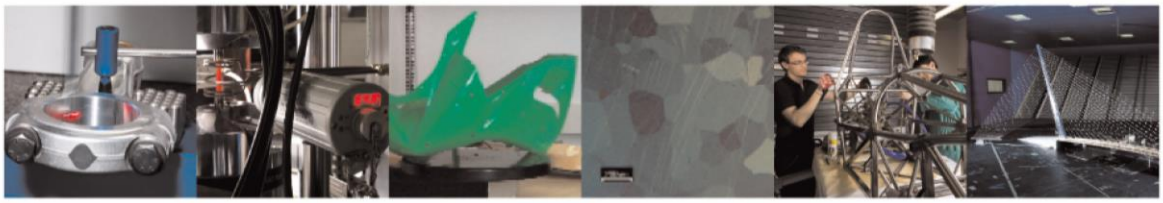
POLITECNICO
MILANO 1863

DIPARTIMENTO DI MECCANICA

mecc



On the Multihead Weigher Machine Setup Problem

Beretta, A., Semeraro, Q., and Castillo, E.

DIPARTIMENTO DI MECCANICA ■ POLITECNICO DI MILANO
via G. La Masa, 1 ■ 20156 Milano ■ EMAIL (PEC): pecmecc@cert.polimi.it
http://www.mecc.polimi.it
Rev. 0

On the Multihead Weigher Machine Setup Problem

Alessia Beretta* and Quirico Semeraro†
Dipartimento di Meccanica
Politecnico di Milano, 20156 Milano (MI), Italy
Enrique del Castillo‡
Department of Industrial and Manufacturing Engineering
The Pennsylvania State University, University Park, PA 16802, USA

Graphical Table of Contents

A multihead weigher machine is a computer-controlled machine used to fill a package with small products or parts with a given target weight. The initial setup can have a dramatic effect on the machine performance and it is still an open problem in industry. After the formalization of this problem and its main variables, we tackled it with different optimization techniques. Figure 1.

Abstract

A multihead weigher machine (MWM) is a computer-controlled machine used to fill a package with small products or parts with a given target weight. MWMs are widespread in the food and non-food industries due to their high performance in terms of variety of products to be packed, high throughput, and quality of the packaged items. Despite the popularity of these machines, studies aimed at how to setup their settings are lacking. In industrial practice, operators currently use ad-hoc rules to setup a MWM, but the initial settings can have a dramatic effect on the machine performance. The aim of this paper is to formalize the setup problem and its main variables. Then an optimization technique is proposed to solve the problem.

Keywords: Combinatorial weighing machine, Optimization, Simulation, Cost reduction.

Notation

α proportionality constant.

*Corresponding author. Dr. Beretta is Postdoctoral Research Fellow in Mechanical Engineering. e-mail: alessia.beretta@polimi.it

†Dr. Semeraro is Professor in Mechanical Engineering. e-mail: quirico.semeraro@polimi.it

‡Dr. del Castillo is Distinguished Professor of Industrial & Manufacturing Engineering and Professor of Statistics. e-mail: exd13@psu.edu

C_0 (€cent) fixed cost in a fixed time period.

C_S (€cent/pkg) sorting cost per "non conforming" package.

C_U (€cent/g) cost of the raw material.

$C_{T,u}$ (€cent/pkg) production cost per "conforming" package.

$\mathbb{E}(W|W \geq T)$ (g/package) expected weight value of the "conforming" packages.

H number of the machine hoppers.

M maximum capacity of a hopper.

n_C number of "conforming" packages in a fixed time period.

p probability that a package has a weight equal or greater than the target weight.

P number of packages produced in a fixed time period.

T (g) target weight of a package.

W (g/pkg) amount of product inside a package. It is modeled as a random variable with an unknown distribution.

W_C (g/pkg) amount of product inside a "conforming" package. $W_C = (W|W \geq T)$

W_{Ci} (g/pkg) amount of product inside the ith "conforming" package.

W_j (g/pkg) amount of the product inside the jth hopper. It is modeled as a normal random variable with mean μ_j and variance σ_j^2

1 Introduction

The rapid development of the packaging industry has motivated the manufacturers of packaging machines to propose new versatile, productive and highly automated solutions. Among the solutions oriented to the packaging of small discrete items, the multihead weigher machine has increasingly been used due to its high speed, accuracy and reliability. A multihead weigher machine (hereafter a MWM, sometimes called a combinatorial weighing machine) is a computer-controlled machine used to fill a package with small products or parts at a given target weight.

Since its first appearance in the marketplace in the seventies, MWMs have been continuously improved to meet the ever-changing market requirements and consumer tastes. Today, this machine has a wide range of applications in food industry for dried foods such as pasta, pet foods, coffee, cereals, a wide variety of candies, or even fresh foods such as vegetables, poultry pieces, etc. Its applications cover also the packaging of non-food items, for instance, packages of paper clips and other small office items are filled using this kind of machines. Among the multihead weigher manufacturers, the one with the world leading position has about 31,000 MWMs installed all over the world. Despite its worldwide spread

and the high number of manufacturers, there are few scientific studies aimed at studying its characteristics and most of the literature is mainly technical or commercial (e.g. patents, machine brochures, etc.).

A MWM is mainly composed of a system of feeders, a set of H pool hoppers, a set of H weight hoppers and a discharge chute to the packaging machine as depicted in Figure 1. The product is continuously fed via a central dispersion feeder (usually a vibrating cone) and H radial feeders (vibrating channels) to the pool hoppers. The role of the pool hoppers is to stabilize the product before dropping it into the weight hoppers. At the startup of a new lot, the operator has to set the product flow rate for each radial feeder and, since the radial feeders are independent, the software allows the operator to choose a different flow rate for each of them. This setting may change depending on the type of product to be packed and the target weight of the package. Each weight hopper is equipped with a load cell that weighs the product and transmits the information to a computer. The computer then selects a subset of hoppers whose total weight is equal to or greater than the target weight T (i.e., the weight value labeled on the package). Then, the computer opens the selected hoppers releasing the product through the discharge chute into the downstream packaging machine. For customer protection, the law requires that the weight of each package must be no less than the target weight. Consequently, a package filled with a quantity of product below the target weight is defined "non conforming" and cannot be sold in the market.

The machine works in a cyclic way with a quite short fixed cycle time. For instance, some machines are able to produce about ninety packages per minute. There are more complicated machines offered in the market than the ones previously described. For instance, a double-layered machine has a further set of "booster" hoppers (usually located below the weight hoppers) which increase the number of possible hopper combinations among which the computer algorithm can select the best one to open. Moreover, a more complex operation strategy can be implemented. For instance, for perishable food items, when a specific hopper remains closed for a certain number of cycles, the system can release it into a trash box or can force it to belong to the next hopper combination to open in order to avoid excessive waiting time and spoilage. In this paper, we focus the analysis on a single-layer MWM (Figure 1).

