been used in a generation. The number of possible LS methods is J, and the set of local search methods is denoted by $\mathscr{J} = \{1, 2, \ldots, J\}$. LS method j is denoted as LS_j with $j \in \mathscr{J}$. The following searches have been implemented:

- LS_1: Change the bin allocation of the last item added to the fullest bin.
- LS_2: Swap the bin allocation of a random item n in the fullest bin with the bin allocation of item $n-1$ in the horizon.
- LS_3: Randomly change the bin allocation of a random item.
- LS_4: Randomly change the bin allocation of a random item in the fullest bin.

The probability by which a LS method is used is determined by a learning automata. The LA adapts the probability to execute an LS method based on its effectiveness, after allocating an item to a bin. An execution of an LS method is effective if it reduces the fitness value of an individual. The number of executions as well as the number effective executions for a specific LS method during all generations for an item is determined. Consider an LS method $j \in \mathscr{J}$ after an item $c \in \mathscr{C}$. The number of executions of the method is given by $LS_{j,c}^{\text{total}}$. The number of effective executions is denoted by $LS_{j,c}^{\text{effective}}$. The effectiveness of the LS method $Z_{j,c}$ is given by Equation (2). The probability of executing method LS_j for item c is calculated by Equation (3). For the first product, each LS method is selected with equal probability.

$$Z_{j,c} = \frac{LS_{j,c}^{\text{effective}}}{LS_{j,c}^{\text{total}}} \tag{2}$$

$$P_{j,c} = \begin{cases} \frac{1}{J} & \text{if } c = 1 \\ \frac{P_{j,c-1}+Z_{j,c-1}}{\sum_{i=1}^{J}[P_{i,c-1}+Z_{i,c-1}]} & \text{if } c > 1 \end{cases} \tag{3}$$

4.7 Repetition and Termination

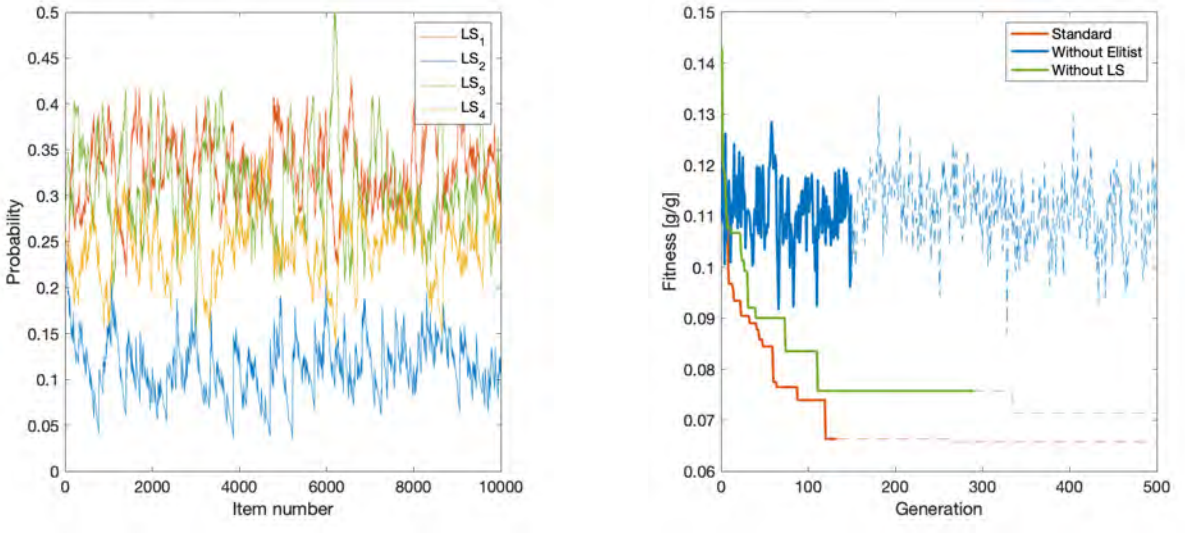
After the first generation and $LS_{\max}$ local search attempts, a population of new individuals is obtained. Next, this population will be ranked, selected and modified by the cross-over step, mutation step and the local search methods until $G_{\max}$ generations are performed. Then, the HGA is terminated and an the elite individual with the lowest fitness value is selected. The first parameter value (X_1) of this individual is used to allocate the first item in the horizon into the corresponding bin. Next, this individual is reused for starting the initialization step of the HGA for the next product, as described in Section 4.3. The HGA is terminated when all items $c \in \mathscr{C}$ have been allocated.

