POLITECNICO DI MILANO

Scuola di Ingegneria dei Sistemi

Corso di Laurea Magistrale in
Ingegneria Gestionale

Heuristic algorithms on a Multihead Weigher Machine setup problem

Relatore: Ch.mo Prof. Quirico Semeraro

Correlatore: Ing. Alessia Beretta

Tesi di Laurea Magistrale di:

Gianmarco Meroni Matr. 786707

Anno Accademico 2013 – 2014

"In the middle of difficulty lies opportunity."

Albert Einstein

Contents

List of Figures

List of Tables

Abstract

This thesis work summarizes the results of a research project involving a computer controlled machine called Multihead Weigher. This machine has a wide range of applications in the food industry for dried foods such as pasta, pet foods, coffee, cereals, or fresh foods such as salad, spare ribs, etc. Since the speed and the accuracy of the packaging line mainly depend on its performances, the machine needs an optimal setup strategy and a proper operation software. The operation software, working in real time, selects the best hopper subset to be opened according to the product, the time constraints and the objective function. This industrial problem can be modeled as a knapsack problem. Instead, the setup problem deals with the definition of the average quantity of product (setpoints) delivered by the radial feeders to each hopper. Thus, the focus of this research is on the jointly determination of the optimal setpoints for a specific machine. Those setpoints define the output of the multihead weigher, which is the average weight discharged by the machine. In order to evaluate the output, thus the performances, an economic objective function has been proposed. To assess the system performances we choose to rely on a simulator that mimics the behaviour of a simple Multihead Weigher machine since the analytical way was not viable.

In this project three different methods are proposed to find the setpoints that minimize the objective function. The discretized case is fulfilled with a Brute-force search in combination with some heuristic rules that can reduce the number of solutions to be evaluated. On the other hand, to tackle the continuous case, the Response Surface Methodology and the Particle Swarm Optimization are implemented. The three methods have been evaluated with two different operation knapsack problems and with an *ad hoc* developed economic iterative knapsack.

Having defined all the optimizer and the operation policies for the simulator, the research proposes a full comparison of all methods in order to pinpoint the best approach. Eventually, we propose a comparison between

the currently used approach and our best approach.

Key words: Multihead Weigher; Combinatorial weighing machine; Simulation; Optimization; PSO; RSM; Heuristic algorithms;

Sommario

Questo lavoro di tesi riassume i risultati di un progetto di ricerca relativo ad una macchina controllata computericamente chiamata Pesatrice Multitesta. Questa macchina ha ampie possibilità di applicazioni soprattutto nelle industrie alimentari sia per il confezionamento di prodotti secchi come pasta, cibo per animali, caffè e cereali, sia per il confezionamento di prodotti freschi quali insalate, costine, ecc. Poiché la velocità e la precisione della linea produttiva dipendono principalmente dalle sue prestazioni, la macchina necessita si una strategia di setup ottimale e un adeguato software di gestione. Quest'ultimo, operando in tempo reale, seleziona il miglior sottoinsieme di cestelli da aprire in relazione al prodotto, ai vincoli di tempo e alla funzione obiettivo. Questo problema industriale può essere modellato con il cosiddetto "Knapsak problem". Invece, il problema di setup riguarda la definizione della quantità media di prodotto (*setpoint*) trasportata dai canali radiali a ciascun cestello. Quindi, l'obiettivo di questo lavoro è la determinazione del *setpoint* ottimale per una specifica macchina. Questi *setpoint* influenzano l'output della Pesatrice Multitesta, ovvero il valor medio di prodotto contenuto all'interno di un pacchetto. Al fine di valutare l'output, e di conseguenza le prestazioni della macchina, è stata proposta una funzione obiettivo di tipo economica. Per valutare le prestazioni della macchina abbiamo scelto di basarci su un simulatore che replichi il comportamento di una pesatrice Multitesta poiché la via analitica non risulta attualmente percorribile.

In questo progetto sono stati proposti tre differenti metodi per trovare i *setpoint* che minimizzano la funzione obiettivo. Il caso discreto è trattato seguendo un approccio *Brute-force* in combinazione con alcune regole euristiche atte alla riduzione del numero di soluzioni da valutare. Il caso continuo, invece, è stato analizzato mediante sia la tecnica *Response Surface Methodology* che la *Particle Swarm Optimization*. Questi tre metodi sono stati valutati con due differenti problemi di gestione (*Knapsak Problems*) e con un Knapsak iterativo sviluppato appositamente.

Avendo definito tutti gli ottimizzatori e tutte le politiche di gestione per il simulatore, la ricerca propone un confronto tra tutti i metodi per delineare il miglior approccio. In conclusione, proponiamo un confronto tra in metodo correntemente usato e il metodo da noi proposto.

Parole chiave: Pesatrice Multitesta; Simulazione; Ottimizzazione; PSO; RSM; Algoritmi euristici;

Introduction

The stakeholders of the packaging macrosystem are (Fig. 1): the producers of packaging machines, the packagers (those that use packaging machines), the suppliers of packaged products and the final consumer. Since, the rapid development of the packaging industry is causing the producers of packaging machines to propose new versatile, productive and highly automated solutions. Among the solutions oriented to the packaging of discrete parts, the multihead weigher machine (hereafter MHW) is spreading due to its high speed, accuracy and reliability. In particular, a MHW machine (sometimes called combinatorial weighing machine) is a computer-controlled machine used to fill a package of products at its target weight.

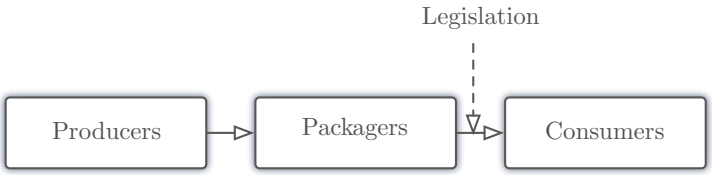


Figure 1: Stakeholders of the packaging macrosystem

Since its first appearance in the marketplace in the seventies, it has been continuously improved to meet the ever-changing market requirements and consumer tastes. Actually, this machine has a wide range of applications in the food industry for dried foods such as pasta, pet foods, coffee, cereals, or fresh foods such as salad, spare ribs, etc. Its applications cover also the non-food areas, for instance the paper clip packages are filled using this machine. Among the MHW manufacturers, the one with the world leading position has 31 000 MHW machine installed all over the world. Despite its worldwide spread and the high number of manufacturers, scientific studies

aimed at studying its characteristics in order to reach optimal performance are few and most of the literature on this topic is mainly commercial (e.g. patents, machine brochures).

The MHW machine is a computer-controlled machine composed by a set of pool hoppers, a set of weight hoppers and a discharge chute. Each weight hopper is equipped with a load cell that weighs the product and transmits the information to a central microprocessor. The central microprocessor, according to a suitable algorithm, selects a subset of hoppers whose total weight is equal or closest to the target value and opens the selected hoppers releasing the product through the discharge chute to the packaging machine.