Summing up, MWM is a complex machine which needs a setup strategy and a suitable operation software. The control software works in real time and its goal is to select the best hopper subset to open in order to achieve the package target weight, according to the product, the cycle time constraint and the objective function. This problem is equivalent to the well-known knapsack problem [1]. In the literature, different knapsack problem formulations have been proposed: most of them utilize linear objective function(s) and linear constraints [2, 3, 4, 5]. Karuno et al. [2] proposed to use dynamic programming because their interest is to speed up the selection phase of the best hopper subset. Among many scientific studies focused on different weighing machines, Silvestrini at al. [6] proposed an algorithm to check if a component is missed in a package, this could be helpful when the MWM is used to multiproduct packages.

In this paper we want to tackle the setup strategy which is still an open problem. The setup problem of a MWM requires the determination of the optimal flow rate of each ra-

dial feeder. Currently, the setup procedures adopted in industrial practice mainly rely on the operator's skill and experience during a trial-and-error manual setup which does not guarantee the best performance. To the best of our knowledge, the setup problem has not been addressed in the scientific literature apart from some preliminary results discussed in [5] and [7].

The rest of the paper is organized as follows. In section 2 the definition of the setup problem and its main variables together with the expression of the economic objective function are presented. In section 3, an analysis of the Solution Space is presented. According to the characterization of the Solution Space, section 4 presents different algorithms to tackle the setup problem. In section 5, the results of the optimization are shown and an optimization method is proposed. The paper ends with recommendations and directions for further research.

2 Problem Definition

The weight of the raw material inside the jth weight hopper, that is the product of the radial feeder flow rate and the cycle time, changes at each machine cycle due to random fluctuation. By analyzing real data [5], it has been observed that this weight is a realization of the random variable W_j, normally distributed with mean μ_j ($\mu_j > 0$) and variance σ_j^2. Moreover, it has been observed that the standard deviation σ_j depends on the mean quantity μ_j according to the following linear relation:

$$\sigma_j = \alpha \mu_j \tag{1}$$

where α ($0 < \alpha < 1$) is an experimental constant depending on the product to be packed. This relation was also pointed out by [8].

A machine setup consists of specifying the values of $\mu_j, j = 1, ..., H$ and, as a consequence, $\sigma_j^2, j = 1, ..., H$ for a given package target weight T. Once the machine is set, the MWM starts to pack bags of product, solving a knapsack algorithm per bag. This setting has to be done each time the type of raw material (product) changes or when a different target weight T is required. An improper selection of the machine setup affects the machine efficiency in terms of "non conforming" rate, material cost, scrap or rework cost and possible losses due to the deviation of the product performance from customer's and/or producer's target. Thus, the initial setting of the machine is a very important decision affecting the general economic performance. Our goal is to propose a standard procedure to determine the best machine setup $\boldsymbol{\mu}^{opt} = (\mu_1, \mu_2, ..., \mu_H)$ for a specific machine and product according to a suitable economic objective function as explained in the next section.

2.1 The economic objective function

As previously outlined, an improper machine setup turns into a lack of machine efficiency. In fact, setting the machine to have the mean of the package weight at a very low level

reduces the production cost but increases the rejection cost due to more "non conforming" packages. On the other hand, setting the machine to a very high level reduces the rejection cost but increases the material cost.

In the economic objective function the quantity and the quality of the packages filled must be jointly considered to evaluate the costs. The quantity can be expressed by the throughput (number of packages per time unit). The quality is expressed by the number of "non conforming" packages. These packages are usually broken and the raw material reutilized. Sometimes a package cannot be reworked but has to be scrapped, for instance when perishable products are handled.

To approach this problem we propose an objective function that is easier to understand by the MWM users: the expected value of the production cost per "conforming" package in a fixed time period. The idea is to allocate the total production cost only to the packages that are sold (n_C). Let P be the number of packages filled in a fixed time period, let C_0 be the fixed cost incurred in the time period, such as depreciation cost, energy cost and packaging material cost, let C_U be the raw material cost per gram and let C_S be the sorting cost to take care of each "non conforming" package. If a package can be reworked, this cost term takes into consideration that the operator has to open the package and to tip out the content into the dispersion feeder. Otherwise, if a package has to be scrapped, this cost item considers the time used by the operator to take care of it. In the following we will assume that a "non conforming" package can always be reworkable, then the production cost per "conforming" package ($C_{T,u}$) is formulated as follows:

$$C_{T,u} = \frac{C_0}{n_C} + C_S \frac{(P - n_C)}{n_C} + C_U \frac{\sum_{i=1}^{n_C} W_{Ci}}{n_C} \tag{2}$$

where:

- n_C is the number of "conforming" package filled in the time interval modeled as binomial density function;
- $W_C = (W|W \geq T)$ is the random variable that characterized the amount of product filled in each "conforming" package;
- W_{Ci} is the realization of the random variable W_C at the ith "conforming" package.

Let's set:

$$\frac{\sum_{i=1}^{n_C} W_{Ci}}{n_C} = \overline{W}_{C\bullet} \tag{3}$$

Since

$$\mathbb{E}[\overline{W}_{C\bullet}] = \mathbb{E}\left[W|W \geq T\right] \tag{4}$$

the expected value of the production cost per "conforming" package is expressed as follows:

$$\mathbb{E}[C_{T,u}] = C_0 \mathbb{E}\left[\frac{1}{n_C}\right] + C_S \; P\mathbb{E}\left[\frac{1}{n_C}\right] - C_S + C_U \mathbb{E}\left[W|W \geq T\right] \tag{5}$$

It is easy to find that as the number of packages filled in a fixed time period increases, the following approximation is true:

$$\mathbb{E}\left[\frac{1}{n_C}\right] \approx \frac{1}{P \cdot p}$$

where $p = Pr\,(W \geq T)$.

Since p and $\mathbb{E}\,[W|W \geq T]$ depend on the setting parameters $\boldsymbol{\mu}$, the optimal setup could be found solving the following optimization problem:

$$\underset{\boldsymbol{\mu}\in\mathbb{S}}{\text{minimize:}} \qquad \mathbb{E}[C_{T,u}] = \frac{C_0}{P \cdot p} + C_S\frac{1-p}{p} + C_U\mathbb{E}\,[W|W \geq T] \tag{6}$$

where:

$\mathbb{S}$ is the Solution Space

$$p = Pr\,(W \geq T) \tag{7}$$

W is an unknown probability function whose parameters depend on $(\boldsymbol{\mu}, T, \alpha)$ (8)

Analyzing (6), the first two terms are always minimized by maximizing p. To increase p, a package can be filled with an amount of product greater than T, but as a consequence the third term of (6) gets bigger as the "give away" product increases.