5 SIMULATION RESULTS

In this section the model introduced in Section 3 is used to obtain insights in the performance of the HGA described in Section 4. First, the performance of the LS methods and corresponding fitness were analyzed. Then, a small problem was considered to obtain simulation results for the HGA, Complete Enumeration (CE) method and the MSS algorithm. Furthermore, a large (realistic) problem was introduced for analyzing the performance of the HGA. For the large problem instance we ran simulations with varying number of bins, varying bin target weights and different weight distributions for the items. These simulations were performed using data of broiler weights provided by Marel Poultry. The data showed that the broiler weights could be represented as a Normal distribution $\mathscr{N}(\mu,\sigma)$, and was scaled to the mean $\mu = 100$ grams and a standard deviation $\sigma = 15$ grams. The weight distribution was scaled for confidentially reasons. As mentioned in Section 1, the maximum throughput of a processing line is 15.000 items per hour. As a result, each item needs to be allocated within 240 milliseconds. The HGA was coded in Mathworks Matlab 2018b and simulated on a computer with an Intel Core i5 processor running at 2.7 GHz and 8 GB RAM. All simulations allocated $C = 10.000$ items to reduce start-up effects and to obtain representative simulation results. The simulations with an identical broiler weight distribution used the same weights for each broiler $c \in \mathscr{C}$.

5.1 Performance local search and fitness

In Figure 4a, the probability of selecting a local search method in the HGA for each item is plotted. The figure shows that local search methods LS_1 and LS_3 gave the highest probability to improve the fitness value. Interestingly, local search method LS_3 that randomly changes the bin allocation of an item, performed relatively well. This may due to the exploration of different local or global optima. Figure 4b shows the fitness of the individual with the lowest fitness value per generation for the first item ($c = 1$) that needed to



(a) Probability to execute a LS-method per item.

(b) Fitness per generation for the first item ($c = 1$) with different methods.

Figure 4: LS and fitness in the HGA.

be allocated. This figure shows the impact of using local search methods and elitist selection in the HGA. The fitness plot shows: the HGA including LS and elitism (Standard), the HGA including LS only (Without Elitist), and the HGA including elitism only (Without LS). The bold part of each curve presents the fitness value achieved within 100 milliseconds. The figure shows that the HGA without the elitist selection was not able to effectively reduce the fitness between generations. The HGA without elitism and the HGA without LS processed more generations than the HGA including both, using the same calculation time. However, the latter was eventually able to achieve a significantly lower fitness, illustrating the effectiveness of adding elitism and local search.

5.2 Small problem

Next, a small problem with $K = 2$ bins and a horizon of $N = 15$ items was considered. For this problem it was possible to evaluate, within a reasonable time frame, every feasible allocation of all items in the horizon. From these feasible allocations we used the one yielding the minimal fitness, where ties were resolved arbitrarily. This heuristic is named the Complete Enumeration (CE) method and will be used as benchmark. In this setting there are K^N (32.768) possible allocations. Several simulations were performed to gain insight in the variance of the giveaway per filled bin. The number of CE simulations was determined such that the 95% confidence interval was within 2% of the mean giveaway per filled bin, for target weights $B \in \{200, 250, \ldots, 600\}$. 15 CE simulations were needed to achieve this goal. The MSS, introduced by Ásgeirsson (2014), was tested by running the same simulations as for CE method. The parameters for the HGA simulation were set through manual experimentation. The parameters for the first HGA simulation (HGA with G=500) were: $C = 10.000$, $N = 15$, $K = 2$, $G = 500$, $I_{\text{best}} = 5$, $I = 10$, $M = 12$, $LS_{\max} = 1$ and $B = \{200, 250, \ldots, 600\}$. In the second simulation of the HGA (HGA with G=2000), the number of generations G was increased to 2000 to get insight in the quality of the achieved solution in case more generations per item were performed. Again, 15 simulations for each target weight in the HGA with G=500 and the HGA with G=2000 were performed. Table 1 shows the average giveaway per target weight and algorithm as well as the range of the 95% confidence interval. The 95% confidence interval is given after the $\pm$-sign.