Since the MHW machine owns both the weighing and the weigh-out task, the ammount of material stored in each package need to fit the legislation limits instruct by the [Council Directive 1976]. Furthermore, the package weight, in order to make the packaging process profitable, needs to be as close as possible to the legislation limit W_L or to the target weight W_T.

This thesis aims to exploit the logic structures of the machine in order to define a machine setup since nowadays this relies on the operator's skill and experience. Chapter 1 proposes an insight to: the MHW machine hardware, the operation problem and how to tackle it and the setup problem. In the Chapter 2, the problem definition is introduced therefore, how to measure and evaluate the machine performance as well as the MHW simulator and its features. Then, in Chapter 3 the heuristic techniques discovered and the optimizator will be discussed in order to solve the setup problem in a continuos solution space. In the end, Chapter 4 proposes the results of all methods compared to each other and to the actual industrial solution.

Chapter 1

The Multihead Weigher machine

A packaging system is composed by a series of integrated machines that take the raw product from the processing line and fill a package according to a predefined weight. The packing system operates at very high speed, 24 hours a day and the throughput is more than 100 packages per minute, according to the specific machines used.

The distribution shakers are vibratory conveyors that transport the raw product from the processing line to the scale feed shaker. These conveyors can be also horizontal motion shakers when the product to be moved is fragile and/or coated. The downstream scale feed shaker is a vibratory conveyor that feeds the multihead weigher machine with the right amount of product according to a proper flow rate. Thus, the product (via the infeed chute and the feed system) is continuously fed to the hoppers of the multihead weigher machine. This machine is particularly efficient to select an opportune hopper combination so that the Form-Fill-Seal (FFS) machine can fill a package at its desiderated weight. In particular, a FFS machine pulls film to construct plastic bag while simultaneously fills it with product, and then seals the filled bag. The success of this operation depends on the integration between the FFS machine and the upstream multihead weigher machine. The FFS is the "master" machine, it send a signal to the multihead weigher when it is ready to accept the product to be packed. On this purpose the multihead weigher should be slightly faster than the FFS machine in order to meet its demand [Yamato Corporation 2012].

This chapter focuses the attention on the multihead weigher machine

because the speed and the accuracy of the packaging line mainly depend on the performance of this machine. In the follow, a detailed description of the multihead weigher machine and its main problem are presented.

1.1 Machine Hardware

A multihead weigher machine is a computer-controlled machine composed basically by a set of pool hoppers, a set of weight hoppers and a discharge chute, as showed in Figure (1.1), used in production area in which structural vibration may be present. One of the machine requirement is to work with high level of accuracy and this vibration can affect its performance. Thus, the weigher is normally mounted on a support frame that is rigid with a natural frequency high enough to prevent resonances excited by minor vibrations, compromising the measurement accuracy.

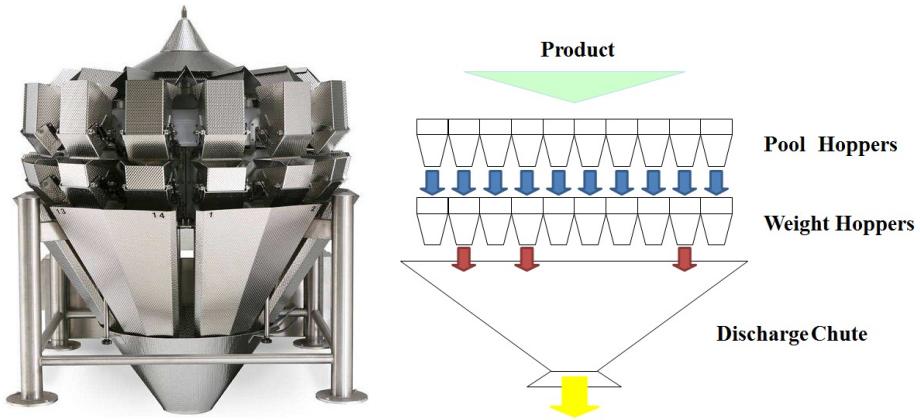


Figure 1.1: The multihead weigher machine

Infeed Chute (Figure (1.2)) is used to neatly guide the product from the scale feed shaker to the dispersion feeder.

Dispersion feeder is a circular feeder, cone shaped with a central apex, as shown in Figure (1.3). It is provided with controlled vibrators to ensure the movement of the product into the radial feeders.

Figure 1.2: Infeed chute.

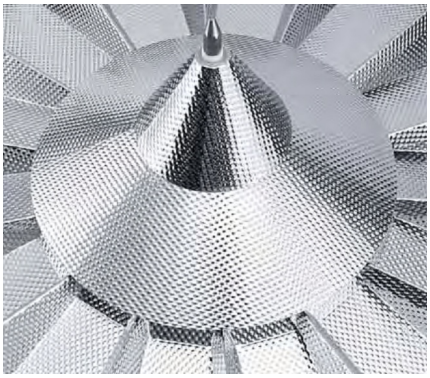


Figure 1.3: Dispersion feeder.

Weighing Station includes a radial feeder, a pool hopper, a weight hopper and a load cell (weight sensor), as depicted in Figure (1.4). The weighing stations are arranged around the dispersion feeder along radially extending lines to receive the product dispersed by the the dispersion feeder [Sakaeda & Sashiki 1986].

Radial feeders (Figure (1.5)) are vibrating channels used for feeding the below pool hoppers by means of vibration. Each radial feeder receives the product from the dispersion feeder and introduces it into the corresponding pool hopper.

Pool Hoppers are buffer stores used to stabilize the product received from the feed system, before dropping it into the weight hoppers. The hopper should be designed to prevent entrapment of the product, and the discharge mechanism should be fast and efficient - usually bomb doors or swivel gates. The construction material is usually mild or stainless steel depending upon the industry [The Weighing & Force Measurement Panel 2010]. Each pool hopper is provided with an independent radial feeder, so that when this hopper has discharged the product to the below empty weight hopper, the corresponding feeder feeds a suitable amount of product to this pool hopper.

Weight Hoppers are supported by suitable load cells that have to weight the product inside the hoppers. The construction of the weight hopper is such that it must robustly contain the product in order to improve the weighing

accuracy [The Weighing & Force Measurement Panel 2010]. This kind of hopper is typically of a welded steel construction.

Load cells (Weight Sensors) measure the weight of the product introduced into the weight hoppers and apply an electronic signal with the weight information to the combination control unit. A load cell is usually characterized by a maximum capacity within which the load cell operates coherently with its specifications.