The optimization of the problem (6) maximizes the probability to produce conforming packaging with a weight as close as possible to T. In this way the producer is protected against the possibility to produce "big packs" (i.e. packages with a weight very larger than T) because they would be very expensive. As a matter of fact, the value of the economic objective function gets bigger and bigger as the package weight increases.

Unfortunately, an analytical approach to address the optimization problem is unfeasible because the distribution of W is unknown and the relationships (7) and (8) are unknown. Thus, a simulator of the machine has been built and a Monte Carlo procedure is used to evaluate the production cost for each machine setup (see App. A).

3 Solution Space

The Solution Space is a discrete space depending on the resolution of the load cells. In order to solve the optimization problem, we decided to approximate this space in two different ways: firstly with a grid characterized by a regular *step* and secondly with a continuous space to cope with the characteristics of some optimization algorithms.

Let $\mathbb{S}$ be the Solution Space and let us approximate it with an ordered grid of H-dimensional space where H is the number of hoppers. The number of solutions, that is the cardinality $N_{\mathbb{S}}$ of $\mathbb{S}$, depends on the grid dimension. If we suppose that each hopper weight may assume a discrete value in the interval $I = [a, b]$ and let L be the number of discretization levels depending on the *step* used to discretize the interval I, then

$$N_{\mathbb{S}(full)} = H^L \tag{9}$$

It is clear that as the hopper number or the level number increases, the number of solutions increases very rapidly.

If we approximate $\mathbb{S}$ with a continuous space, the number of solution can be expressed by the volume of this space. Let us suppose that each hopper has a maximum capacity M, then the volume of $\mathbb{S}$ is defined by:

$$Vol_{(full)} = \int_0^M \int_0^M \cdots \int_0^M \mathrm{d}w_1 \mathrm{d}w_2 \ldots \mathrm{d}w_H = M^H \tag{10}$$

The volume of $\mathbb{S}$ gets bigger and bigger increasing the number of hoppers or using hoppers with an increased maximum capacity.

Luckily, an interesting symmetry property can reduce the cardinality/the volume of the Solution Space. As a matter of fact, the MWM does not give preference to any hopper over the others, and each hopper can be set in exactly the same way as the others. In other words, a solution $\boldsymbol{\mu} = (\mu_1, \mu_2, ..., \mu_H)$ is invariant to any permutation $\sigma(\cdot)$ of the hopper indices $(\mu_{\sigma(1)}, \mu_{\sigma(2)}, ..., \mu_{\sigma(H)})$ in terms of any objective function. This characteristic is useful since the Solution Space can be reduced considering only the vectors $\boldsymbol{\mu}$ such that:

$$\mu_1 \geq \mu_2 \geq ... \geq \mu_H \tag{11}$$

Hence, reducing the grid Solution Space according to (11), the number of solutions is now:

$$N_{\mathbb{S}(red)} = \frac{(L + H - 1)!}{H!(L - 1)!} \tag{12}$$

The comparison between the two Solution Space cardinality for $H = 8$ by increasing the number of levels L is presented in Figure 2. It shows the drastically reduction of the space cardinality thanks to the use of (11).

Also approximating $\mathbb{S}$ with a continuous space, the symmetry property allows for a reduced Solution Space volume as follows:

$$Vol_{(red)} = \int_0^M \int_0^{w_1} \cdots \int_0^{w_{H-1}} \mathrm{d}w_1 \mathrm{d}w_2 \ldots \mathrm{d}w_H = \frac{M^H}{H!} \tag{13}$$

In particular, $Vol_{(red)}$ is always smaller than $Vol_{(full)}$ and the difference between the two volumes $(Vol_{(full)} - Vol_{(red)})$ gets bigger as H increases.

4 Optimization procedures

As described in Section 2 the optimization problem depends on the unknown probability distribution W. To estimate the W distribution and its main moments a simulator of the MWM has been developed (see App. A). The simulator estimates the W distribution for each $(\boldsymbol{\mu}, T, \alpha)$.

In order to find the optimal solution $\boldsymbol{\mu}^{opt}$ of the problem (6), a suitable optimization technique is required. In our case the standard optimization techniques requiring the knowledge of the function or the possibility to numerically compute it cannot be used (i.e. [9] for standard methods). We would rather need some technique able to cope with

the stochastic nature of the problem and with the characteristics of the Solution Space (continuous or discrete). The proposed general optimization scheme is in Figure 3. Given T, α and the desired number of simulations ($noSims$), at iteration k, for a certain $\boldsymbol{\mu}_k$ the simulator gives the W_k distribution and, as a consequence, the objective function value. Then the optimizer decides either to stop the routine thus obtaining the optimal $\boldsymbol{\mu}^{opt}$ or to go to the next step $(k+1)$. Details about the implemented algorithms in the optimizer are in Sections 4.1 and 4.2.

4.1 Continuous Optimization

Gradient-based methods. In the continuous $\mathbb{S}$ domain, two gradient-based methodologies are selected to tackle the stochastic optimization problem: Simulation Perturbation Stochastic Approximation (SPSA) [10, 11, 12] and Response Surface Methodology (RSM)[13, 14, 15, 16] (see App. B). SPSA is proposed due to its high potential efficiency given that it requires only two measurements of the objective function (i.e., two Monte Carlo simulations) for estimating the gradient in each iteration . Instead, RSM is proposed because its estimate of the gradient is more accurate.

In the simulation-optimization routine (Figure 3), at each kth iteration of the algorithm the optimizer varies the mean weights μ_j trying to minimize the Cost value estimated by the simulation module. The two modules work iteratively until a predefined stopping condition is met. Let η_1 and η_2 be some small positive numbers, k_{max} be the maximum number of iterations in each optimizer search, and $L(\boldsymbol{\mu}_k)$ be the objective function value evaluated using the setup $\boldsymbol{\mu}_k$. Each optimizer search is stopped when one of the following three criteria is satisfied:

1. $k \geq k_{max}$
2. $\mid \boldsymbol{\mu}_k - \boldsymbol{\mu}_{k+1} \mid \leq \eta_1$
3. $\frac{|L(\boldsymbol{\mu}_k) - L(\boldsymbol{\mu}_{k+1})|}{L(\boldsymbol{\mu}_k)} \leq \eta_2$

Since the Cost function is plainly not convex in $\boldsymbol{\mu}$, the simulation-optimization procedure is performed from a set of m initial points $\boldsymbol{\mu}_l$, $l = 1, ..., m$. In particular, to enhance the routine performance, each initial point needs to be different from the others. For this purpose, a sequence of initial points is generated satisfying the symmetry property of $\mathbb{S}$ according to (11). To ensure that an initial point implies a high probability of packing a conforming package, the following constraint is used [7]:

$$Prob\,(W \leq T) \leq \epsilon$$

that is roughly approximated by:

$$\Phi\left(\frac{T - \sum_{j=1}^{H} \mu_j}{\alpha\sqrt{\sum_{j=i}^{H} \mu_j^2}}\right) \leq \epsilon \tag{14}$$

Empirically, the number of initial points m is picked according to the following relation:

$$n = (\overline{k} \times n_k) \times m \tag{15}$$

where $\overline{k}$ is the average number of iterations of the algorithm per each search and it is estimated thanks to a previous empirical analysis, n_k is the number of objective function evaluation at each kth iteration, n is the overall number of objective function evaluations (see Section 5). The simulation-optimization routine returns the solution among the m searches which has the lowest Cost value.