Table 1: Average giveaway per target weight (gram) achieved by different heuristics, including the 95% confidence intervals.

Target weight	CE	MSS	HGA G=500	HGA G=2000
200	19.0±0.00	26.8±0.07	19.0±0.01	19.0±0.01
250	40.5±0.03	46.3±0.14	40.8±0.05	40.6±0.08
300	12.7±0.00	23.3±0.21	12.7±0.02	12.7±0.00
350	23.0±0.00	37.3±0.24	23.4±0.09	23.1±0.07
400	9.79±0.05	21.8±0.15	9.78±0.07	9.70±0.05
450	12.0±0.00	29.3±0.21	11.9±0.07	11.9±0.09
500	7.74±0.08	20.8±0.17	7.38±0.10	7.52±0.11
550	6.91±0.09	24.3±0.09	6.47±0.09	6.56±0.09
600	5.79±0.11	20.0±0.01	5.42±0.10	5.63±0.05

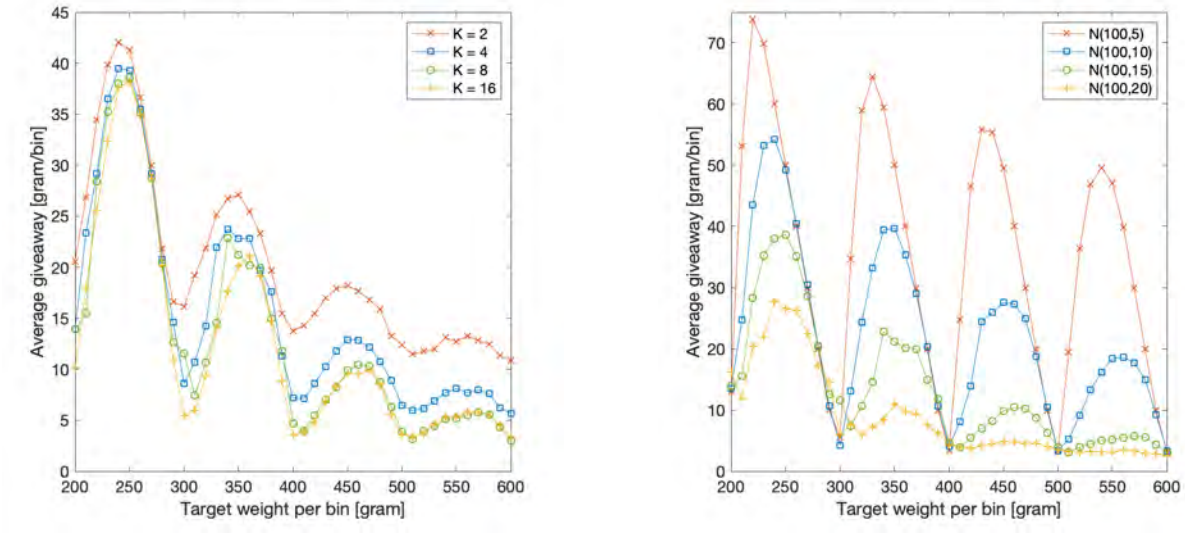
Table 1 shows that both HGA instances performed significantly better than the MSS algorithm. The gap between the CE and HGA with G=2000 results was very small. Moreover, the results of the HGA with G=500 were very close to the results of HGA with G=2000. The average calculation time to allocate an item for the CE, MSS, HGA with G=500 and HGA with G=2000 was 13.2, 0.732, 220 and 650 milliseconds respectively. The allocation time for the CE and MSS method were relatively low compared with both HGA algorithms. However, the computational time of the CE algorithm explodes if the items in the horizon or the number of bins increases, and the giveaway of MSS was much higher than that of the HGA. The HGA achieved low giveaway and its calculation time scales much better with the problem size. The HGA with G=2000 had a calculation time of 650 milliseconds which exceeded the maximum calculation time of 240 milliseconds per item. However, the average calculation time to allocate an item in the HGA with G=500 simulations was 220 milliseconds, which was below the maximum calculation time.