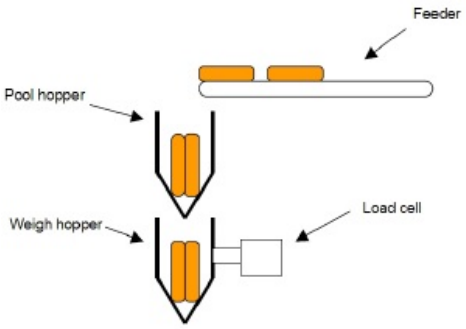


Figure 1.4: Weighing Station

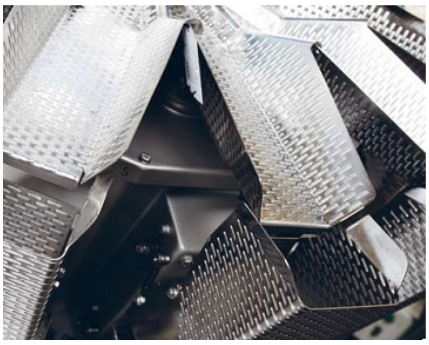


Figure 1.5: Radial feeders

Combination control unit computes the total weight of different hopper combinations and selects the combination having a total weight equal or nearest to the predefined weight. Then, the unit sends a signal to the selected weight hoppers to drop the product into the discharge chute. Moreover, it sends a signal to the the feeder system to feed the hoppers whenever they are empty. The unit, in order to improve its efficiency, previously needs some information, such as the vibration amplitude of each feeders, the discharge time of each hopper, the stabilization time of each load cell to read the weight minimizing the noise, etc.

Discharge chute is a funnel-shaped temporary hopper and its design can be very critical if high speed is required and/or difficult products are being weighted. It collects the product dropped by the weight hoppers and releases it into the packaging machine whenever the latter gets available.

1.1.1 How the machine works

The product is continuously fed by the scale feed shaker to the top of the multihead weigher machine where a dispersion feeder moves it out towards the radial feeders. Each feeder transfers the product into the pool hopper beneath by means of vibration. This set of hoppers performs the function of stabilizing the product before dropping it into the weight hoppers. Each weight hopper is equipped with a load cell that weighs the product and transmits the information to a central combination control unit. This unit calculates the total weight of many different hopper combinations and, according to a suitable algorithm, selects a subset of hoppers whose total weight is equal or closest to the desiderated weight value. The selected hoppers are opened to release the product, via the discharge chute, to the packaging machine (FFS) and, subsequently, the now-empty hoppers are refilled. The foregoing sequence of steps, outlined in Figure (1.6), is automatically repeated to produce a large number of packages one by one.

Since at each cycle only a subset of hoppers is selected, the product delivered could remain for more than one cycle in the weight hoppers without being delivered in the discharge chute. Depending on the type of machine and product (e.g. frozen food, fresh food), when a specific hopper remains closed for a certain number of cycles, the system can deliver its product into a trash box or can force it to belong to the next hopper combination to open.

1.1.2 Additional Features

In the market different kind of multihead weigher machines are offered, such as the double-layered multihead weigher machine [Karuno, Nagamochi & Wang 2009], [Karuno, Nagamochi & Wang 2010]. This machine is characterized by two different hopper layers, the weight hoppers and the booster hoppers. The presence of a second hopper layer increases the number of hopper combinations among which the microprocessor can select the optimal one.

Booster hoppers are just hoppers without weighing function but their weights are always known. As a matter of fact, each weight hopper sends its product and its weight information to the corresponding booster hopper,

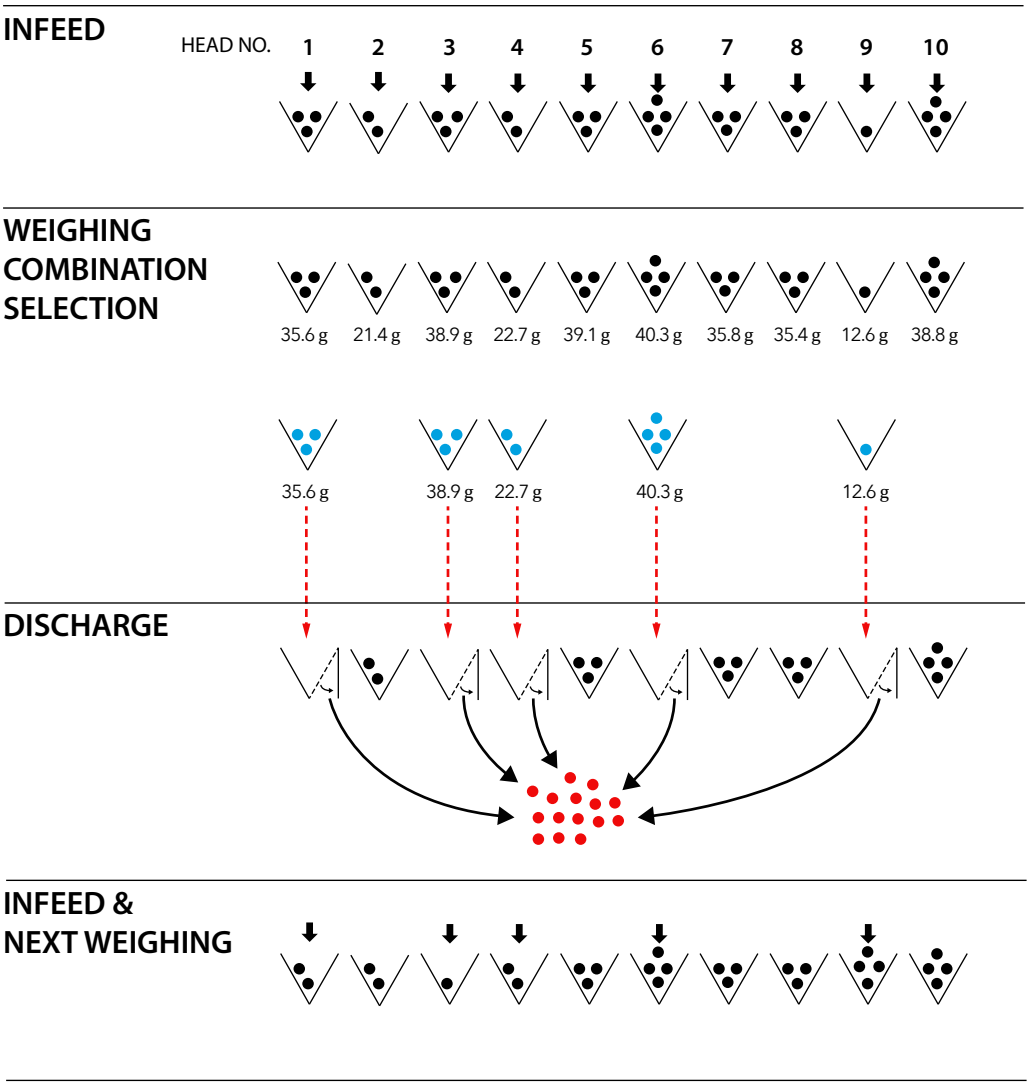


Figure 1.6: Steps sequence performed by the multihead weigher machine

whenever the latter becames empty. Then the weight hopper is refill with a new amount of product.