Continuous Random Sampling Algorithm (RC). The RC is a "foolproof" algorithm that randomly samples n independent points, where each point is a machine setup $\boldsymbol{\mu}$, from the continuous $\mathbb{S}$ domain subject to the constraints (11) and (14). The n points are evaluated by the machine simulator and the algorithm stores the solution with the minimum cost value. This heuristic algorithm is introduced to compare its performance with the performances of structured algorithms.

4.2 Discrete Optimization

Discrete Random Sampling Algorithm (RD). The RD algorithm randomly samples n independent points from the discrete $\mathbb{S}$ domain. Then, each machine setup is evaluated by the simulator and the best solution found is taken. Each point has to fulfill the constraints (11) and (14). This is a very easy and intuitive algorithm as the RC algorithm. It is introduced to understand if the domain nature can affect the performance of the algorithm.

"Brute Force" algorithm. In the discrete $\mathbb{S}$ domain, the Brute Force algorithm (BF) requires the construction of an H-dimensional ordered grid characterized by a predefined regular *step*. Each point (i.e. a machine setup) that does not satisfy the constraints (11) and (14) is deleted from the grid.

Let M be the maximum capacity of each hopper, let H be the number of hoppers, then the maximum capacity of the machine is expressed by the product $H \cdot M$. The ratio between the actual capacity of a machine ($\sum_{j=1}^{H} \mu_j$) and the maximum capacity is called *K_Ratio*.

$$K\_Ratio = \frac{\sum_{j=1}^{H} \mu_j}{H \cdot M} \tag{16}$$

Empirically, it was observed that the machine setups characterized by small values of objective function lie within a narrow *K_Ratio* range. This property is added to further reduce the grid dimension.

Since we want to compare the algorithm using the same number of objective function evaluations, n points are sampled from the grid. Then, the n machine setups are evaluated by the simulator and the one with the minimum value of objective function is picked up as the best one.

5 Results

We consider a single-layer MWM with 8 hoppers. The product to be packed is a kind of dried pasta, it is always reworkable and from previous empirical analysis the constant α is equal to 0.123 [5]. The target weight T is fixed at 500. The specific parameters used in (6) are reported in Table 1 (see [5] for more details).

We compare the performance of gradient-based algorithms (SPSA and RSM), the Brute Force algorithm (BF) and the random sampling algorithms (RC and RD) using the same number of objective function evaluations (15), that is selected as a proxy of the computational effort.

To determine the value of n, the algorithm SPSA is picked as "base algorithm". According to its tuning (see App. C), the optimal number of initial points is $m = 500$ and the number of iterations $\overline{k}$ before ending are 17.89. Moreover SPSA requires only two measurements of the objective function (i.e. $n_k = 2$) to estimate the gradient in each iteration, then n turns out to be equal to 17890. This implies that the BF, RD and RC algorithms have to sample 17890 points. Instead RSM needs a certain number of initial points. From previous simulations, we estimated that RSM with $H = 8$ performs an average of 1280 experiments ($\overline{k} \times n_k$) before ending (see App. B). Given $n = 17890$ and according to (15), the number of initial points is equal to 14.

As indicators of performances we selected the median and the standard deviation of the expected production cost, that are widely used to measure the central tendency and the variability. These indicators are estimated over 50 simulated replications. In each replication, each algorithm stores the best solution and the results are shown in Figure 4. We can notice that the nature of the domain (continuous or discrete) does not affect the algorithm performances. The BF has a smaller value of median than the RSM one, that instead is characterized by a smaller value of standard deviation. The RSM and the RD seems to have the same performances. The same remark can be done for the RD and RC algorithms. The SPSA has a standard deviation not different from that of the BF and RC algorithms, but its median value is the worst.

Then we decrease the value of n to observe its impact on the algorithm performance. In particular, n is set equal to 7090 (that corresponds to $m = 200$ for SPSA and $m = 5$ for RSM) and 2583 (that corresponds to $m = 75$ for SPSA and $m = 2$ for RSM). Each algorithm is replicated 50 times to compare the relative performance and the best $\mathbb{E}[C]$ value is stored at each replicate.

Figure 5 presents the results of the analysis in terms of median and standard deviation of the expected cost of each algorithm for different n values. As easily predictable, the increasing of n causes the values of both indicators to diminish. In fact, a greater number of evaluated machine setups allows for an improvement of the algorithm performance.

We can surmise that the performance of BF and RSM algorithms become more and more comparable as n decreases. The performance difference between the two random sampling algorithms (RD and RC) remains irrelevant by varying n and their performance are always worse than the BF and RSM ones. Instead, SPSA has always the worst perfor-

mance regardless the number of objective function evaluations. According to this results, for the next analysis we do not consider SPSA anymore.

In order to generalize these results, the MWM parameters are changed and the performance of the methods in terms of cost value are presented in Figure 6. In particular, the parameters are changed according to $H = [5, 8]$, $\alpha = [0.1, 0.3]$ and $T = [250g, 500g]$.

Consistent with [7], Figure 6 shows that a greater number of hoppers causes the cost value of a package to decrease. This drop becomes more significant as the target weight increases. To increase the target weight means to pack a greater amount of product, consequently a package of $500g$ costs more than a package of $250g$, as viewable in the graph. Moreover, the graph, along with a proper statistical analysis, confirms that the performance of the BF and RSM algorithms are equivalent and always better than the performance of the random algorithms.

In summary, the nature of the Solution Space does not affect the performance of the algorithms. The BF and RSM have similar performance, but RSM is a more structured algorithm and consequently it is more time and space efficient. For these reasons, we suggest RSM as optimizer.

5.1 Case study

The idea is to compare the optimal solution obtained from the RSM algorithm with a "rule of thumb" currently used in the industrial practice. Easily this heuristic rule sets all the hoppers with the same mean weight, that is equal to the target weight divided by the number of hoppers (i.e. $\mu_j = T/H$). Thus, we expect that all the hoppers are selected in the combination to open at each machine cycle.