5.3 Large problem

Next, simulations were executed to compare the performance of the algorithm for a larger problem in instances with various target weights, various number of bins and various item weight distributions. Figure 5a shows the average giveaway per filled bin for $B = \{200, 210, \ldots, 600\}$ in grams for $K = \{2, 4, 8, 16\}$. The other parameters for these simulation runs were: $C = 10.000$, $N = 100$, $G = 500$, $I_{\text{best}} = 5$, $I = 10$ and $M = 12$, $LS_{\max} = 1$. The item weight distribution $N(100, 15)$ was used for this simulation. Figure 5b illustrates the performance of the HGA algorithm with various item weight distributions. The same parameters were used to generate Figure 5a, with the exception of the number of bins and the item weight distribution. In this simulation the number of the bins was set to $K = 8$ and the item weight distribution used was a normal distribution $N(100, \sigma)$ with standard deviations $\sigma = \{5, 10, 15, 20\}$ in gram.

Figure 5a shows a decrease in average giveaway per bin when the number of bins or the target weight increased. This is explained by the increased freedom of the HGA in selecting bins and composing batches. In addition, the giveaway was lower for bins with a target weight that was close to a multiple of the mean item weight. Figure 5b also shows small differences in giveaway for the different item weight distributions at these points. However, increasing the variance of the items weight distribution resulted in a lower average giveaway overall.

The average calculation time to allocate an item for 2, 4, 8 and 16 bins was 220, 240, 270 and 310 milliseconds respectively. Here, the average calculation time to allocate an item using 8 bins was just above the maximum calculation time, but yielded a giveaway that was almost equal to the 16 bins case. This suggests that the 8 bins case creates sufficient freedom for the HGA in selecting bins and composing batches. The calculation time was extremely dependent on the settings of the HGA parameters, the performance of



(a) Average giveaway for various target weights and number of bins.

(b) Average giveaway for various target weights and item weight distribution.

Figure 5: Average giveaway with the HGA.

the computer, the used software and the programmed code to execute the algorithm. With this in mind, the calculation time could be reduced by improving these four points.

6 CONCLUSIONS AND RECOMMENDATIONS

This study set out to fill the knowledge gap of applying a Hybrid Genetic Algorithm (HGA) to the K-bounded semi-online bin covering problem. A HGA was developed and applied to a problem occurring in the poultry processing industry. The problem involves filling batching with a given target weight, where the goal is to minimize the giveaway. A simulation study was performed using data from Marel Poultry to evaluate the performance of this algorithm in different settings.

First, a scenario with 2 bins and a small horizon of 15 items was considered. There, the HGA, MSS, and Complete Enumeration (CE) were compared. The comparison showed that the difference between CE and HGA was at most 6.4%, and that the HGA significantly outperformed MSS. In the second scenario, the number of bins, item weight distributions, and target weights were varied. No noticeable improvements were found by using more than 8 bins. Furthermore, the performance of HGA was better for item weight distributions with a higher variance and larger target weights, especially for target weights that were multiples of the mean item weight. In our study we showed that two local search methods yielded the highest chance of reducing the fitness: changing the bin allocation for a random item, and changing the bin allocation of the item last added to the fullest bin. Moreover, HGA with elitist selection and local searches quickly reduced the fitness compared to a HGA without LS methods or elitist selection. Last, the HGA calculation time was determined. This calculation time was highly dependent on the HGA parameter settings, the performance of the computer, the used software and the programmed code to execute the HGA. The average calculation time to allocate a single item of a batching processing line with a horizon consisting of 100 items and 8 bins was 270 milliseconds. This calculation time is slightly higher than the maximal allowed calculation time of 240 milliseconds.

Further research should be aimed at improving the speed of the HGA in order to achieve a lower fitness in the same calculation time. This can be accomplished by more efficiently implementing the HGA or by implementation in other programming languages. In addition, the performance of HGA may be improved

by extending the fitness function. For instance, the fitness function may be improved by including the items in the horizon that were not allocated to a filled bin, or by giving the first items in the horizon more weight in fitness value. Lastly, it would be interesting to investigate the impact of the horizon length, since it affects the footprint of the batching processing line.