Two different types of double-layered machine are presented in literature: the Upright and the Diagonal, which are respectively depicted in Figure (1.7) and Figure (1.8). In the Upright machine, a weight hopper can be chosen only if the corresponding booster is chosen for a package. Differently, the Diagonal machine can not simultaneously open a weight hopper and the

corresponding booster.

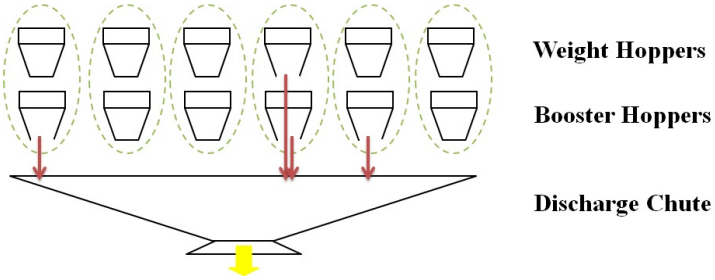


Figure 1.7: Upright multihead weigher machine

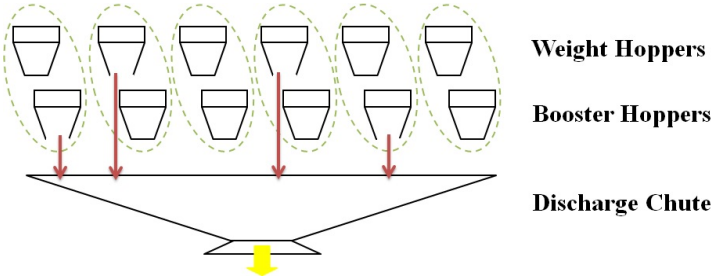


Figure 1.8: Diagonal multihead weigher machine

Instead, a duplex multihead weigher machine [Imahori, Karuno & Yoshimoto 2010],[Imahori, Karuno, Nishizaki & Yoshimoto 2012] is composed by a set of weight hoppers, as depicted in Figure (1.9), among which two disjoint subsets are simultaneously chosen to produce two different food packages. The main benefit of this machine is to increase the throughput of the packaging line. In fact, at each machine cycle a larger number of packages is produced two by two.

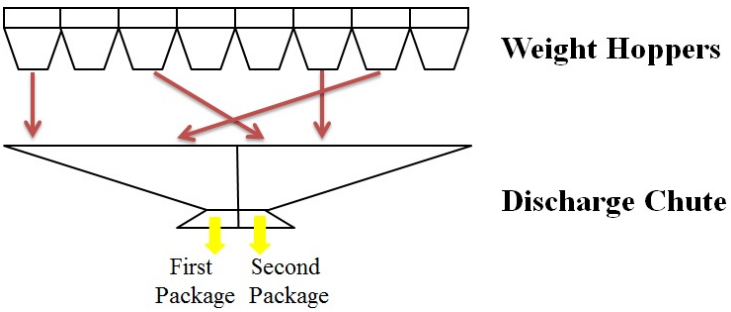


Figure 1.9: Duplex multihead weigher machine

1.2 Variable modeling

The weight of the product delivered in the j-th weighing hopper changes at each machine cycle, in fact it is subject to random fluctuation. As a consequence, this weight is modeled as a random variable $w_j,\ j = 1, 2, ..., H$ where H is the number of hoppers in the machine. By analyzing real data [Ferrario & Laghezza 1999], [Beretta 2010], [Incletolli 2012], it has been observed that:

- w_j is a realization of the random weight $W_j \sim N(\mu_j, \sigma_j^2),\ \mu_j > 0, j = 1, 2, ..., H$;
- each weight is independent of other weights $W_i (i \neq j)$;
- the standard deviation σ_j depends on the mean quantity μ_j according to a linear relation $\sigma_j = \alpha\mu_j$, where α is a proportionality constant that depends on the product to be packed ($0 < \alpha < 1$). This relation was also pointed out by [Kameoka, Nakatani & Inui 2000].

1.3 Problems

The multihead weigher machine is a complex machine which needs a setup strategy and a suitable operation software. The operation software works in real time and its goal is to select the best hopper subset to open according to the product, the time constraints and the objective function. This is a kind of knapsack problem. Instead, the setup problem deals with the definition of the average quantity of the product delivered by the radial feeders to each hopper at each cycle. This setting has to be done each time that the type of raw material (product), to be packed, changes or when a different target is required. An improper machine setup turns into a lack of machine efficiency.

1.3.1 Operation Problem

A package is characterized by a nominal weight, i.e. the Target weight value W_T, that has to be clearly labeled on the package. At each machine cycle, the combination control unit runs the operation software that has to find the best combination of hoppers in order to achieve the package target weight.

Moreover, for customer protection, the total weight of each package must be no less than this Target value (i.e., sometimes it is defined as the target weight constraint). In some countries, it is legally permitted to adopt a lower law limit weight W_L which is below the Target value W_T [Haze 1987] and so the package weight has to be greater than this limit. For instance, in Europe the [Council Directive 1976] imposes that the package weight has to be as close to the target weight as possible within a predetermined allowable error. This error is a fixed value in accordance with the 76/211/ECC law.

The problem that the combination control unit has to solve at each cycle is very similar to a particular knapsack problem: the subset sum problem [Martello & Toth 1990].

1.3.1.1 Problem formulation

Let us consider the easier kind of multihead weigher machine, thus composed by H pool hoppers and H weight hoppers. In literature, different knapsack formulations have been proposed: linear or absolute value objective function, linear constraint or different rules to choice the best combination. In this section, the state of the art regarding the problem formulation is tackled.

[Karuno, Nagamochi & Wang 2007] and [Imahori, Karuno, Nagamochi & Wang 2011] formulated the knapsack problem using linear objective function and a linear constraint. The problem was stated as follows:

$$\text{objective:} \qquad \text{minimize} \sum_{j=1}^{H} w_j x_j \tag{1.1}$$

$$\text{subject to:} \qquad \sum_{j=1}^{H} w_j x_j \geq W_T \tag{1.2}$$

$$x_j \in \{0,1\}\ j = 1, 2, ..., H \tag{1.3}$$

where

$$x_j = \begin{cases} 1, & \text{if the hopper } j \text{ is chosen;} \\ 0, & \text{otherwise.} \end{cases} \tag{1.4}$$

and Eq.(1.3) expresses the binary constraint of this variable. The objective by Eq.(1.1), together with Eq.(1.2), aims at attaining the total weight as close to the Target weight W_T as possible.