Let us consider a MWM with 8 hopper and the target value T fixed at 500, the heuristic rule suggests as optimal setup $\boldsymbol{\mu}^{opt} = (62.5, 62.5, 62.5, 62.5, 62.5, 62.5, 62.5, 62.5)$. Instead, the RSM algorithm, according to the parameters presented in Table 1, gets $\boldsymbol{\mu}^{opt} = (221.2, 219.2, 193.1, 68.6, 33.1, 26.8, 13.8, 11.0)$ as the optimal setup. Then, 3000 replicates of the simulations ($noSims$) at the optimal setups are run to estimate the expected value of the production cost per "conforming" package and the standard deviation of this cost. The results are reported in Table 2.

According to Table 2, the RSM optimal solution assures an expected cost and a standard deviation smaller than the heuristic rule. In order to better understand the results, let us suppose that the MWM produces 60 packages per minute, and suppose that the firm works two 8-hour shifts, 300 days per year. The solution $T/8$ involves a cost equal to 3519936 €/(year-machine), instead the RSM setup involves a cost equal to 2745792 €/(year-machine). Thus the use of the RSM allows the firm to save 774144 €/(year-machine), a saving of about 22%, and this saving rises as the number of packages per minute increases.

6 Conclusions and further research

In this paper, the setup problem of a MWM has been studied. The setup problem requires the determination of the optimal flow rate of each radial feeder. We decided to tackle this problem by finding the optimal mean product quantity to deliver into the pool hoppers at each cycle in order to minimize the expected production cost value per "conforming" package in a fixed time period. The setup affects the machine performance and has to be done each time the characteristics of the packaging (product, target weight, etc.) change. In industrial practice, operators currently use their skill and experience to setup the machine and the scientific literature is inadequate. The wide distribution and availability of MWM worldwide and the impact that an improper machine setup can have in the machine efficiency and speed underline the importance of the problem.

Thanks to the definition of the setup problem and its main variables together with the expression of the objective function to minimize, the problem to find out the optimal setup of a MWM has been formalized. The Solution Space of the problem has been characterized. Its deep analysis allows us to discover an interesting symmetry property to reduce its dimension and, consequently, to tackle the setup problem faster. According to the characterization of the Solution Space, five algorithms have been considered: gradient based algorithms (SPSA and RSM), a "Brute Force" (BF) algorithm and two random sampling algorithms (RC and RD). Their performance in terms of median and standard deviation of the expected cost have been compared using the same number of objective function evaluations, which is used as a proxy of the computational effort. The nature of the Solution Space does not affect the algorithm performance. The BF and RSM performance appear to be very similar, the same conclusion can be drawn for the two random sampling algorithms but their performance are always worse than the BF and RSM ones. These conclusions have been generalized changing the MWM main parameters. Instead, the SPSA has always the worst performance. The RSM is more efficient in terms of space and time than the BF, thus we suggest to use RSM to setup the MWM. Lastly, the optimal solution found with the RSM algorithm is compared with an heuristic rule currently used in industrial practice. The expected cost of the RSM solution and its standard deviation are lower than the industrial solution allowing a firm to save money as much as the number of packages per minute increases. A structured approach to setup the machine is the best way to increase the performance of a firm.

However we think that further research is necessary to generalize the results found. Possible topics are to increase the complexity of the machine (machine with multiple products and multiple hopper layers, vibrating cone model, etc.); to add a time constraint due to the cycle time, that implies the use of a smarter algorithm to tackle the knapsack problem; to use other optimization methods to solve the setup problem. In regard to the simulation of the item flow in the vibrating channels, when the size of a single item is large and few items are needed to form a package the approximation of the random variable weight with a normal distribution may no longer be valid, and furthermore approximation in (1) may not be justified.

References

[1] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Inc., New York, NY, USA, 1990.

[2] Y. Karuno, H. Nagamochi, X. Wang, Bi-criteria food packing by dynamic programming, Journal of the Operations Research Society of Japan 50 (4) (2007) 376–389.

[3] S. Imahori, Y. Karuno, H. Nagamochi, X. Wang, Kansei engineering, humans and computers: Efficient dynamic programming algorithms for combinatorial food packing problems, International Journal of Biometrics 3 (3) (2011) 228–245.

[4] K. Sakaeda, T. Sashiki, Combinatorial weighing method and apparatus therefor, Google Patents, uS Patent 4,609,058 (September 1986).

[5] A. Beretta, Q. Semeraro, On a RSM approach to the multihead weigher configuration, in: Proceeding of ASME 2012 11th Biennial Conference On Engineering Systems Design And Analysis, 2012.

[6] A. Silvestri, D. Falcone, C. Cerbaso, A. Forcina, A weighing algorithm for checking missing components in a pharmaceutical line, International Journal of Engineering Business Management 6.

[7] E. del Castillo, A. Beretta, Q. Semeraro, Analysis and optimal setup of a multihead weigher machine, Submitted.

[8] K. Kameoka, M. Nakatani, N. Inui, Phenomena in probability and statistics found in a combinatorial weigher (in japanese), Transactions of the Society of Instrument and Control Engineers 36 (2000) 388–394.

[9] A. Ravindran, G. V. Reklaitis, K. M. Ragsdell, Engineering optimization: methods and applications, John Wiley & Sons, 2006.

[10] J. Spall, A stochastic approximation technique for generating maximum likelihood parameter estimates, in: Proceedings of American Control Conference, 1987, pp. 1161–1167.

[11] J. Spall, Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, IEEE Transaction on Automatic Control 37 (1992) 332–341.

[12] J. Spall, Introduction to stochastic search and optimization: Estimation, simulation, and control, John Wiley & Sons, 2003.

[13] G. E. P. Box, K. B. Wilson, On the experimental attainment of optimum conditions, Journal of Royal Statistical Society B13 (1951) 1–38.

[14] H. G. Neddermeijer, G. J. van Oortmarssen, N. Piersma, R. Dekker, A framework for response surface methodology for simulation optimization, in: Proceedings of the 32nd conference on Winter simulation, 2000, pp. 129–136.

[15] E. Del Castillo, Process optimization: a statistical approach, Vol. 105, Springer, 2007.

[16] R. H. Myers, D. C. Montgomery, C. M. Anderson-Cook, Response surface methodology: process and product optimization using designed experiments, Vol. 705, John Wiley & Sons, 2009.

Appendix A. The simulator.

In this section, the simulator that mimics the behavior of a MWM is presented. As depicted in Figure 7, given a minimum target weight T, the $\alpha = \sigma_j/\mu_j$ parameter, a desired maximum number of simulations ($noSims$) and a vector of hopper mean weights $\boldsymbol{\mu}$, the simulator solves a specific knapsack problem and estimates via Monte Carlo methods the distribution of the package weight W. Based on this information, the objective function is evaluated.

In this paper, we consider a MWM with 8 hoppers, the target weight T is fixed at 500, α is set equal to 0.123 and $noSims$ is fixed at 3000.