REFERENCES

Adan, J., I. Adan, A. Akcay, R. Van den Dobbelsteen, and J. Stokkermans. 2018. "A Hybrid Genetic Algorithm for Parallel Machine Scheduling at Semiconductor Back-end Production". In *Twenty-Eighth International Conference on Automated Planning and Scheduling*, 298–302. Palo Alto, California: Association for the Advancement of Artificial Intelligence.

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

Asgeirsson, E., 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 & Experiments*, 75–85. Philadelphia, Pennsylvania: Society for Industrial and Applied Mathematics.

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

Bäck, T., D. B. Fogel, and Z. Michalewicz. 2018. *Evolutionary Computation 1: Basic Algorithms and Operators*. New York: CRC press.

Baluja, S., and R. Caruana. 1995. "Removing the Genetics from the Standard Genetic Algorithm". In *Machine Learning Proceedings 1995*, 38–46. Tahou City, California: Elsevier Inc.

Coffman, E. G., M. R. Garey, and D. S. Johnson. 1996. *Approximation Algorithms for Bin Packing: A Survey*. Boston: PWS Publishing.

Csirik, J., J. Frenk, G. Galambos, and A. R. Kan. 1991. "Probabilistic Analysis of Algorithms for Dual Bin Packing Problems". *Journal of Algorithms* 12(2):189–203.

Falkenauer, E. 1996. "A Hybrid Grouping Genetic Algorithm for Bin Packing". *Journal of Heuristics* 2(1):5–30.

Goldberg, D. E., and J. H. Holland. 1988. "Genetic Algorithms and Machine Learning". *Machine Learning* 3(2):95–99.

Grove, E. F. 1995. "Online Bin Packing with Lookahead". In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, 430–436. San Francisco, California: Symposium on Discrete Algorithms.

Hamzaçebi, C. 2008. "Improving Genetic Algorithms Performance by Local Search for Continuous Function Optimization". *Applied Mathematics and Computation* 196(1):309–317.

Haupt, R. L., and S. Haupt. 2004. *Practical Genetic Algorithms*. 2nd ed. Hoboken, New Jersey: Wiley Online Library.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. London: MIT Press.

Javadi, A. A., R. Farmani, and T. P. Tan. 2005. "A Hybrid Intelligent Genetic Algorithm". *Advanced Engineering Informatics* 19(4):255–262.

Marel 2019. "Marel Poultry: Secondary Broiler Processing". https://www.marel.com/poultry/, accessed 25th March.

Martello, S., and P. Toth. 1990. "Lower Bounds and Reduction Procedures for the Bin Packing Problem". *Discrete Applied Mathematics* 28(1):59–70.

McCall, J. 2005. "Genetic Algorithms for Modelling and Optimisation". *Journal of Computational and Applied Mathematics* 184(1):205–222.

Mitchell, M. 1995. "Genetic Algorithms: An Overview". *Complexity* 1(1):31–39.

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.

AUTHOR BIOGRAPHIES

BERT H.H. HUNDSCHEID is a M.Sc. Manufacturing Systems Engineering student of the Dynamics and Control research group at the Department of Mechanical Engineering at the Eindhoven University of Technology. His current research interests are manufacturing systems optimization, sustainable manufacturing and sustainable energy. His email address is berthundscheid@gmail.com.

KAY PEETERS is a Ph.D. student of the Dynamics and Control research group at the Eindhoven University of Technology. He holds an M.Sc. 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.

JELLE ADAN is a Sr. Business Process Analyst at Nexperia Industrial Technology Engineering Centre (ITEC) and a doctoral candidate at the Department of Industrial Engineering of the Eindhoven University of Technology, where he obtained his M.Sc. degree in Chemical Engineering in 2017. His current research interests are supply chain, manufacturing and chemical process optimization, and data mining. His email address is jelle.adan@protonmail.com.

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

IVO J.B.F. ADAN is a Professor at the Department of Industrial Engineering of 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 queueing models. His email address is i.adan@tue.nl.