[Sakaeda & Sashiki 1986] proposed a formulation with the absolute value of the difference between the total weight and the Target weight value as the objective function. In their US4609058 patent, the problem was formulated as follows:

$$\text{objective:} \qquad \text{minimize } |\sum_{j=1}^{H} w_j x_j - W_T| \tag{1.5}$$

$$\text{subject to:} \qquad M_i \leq \sum_{j=1}^{H} w_j x_j \leq M_a \tag{1.6}$$

$$x_j \in \{0,1\}\ j = 1, 2, ..., H \tag{1.7}$$

where x_j is the binary variable expressed by Eq.(1.7) and its meaning is explained by Eq.(1.4), M_i is the minimum package weight allowed by the specific country and M_a is the maximum package weight, it is greater than the Target and can coincide with the maximum weight of the product that a package can handle. The objective by Eq.(1.5), together with Eq.(1.2), aims at obtaining that the total weight is lain in the range defined by M_i and M_a and it is as close as possible to the Target weight value.

[Beretta 2010] ([Beretta & Semeraro 2012], [Incletolli 2012]) formulated the knapsack problem using an absolute value function. They considered the presence of a lower law limit (W_L) and they introduced some empirical rules so that the best combination is chosen. The problem can be stated as follows:

$$\text{objective:} \qquad \text{minimize } |\sum_{j=1}^{H} w_j x_j - W_L| \tag{1.8}$$

$$\text{subject to:} \qquad x_j \in \{0,1\}\ j = 1, 2, ..., H \tag{1.9}$$

where, as stated before, x_j is the binary variable expressed by Eq.(1.9) and its meaning is explained by Eq.(1.4). The objective on its own, expressed by

	Lower	Upper
1	Exist	Exist
2	0	Exist
3	Exist	0
4	0	0

Table 1.1: Empirical rules inside the microprocessor.

Eq.(1.8), ensures that the total weight is as close as possible to the Target weight value but some rules are necessary to select the best solution. Firstly, the combinations are split into two subsets: the combinations whose weight is above or equal to the Target value are placed in the first subset and the other combinations in the second one. Secondly, the combination whose weight is the lowest in the first subset in selected and called Upper (W_{ub}), instead the combination whose weight is the greatest in the second subset is selected and called Lower (W_{lb}). The Lower value and/or Upper one may not exist. So the 4 situations, presented in Table (1.1), may occur. In particular:

Case 1: both Lower value and Upper one exists. If $W_{lb} \geq W_L$:

- Let define $\gamma = |W_T - W_{ub}|$ and $\beta = |W_T - W_L|$.
  If $\gamma = \beta$, the combination whose weight coincide with the Upper value is selected as the best one.
- If $\gamma \neq \beta$, the combination selected is the one associated with the value: $\begin{cases} \text{Lower}, & \text{if } \min(\beta,\gamma)=\gamma; \\ \text{Upper}, & \text{otherwise.} \end{cases}$

Otherwise, the combination is selected based on economic considerations.

Case 2: The Lower value does not exist, the Upper value combination is selected.

Case 3: The Upper value does not exist, the Lower value combination is selected.

Case 4: Both Upper value and Lower one do not exist. This situation occurs when all the hoppers are empty. The Lower Value combination (whose

weight value is equal to the Upper one) is selected. So an empty package is produced.

Sometimes the multihead weigher machine can implement a more complex operation strategy. For instance, during a series of machine cycles, an hopper may be left closed for a long time before it is chosen in a combination. To avoid this situation, especially when the machine handles fresh food, a priority γ_j for each hopper j is introduced. The priority γ_j is given by a non-decreasing function that explains how long the hopper j has been closed. Thus the problem can be formulate as a bi-criteria optimization problem [Karuno et al. 2007], [Imahori et al. 2011]:

$$\text{objective 1:} \quad \text{minimize} \sum_{j=1}^{H} w_j x_j \tag{1.10}$$

$$\text{objective 2:} \quad \text{maximize} \sum_{j=1}^{H} \gamma_j x_j \tag{1.11}$$

$$\text{subject to:} \quad \sum_{j=1}^{H} w_j x_j \geq W_T \tag{1.12}$$

$$x_j \in \{0,1\}\ j = 1, 2, ..., H \tag{1.13}$$

where, as stated before, x_j is the binary variable expressed by Eq.(1.13) and its meaning is explained by Eq.(1.4). The first objective, expresses by Eq.(1.10), together with the Target constraint Eq.(1.12), aims at minimizing the total weight. Instead the second objective, expresses by Eq.(1.11), wants to maximize the total priority. At each machine cycle, the combination with the minimum weight and the maximum priority is selected.

Let consider a double-layer multihead weigher machine, thus composed by H pool hoppers, H weight hoppers and H booster hoppers. In literature, only [Karuno et al. 2009], [Karuno et al. 2010] proposed a knapsack formulation for each kind of double layer machine, that is:

Upright machine

$$\text{objective:} \quad \text{minimize} \sum_{j=1}^{H} w_j x_j + \sum_{j=1}^{H} b_j y_j \tag{1.14}$$

$$\text{subject to:} \quad \sum_{j=1}^{H} w_j x_j + \sum_{j=1}^{H} b_j y_j \geq W_T \tag{1.15}$$

$$x_j - y_j \leq 0 \quad j = 1, 2, ..., H \tag{1.16}$$

$$x_j, y_j \in \{0, 1\} \quad j = 1, 2, ..., H \tag{1.17}$$

Diagonal machine

$$\begin{aligned} &\text{objective:} && \text{minimize} \sum_{j=1}^{H} w_j x_j + \sum_{j=1}^{H} b_j y_j \\ &\text{subject to:} && \sum_{j=1}^{H} w_j x_j + \sum_{j=1}^{H} b_j y_j \geq W_T \\ & && x_j + y_j \leq 1 \quad j = 1, 2, ..., H \\ & && x_j, y_j \in \{0, 1\} \quad j = 1, 2, ..., H \end{aligned} \tag{1.18}$$

where

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

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

and Eq.(1.17) expresses the binary constraint of these variables. The objective by Eq.(1.14), together with Eq.(1.15), aims at attaining the total weight as close to the Target weight W_T as possible. Eq.(1.16) represents the constructional constraint for an Upright machine (i.e., the weight hopper j can not be chosen unless its corresponding booster hopper j is chosen): $(x_j, y_j) \in \{(0,0), (0,1), (1,1)\}$. Instead Eq.(1.18) represents the constructional constraint for a Diagonal machine (i.e., the weight hoppers j and the corresponding booster hopper j can not be simultaneously chosen): $(x_j, y_j) \in \{(0,0), (0,1), (1,0)\}$.