In literature, different Knapsack Problem formulations have been proposed: a linear objective function [2, 3] or an absolute value objective function [4] both with linear constraints on the package weight, or an absolute objective function with some rules to choose the combination of hoppers to open [5]. This paper assumes that the machine has a built-in algorithm that solves for each package the deterministic knapsack-problem that follows:

$$\text{objective:} \qquad \text{minimize } \sum_{j=1}^{H} w_j x_j \tag{17}$$

$$\text{subject to:} \qquad \sum_{j=1}^{H} w_j x_j \geq T \tag{18}$$

$$x_j \in \{0,1\}\ j = 1, 2, ..., H \tag{19}$$

where

$$x_j = \begin{cases} 1, & \text{if the hopper } j \text{ is chosen;} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

and (19) expresses the binary constraint of this variable. The objective (17), together with (18), aims at attaining the total weight of a package as close to the target weight T as possible.

Because the knapsack algorithm implemented in real machines has proprietary details which are not available, we tackle this problem with an enumerative algorithm.

Appendix B. Response Surface Methodology.

Usually, classic Response Surface Methodology (RSM) consists in two phases. In the first phase the objective function is locally approximated by a first-order polynomial.

$$y = \beta_0 + \sum_{j=1}^{H} \beta_j x_j + \epsilon \tag{21}$$

where y is the response variable, $(x_1..x_H)$ are the coded input variable, β_0 and β_j are respectively the constant and the linear coefficient that are estimated from the data.

When the first order model is adequate a line search algorithm is applied to find a new region of interest. The line search is ended when there is no further improvement in the response along the line. At that point, a new experiment is performed and a first-order model is fitted and a new line search phase is started. The procedure is repeated until the first-order polynomial model is not adequate (i.e. if a significant lack-of-fit is present). At that time the RSM moves to the second phase and additional experiments are conducted to obtain a more precise estimate of the response function. In this phase the objective function is approximated by a second-order polynomial (22) and a canonical analysis is used to find the optimum point, if it exists.

$$y = \beta_0 + \sum_{j=1}^{H} \beta_j x_j + \sum_{j=1}^{H} \beta_{jj} x_j^2 + \sum_{j=1}^{H} \sum_{i>j} \beta_{ji} x_j x_i + \epsilon \tag{22}$$

Although there are many designs to choose from, a fractional two-level factorial design of resolution-V augmented by three center points is used because of the large number of parameters ($H = 8$). This design allows for a reduction of the experiment number (from 128 experiments of the full factorial design to 64 experiments) without losing any benefits. In fact, this design is orthogonal, gives unbiased estimators of the regression coefficients of a first-order model and it was easily augmented to derive a second-order design (a CCD - Central Composite Design) in the second phase.

According to [16, 15], the steepest descent direction is given by $-\nabla\widehat{y}$. To define the step size, the most important factor is chosen by determining i such that $i = \arg\max_{j=1,...,H} |b_j|$. Then, the step size in the coded variables is set equal to $\Delta_j = \frac{-b_j}{|b_i|},\ j = 1, ..., H$. In the original units, the step size corresponds to $\Delta_j \times R_j/2$, where $R_j/2 = \mu_j \times \lambda$ is the half of the range of values over which we will vary factor j and λ is fixed equal to 0.2 according to a previous experimental campaign. Regarding the stopping rule, despite the possible drawbacks, the 2-in-a-row rule is picked for its simplicity of implementation.

To define the number of initial points m (each one is the central point of the first fractional factorial design), according to (15) we need the average number of iterations of the algorithm and the number of objective function evaluation at each algorithm iteration. Unfortunately, it is impossible to define exactly the value of n_k because at each step of the algorithm n_k is not a constant differently from the SPSA. As a matter of fact, in first phase 64 experiments are performed to estimate the improvement direction; along the line search

2 experiments at each step are done to test the stopping rule; in the second phase only the experiments on the axial points are performed. Accordingly, a deep experimental campaign was performed and from it, we estimated that RSM with $H = 8$ performs an average of 1280 experiments ($\overline{k} \times n_k$) before ending. Then, if the overall number of objective function evaluations n is defined, the number of initial points is picked according to (15).

Appendix C. Number of initial points of SPSA.

The goal is to define the optimal number of initial points for the Simulation-Optimization routine using SPSA as optimizer. Table 3 presents the routine parameters that have been used to perform the analysis. Moreover, twelve values of m are considered [25, 50, 75, 100, 150, 200, 300, 400, 500, 600, 700, 800] and each condition is replicated 50 times. Each search is stopped when one of the stopping criteria (Section 4.1) is met and the minimum cost value is stored.

Figure 8 presents the individual value plot of the results. The graph suggests that as the number of initial points increases, as the routine is able to reach solutions characterized by a lower Cost value. The analysis of the data shows that a number of initial points equal to 500 is not statistically different from 800 and the dispersion around the mean is not too high. Thus, 500 initial points are the optimal setup for the simulation-optimization routine when SPSA is used as optimizer.

Summing up, let m be equal to 500 and estimating that SPSA has about 17.89 iterations ($\overline{k}$) before ending, it has been estimated that the number of objective function evaluations is 17890 per each replicate (at each iteration, SPSA performs two evaluations of the objective functions n_k).

Figures and Tables.

H	8	T	500 (g)	α	0.123
C_0	2534 (€cent)	C_U	0.03 (€cent/g)	C_S	3.2 (€cent)

Table 1: The parameters of the optimization problem.

Method	$\boldsymbol{\mu}^{opt}$	$\mathbb{E}[C]$	$sd[C]$
$T/8$	$(62.5, 62.5, 62.5, 62.5, 62.5, 62.5, 62.5, 62.5)$	20.37	0.1557
RSM	$(221.2, 219.2, 193.1, 68.6, 33.1, 26.8, 13.8, 11.0)$	15.89	0.0010

Table 2: Case study.

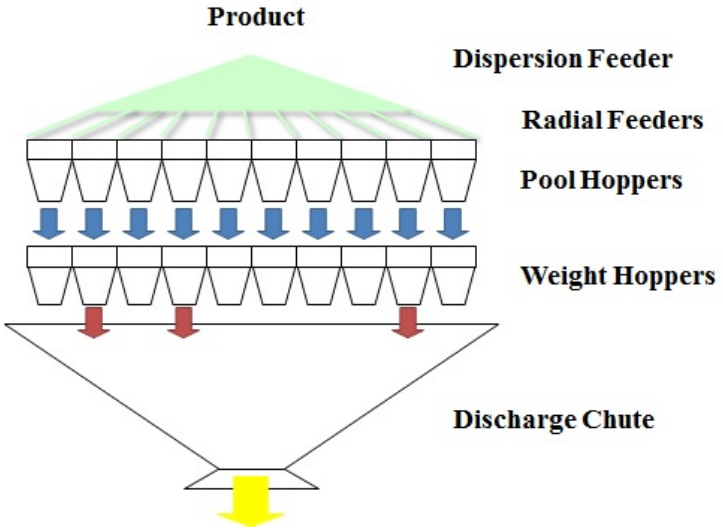


Figure 1: A single-layered multihead weigher machine.