Let consider a duplex multihead weigher machine, composed by H pool hoppers and H weight hoppers [Konishi 1983]. At each cycle time, this machine has to produce two different packages at the same time. The knapsack problem is formulated as follows [Imahori et al. 2010], [Imahori et al. 2012]:

$$\text{objective:} \qquad \text{minimize} \sum_{j=1}^{H} w_j x_j + \sum_{j=1}^{H} w_j y_j \tag{1.19}$$

$$\text{subject to:} \qquad \sum_{j=1}^{H} w_j x_j \geq W_T \tag{1.20}$$

$$\sum_{j=1}^{H} w_j y_j \geq W_T \tag{1.21}$$

$$x_j + y_j \leq 1 \quad j = 1, 2, ..., H \tag{1.22}$$

$$x_j, y_j \in \{0, 1\} \quad j = 1, 2, ..., H \tag{1.23}$$

where

$$x_j = \begin{cases} 1, & \text{if the hopper } J \text{ is chosen for the first package;} \\ 0, & \text{otherwise.} \end{cases}$$

$$y_j = \begin{cases} 1, & \text{if the hopper } i \text{ is chosen for the second package;} \\ 0, & \text{otherwise.} \end{cases}$$

and Eq.(1.23) expresses the binary constraint of these variables. The objective by Eq.(1.19), together with Eq.(1.20) and Eq.(1.21), aims at attaining the total weight as close to the Target weight W_T as possible for both packages. The constraint by Eq.(1.22) expresses the impossibility of giving out the weight hopper j to both subsets.

1.3.1.2 How to tackle the operation problem

A knapsack problem is usually solved by *enumerative algorithms*, that guarantee finding the optimal solution, or *approximate algorithms*, that determine feasible solutions, whose value is a upper bound (in a minimization problem) of the optimal solution value, but in a faster way [Martello & Toth 1990].

The most popular enumerative algorithms are:

Brute Force , or exhaustive search or complete enumeration, is a very general problem-solving technique that consists of systematically enumerating all possible candidates for the solution and checking whether each candidate satisfies the problem statement. While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions - which in many practical problems tends to grow very quickly as the size of the problem increases. Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed.

Branch and Bound is an approach developed for solving different optimization problems, typically discrete and combinatorial optimization problems. A branch-and-bound algorithm searches the entire space of candidate solutions, and discards a large part of the search space by using upper and lower estimated bounds of the quantity being optimized. The use of these bounds combined with the value of the current best solution has the benefit to enable the algorithm to search parts of the solution space only implicitly, thus to speed up the search of the optimal solution [Land & Doig 1960], [Kolesar 1967].

Dynamic Programming is a method for solving complicated problems by breaking them down into a reasonable number of simpler subproblems. In this way an optimal solution is found for each subproblem and then the optimal solution of the initial problem is reached combining all the subproblem optimal solutions. This approach seeks to solve each subproblem only once: its optimal solution is stored and this solution is reused several times when the same partial result is needed. In this way, the number of computations are reduced. The dynamic programming is especially useful when the number of repeating subproblems grows exponentially as a function of the input size [Martello & Toth 1990].

1.3.2 Setup Problem

A machine setup consist of specifying the values of the setpoints $(\mu_1, \mu_2, ..., \mu_H)$, to which according to the assumption in Section(1.2) also determine the hopper weight variances $\sigma_j^2, j = 1, ..., H$.

Nowadays, despite the wide spread of the multihead weigher in the production world, the definition of a machine setup in the firms mainly relies on the operator's skill and experience. Instead in literature, according to my best knowledge, [Beretta 2010],[Beretta & Semeraro 2012], are the first that tackled the setup problem using a statistical approach. This problem was handled assuming that the weight of the single product is so small that the product weight delivered in each hopper can be described by a continuous random variable. A RSM (Response Surface Methodology) is used to find out the best setup for two different multihead weigher machines of an Italian food firm. Then, [Incletolli 2012] solved the setup problem when the weight of the single product is too large that needs to be approximated by a discrete random variable.

Chapter 2

Problem Definition

The selection of the setpoints of a MWM is critically important since it directly affects the quantity of product delivered in each package, and consequently the production costs for the producer, and the loss to the costumer due to the deviation of the package weight from specification. In this chapter a method to evaluate the performance of a setup machine is proposed.

2.1 Performance Evaluation

A MWM works by weighing the amount of product into each hopper and selecting the best hopper combination in order to fill a package whose weight is as close to target weight as possible. Then, the downstream checkweigher weighs each package and takes out of the line any package with a weight less than the desired weight. This problem reminds us the well-known "filling problem" or "canning problem". In fact, a canning problem is concerned with placing a specific amount of product into a package subject to a set of specifications.

Most of the literature on container-filling process deals with the problem of selecting the mean (single-level canning problem) or both the mean and the Upper weight limit (two-level canning problem) for the product quantity delivered in a package in each cycle, in order to optimize an economical objective function. Setting this mean to a very low level can reduce the production costs but it will increase the rejection cost due to more "non-conforming" packages. A package filled with a quantity of product below

the desired weight is defined "non conforming". On the other hand, setting the package mean to a very high level reduces the rejection cost but it will increase the production costs. The objective function has to face this trade-off considering on one side the costs due to "non conforming" packages that can therefore be scrapped or reworked and on the other the costs due to the extra-product, arising when packages are filled with a quantity of product that exceeds the target weight.

Research has been continuing from the 1950s to solve the canning problem under different conditions. Springer [1951], Bettes [1962], Hunter & Kartha [1977], Nelson [1979], Bisgaard, Hunter & Pallessen [1984], Carlsson [1984], Golhar [1987], Golhar & Pollock [1988] and Schimdt & Pfeifer [1989] addressed this problem assuming a 100% inspection of the final packages and they used a normal random variable to describe the behavior of the package weight. Carlsson [1989], Boucher & Jafari [1991] and Al Sultan [1994] considered the case of sample inspection, then Pulak & Al Sultan [1989], Al Sultan & Al-Fawzan [1997] and Al Sultan & Pulak [1997] dealt with the problem assuming a rectifying inspection. In all these studies they proposed an economic objective function based on the expected net profit per packages (i.e. the difference between the expected selling price per package and the expected production costs).

Summing up, in this work, the economic objective function considered is presented in the next section. Moreover, differently from the literature, the mean of the final package cannot be directly optimized because it depends on the the machine setup. Thus, the focus of the MWM canning problem is on the jointly determination of the optimal setpoints for a specific machine.

2.1.1 Economic Objective Function

In order to evaluate the performance of a machine setup, an economic objective function based on the production costs is considered. Both 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). Quality influences performance that the machine can attain through the number of "non conforming" packages. Indeed packages

that contain a quantity of product below the desired weight are considered "non conforming" and cannot be sold to the final customer. These packages are emptied and refilled by the same filling process. Sometimes a package cannot be reworked but has to be scrapped, for instance when perishable products are handled.

Let us define:

C_{DEP} (€) depreciation of the machine.

C_{EN} (€) energy cost.

CE (€/year) energy cost per year.

$CMAT_{(W \geq W_L)}$ (€¢) cost of raw material regarding the "conforming" packages.

$CMAT_{(W < W_L)}$ (€¢) cost of raw material regarding the "non conforming" packages.

CS (€¢) cost of sorting.

CP (€¢) cost of the packaging material.

c_{pack} (€/kg) cost of the packaging material per kilogram.

c_u (€¢/g) cost of the raw material per gram.

δ dummy variable to consider both rework and scrap packages.

DEP (€/year) depreciation of the machine per year.

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

$E(W|W < W_L)$ (g/package) expected weight value of the "non conforming" packages.

LAB (€/h) labor cost per hour.

n_B number of "conforming" packages.

P number of packages produced in a specific time interval.

Q_{pack} (kg/package) quantity of the packaging material per package.

t (hours/package) production time.

W (g/package) quantity of product delivered in each package, it can be modeled as a random variable with an unknown distribution.

W_L (g/package) legal weight of the package. The tolerable negative error in the contents of a prepackage is fixed in accordance with the Table (2.1) below:

W_T in g or ml		Tolerable negative error	
from	to	as % of W_T	g or ml
5	50	9	-
50	100	-	4.5
100	200	4.5	-
200	300	-	9
300	500	3	-
500	1000	-	15
1000	10 000	1.5	-

Table 2.1: Tolerable negative error

W_T (g) Target weight value of a package.

TI (s) time used to take care of the "non conforming" packages.

WH (hours/year) working hours of the machine per year.

The economic objective function is based on the production costs per "conforming" package in a specific time interval, thus the number of packages filled P is fixed. The idea is to allocate the total production costs only on the packages that are sold in the final market (n_B). The total production costs (C_T) is characterized by:

$CMAT$ is the cost of raw material, equal to the product of the raw material cost per gram cu and the quantity of filled product W in the package and the package number P produced in a time interval.

CS is the sorting cost to take care of the "non conforming" packages. If a package can be reworked, this cost element 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 discarded, this cost item considers the time used by the operator to take care of it.

CP is the packaging material cost, equal to the product of the packaging material cost per kilogram c_{pack} and the quantity of packaging material used Q_{pack} to pack P packages in the time interval.

C_{DEP}, C_{EN} are respectively the depreciation and the energy costs to produce P packages in the time interval.

The quantity of product W delivered in each package is a random variable, thus the expect value of the production costs is considered and it is formulated as follows:

$$E(C_T) = E(CMAT_{(W \geq W_L)} + CP + C_{DEP} + C_{EN} + CS + \delta CMAT_{(W < W_L)}) \tag{2.1}$$

$$\begin{aligned} E(C_T) \; = \; & c_u \cdot E(W|W \geq W_L) \cdot n_B + \\ & + \delta \cdot c_u \cdot E(W|W < W_L) \cdot \left(P - n_B\right) + c_{pack} \cdot Q_{pack} \cdot P + \\ & + \left(\frac{DEP + CE}{WH}\right) + LAB \cdot TI \cdot \left(P - n_B\right) \end{aligned} \tag{2.2}$$

where:

$$
\begin{aligned}
E(CS) &= LAB \cdot TI \cdot (P - n_B) && (2.3)\\
E(C_{DEP}) &= \left(\frac{LAB}{WH}\right) \cdot t \cdot P && (2.4)\\
E(C_{EN}) &= \left(\frac{CE}{WH}\right) \cdot t \cdot P && (2.5)\\
E(CP) &= c_{pack} \cdot Q_{pack} \cdot [n_B + (P - n_B)] && (2.6)\\
E(CMAT_{(W \geq W_L)}) &= c_u \cdot E(W|W \geq W_L) \cdot n_B && (2.7)\\
E(CMAT_{(W < W_L)}) &= c_u \cdot E(W|W < W_L) \cdot (P - n_B) && (2.8)
\end{aligned}
$$

and t is the production time that, in this analysis, is approximated by the number of packages produced in a specific time interval.

$$t = 1/P \tag{2.9}$$

Substituting from Equation (2.3) to Equation (2.9) into Equation (2.1), the expected value of the production costs is:

$$
\begin{aligned}
E(C_T) \;=\; & c_u \cdot E(W|W \geq W_L) \cdot n_B + \\
& + \delta \cdot c_u \cdot E(W|W < W_L) \cdot \big(P - n_B\big) + c_{pack} \cdot Q_{pack} \cdot P + \\
& + \left(\frac{DEP + CE}{WH}\right) + LAB \cdot TI \cdot \big(P - n_B\big)
\end{aligned}
\tag{2.10}
$$

The production cost per "conforming" package is defined as follows:

$$
\begin{aligned}
E(C_T) \;=\; & c_u \cdot E(W|W \geq W_L) + \\
& + \frac{P - n_B}{n_B}\left(\delta \cdot c_u \cdot E(W|W < W_L) + LAB \cdot TI\right) + \\
& + c_{pack} \cdot Q_{pack} \cdot \frac{P}{n_B} + \frac{1}{n_B}\left(\frac{DEP + CE}{WH}\right)
\end{aligned}
\tag{2.11}
$$

$$
\begin{aligned}
E(C_T) \;=\; & c_u \cdot E(W|W \geq W_L) + \\
& + \frac{P - n_B}{n_B}\left(\delta \cdot c_u \cdot E(W|W < W_L) + \underbrace{LAB \cdot TI}_{c_l}\right) + \\
& + \underbrace{c_{pack} \cdot Q_{pack}}_{c_p} \cdot \frac{P}{n_B} + \frac{1}{n_B}\left(\underbrace{\frac{DEP + CE}{WH}}_{c_f}\right)
\end{aligned}
\tag{2.12}
$$

Using the short notation c_l, c_p and c_f the object function also known as total cost function is now:

$$
\begin{aligned}
E(C_T) \;=\; & c_u \cdot E(W|W \geq W_L) + \frac{P}{n_B} \cdot c_p + \frac{1}{n_B} \cdot c_f + \\
& + \frac{P - n_B}{n_B}\left(\delta \cdot c_u \cdot E(W|W < W_L) + c_l\right)
\end{aligned}
\tag{2.13}
$$

where $\delta \begin{cases} 0, & \text{if the product is reworable} \\ 1, & \text{otherwise} \end{cases}$

Since two types of strategy have been considered thanks to the delta (δ), two semplified cost functions are obtained. Let δ be equal to 0 (i.e. the raw material cost for the "non conforming" is avoided), the simplified objective function for reworkable material cases follows:

$$
E(C_T) \;=\; c_u \cdot E(W|W \geq W_L) + \frac{P}{n_B} \cdot c_p + \frac{1}{n_B} \cdot c_f + \frac{P - n_B}{n_B} \cdot c_l \tag{2.14}
$$

Otherwise, when δ is equal to 1, if the raw material inside a package is less than the legislation limit W_L, the package is labeled as "non conforming" and will be discharged. The simplified equation is:

$$
E(C_T) \;=\; \frac{P}{n_B} \cdot c_u \cdot E(W) + \frac{P}{n_B} \cdot c_p + \frac{1}{n_B} \cdot c_f + \frac{P - n_B}{n_B} \cdot c_l \tag{2.15}
$$

The objective functions (Equations (2.14) and (2.15)) need an estimate of the $E(W)$. del Castillo, Beretta & Semeraro [2014] showed that the computational expense of the expression needed to obtain the first moment

$E(W|W \geq W_T)$ is too large in general. Thus, in the next section, a simulator of the machine is proposed in order to evaluate the performances given a specific machine.