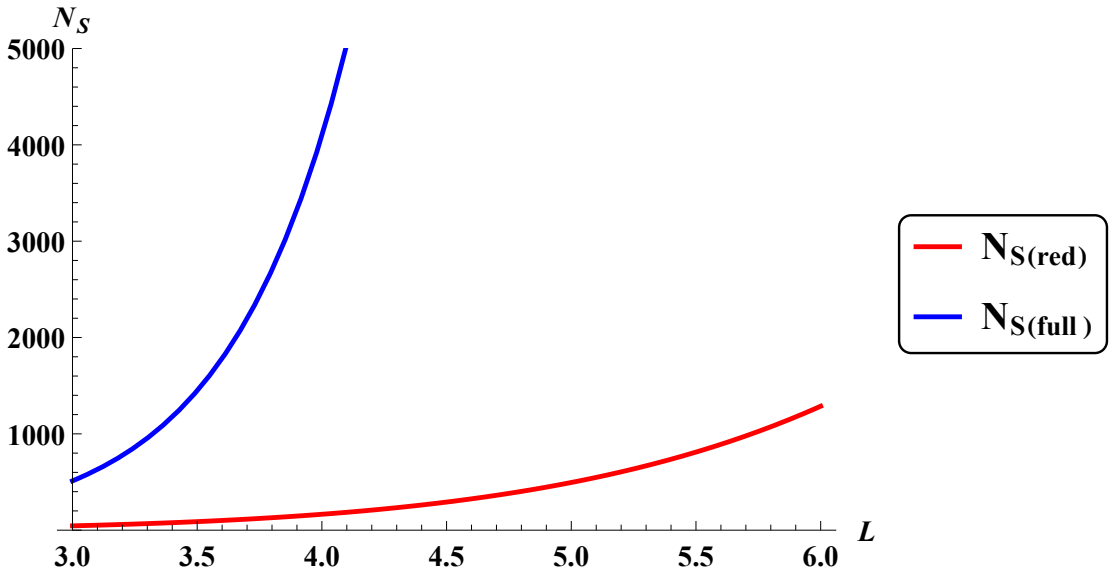


Figure 2: The cardinality plot: $N_{\mathbb{S}(full)}$ versus $N_{\mathbb{S}(red)}$ with $H = 8$

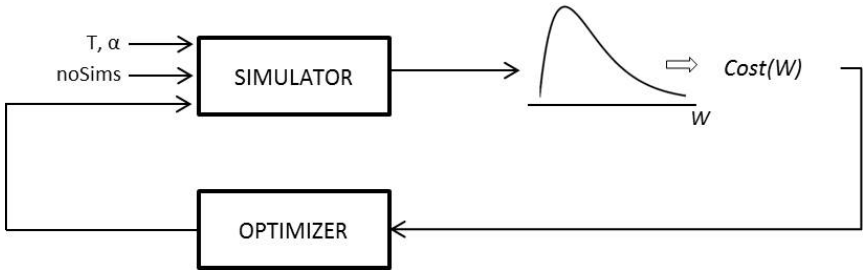


Figure 3: Simulation Optimization Routine

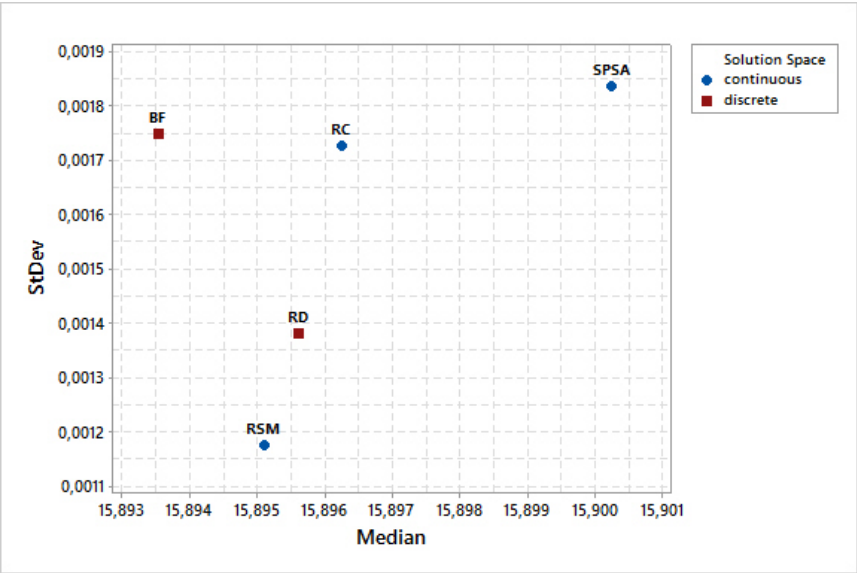


Figure 4: Scatterplot of the performance of the different optimization methods considering $n = 17890$. The median of the $\mathbb{E}[C]$, obtained thanks to the 50 replicates, is plotted on the x-axis. Instead, on the y-axis, the value of the standard deviation of $\mathbb{E}[C]$ is plotted for each optimization method.

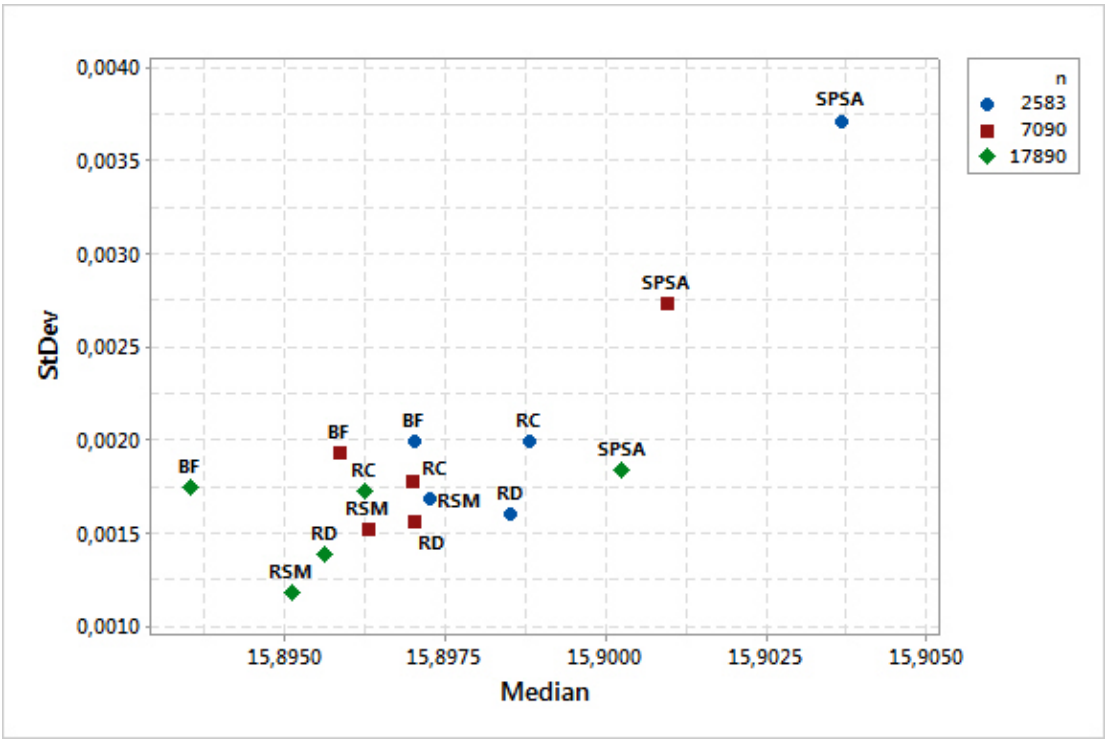


Figure 5: Scatterplot of the performance of the different optimization methods. The median of the $\mathbb{E}[C]$, obtained thanks to the 50 replicates, is plotted on the x-axis. Instead, on the y-axis, the value of the standard deviation of $\mathbb{E}[C]$ is plotted for each optimization method.

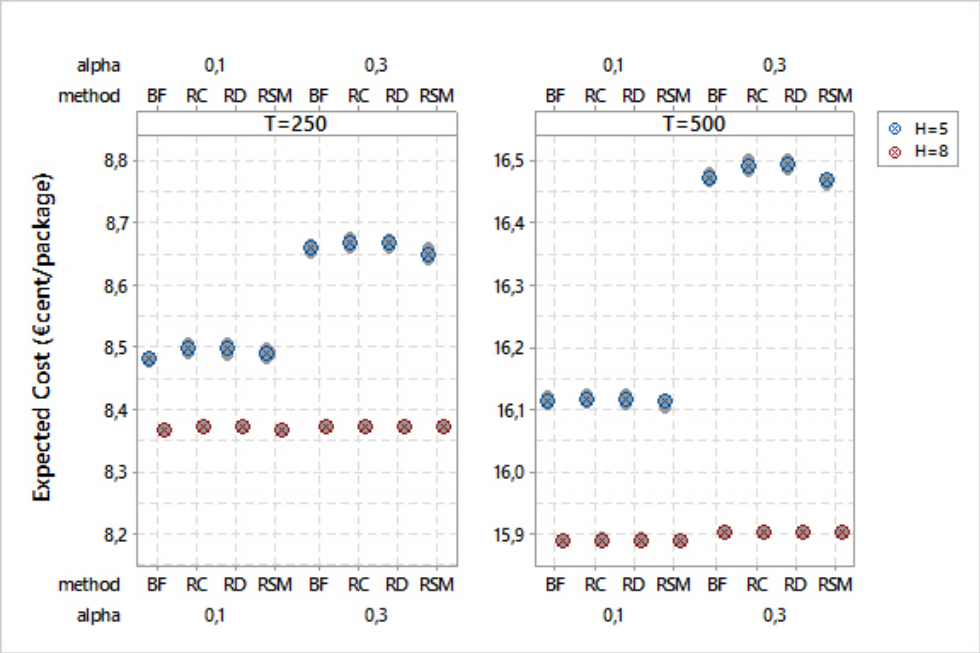


Figure 6: The performance of each optimization method for different MWM parameters.

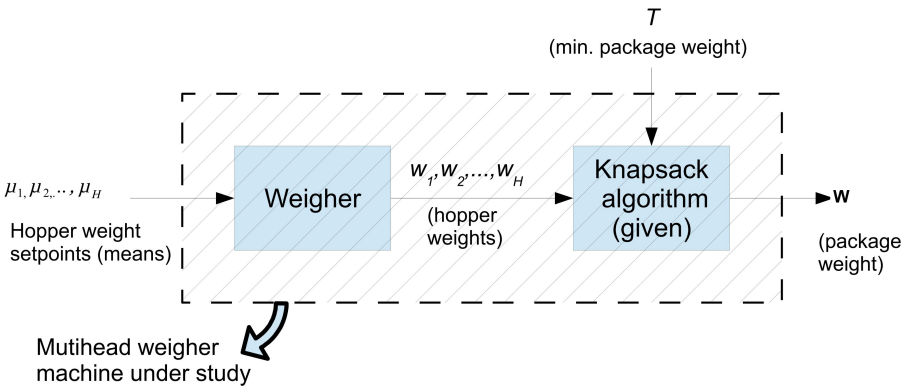


Figure 7: How the machine simulator works [7].

H	8	T	500 (g)	α	0.123
C_0	2534 (€cent)	C_U	0.03 (€cent/g)	C_S	3.2 (€cent)
$noSims$	3000	α_{Spall}	0.602	γ	0.101
k_{max}	100	η_1	1	η_2	0.001

Table 3: The routine parameters used to tune the number of initial points.

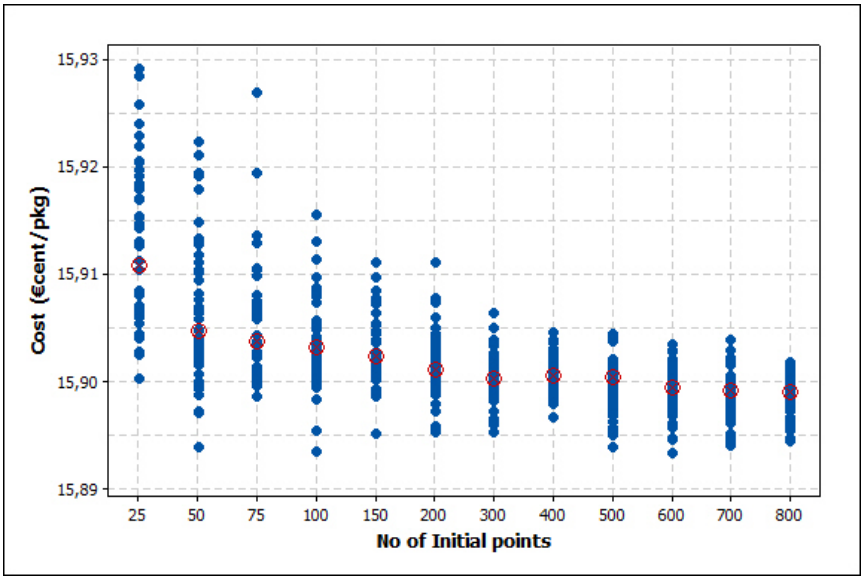


Figure 8: The individual value plot of the expected cost for different number of initial points. The red symbols are the medians.