2.2 The Simulator

The simulator, presented in Figure (2.1), mimics the behaviour of a simple MHW machine. Given in input a setpoint $(\mu_1, \ldots, \mu_H)$, the minimum package weight and other parameters the simulator estimates via Monte Carlo the resulting C_{tot}. At each iteration, a knapsack problem is solved to select the combination of hoppers to open. In the next section the specific knapsack problems are presented.

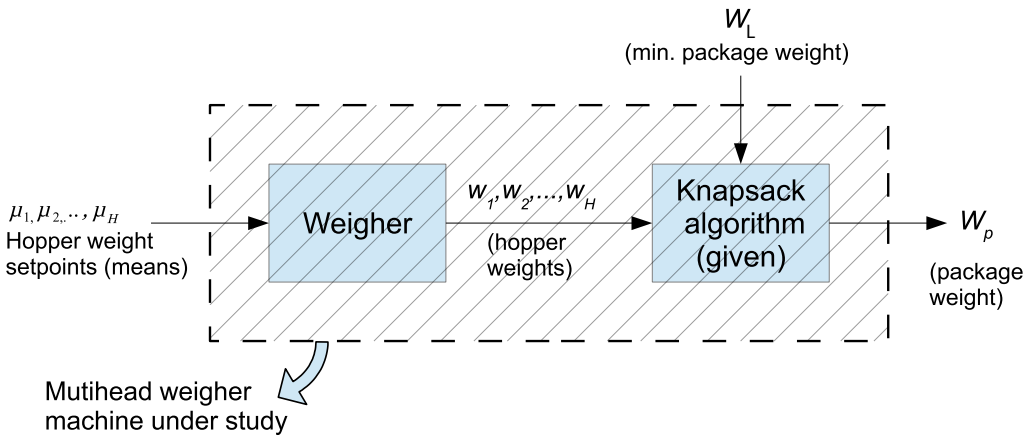


Figure 2.1: Simulator operating principle

The coding of the Simulator (see Fig. (2.2)) faced some issue since loops, such as *for* or *while*, where not a viable solutions. The workaround that made possibile the Simulator construction was using matrix calculus to parallelize and therefore reduce the computational effort. To be fair, this approach present a drawback, since the matrix operations are simultaneously, *Matlab*® needs to write swap files[1]. These files are necessary since the RAM (Random-access memory) is not always enough to contain all the matrices used in the

[1]It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory

simulator. The swap allocation reach its limit with complex problems ($H \geq 8$ and $L \geq 50$) since matrices of $1.652{\times}10^9$ elements ($H = 8, L = 50$) needs very large Hard-disk to be properly stored.

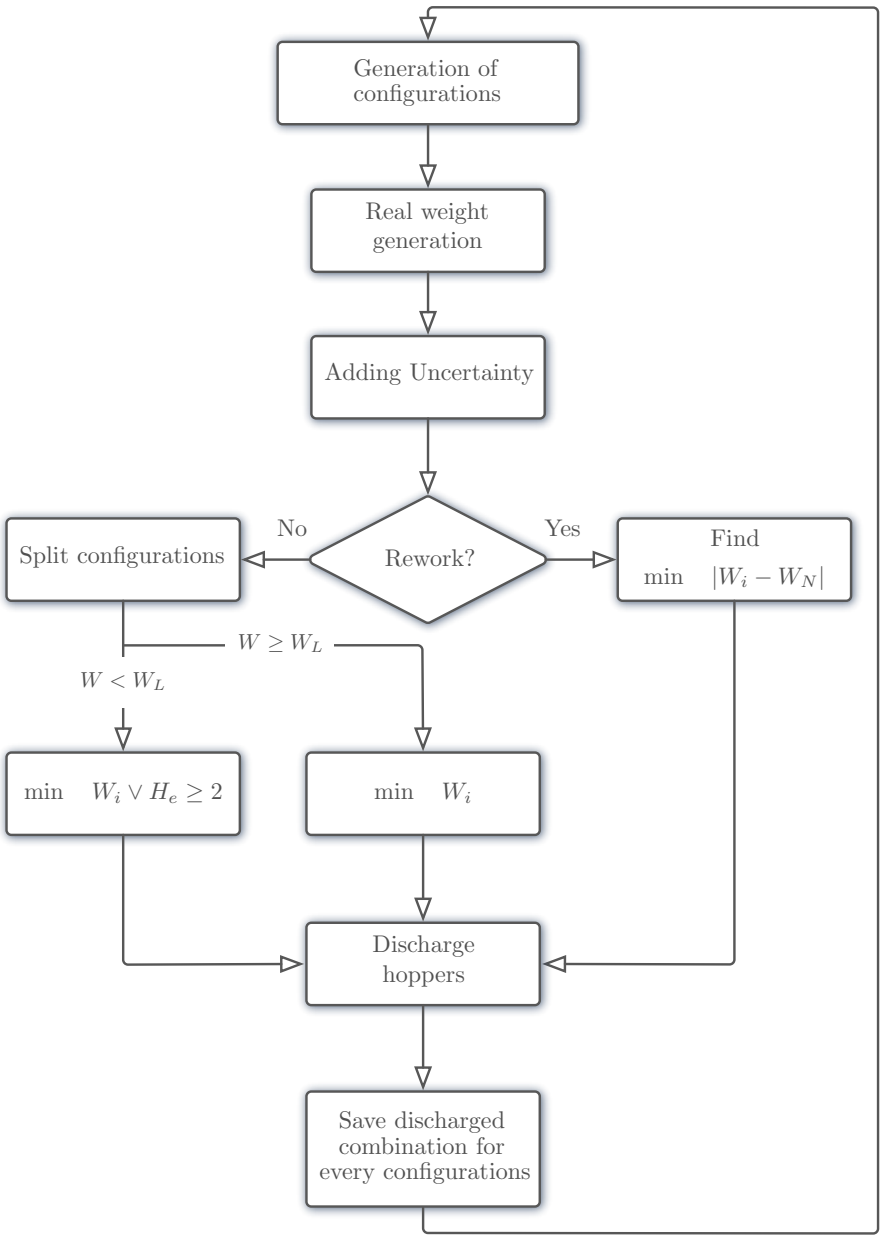


Figure 2.2: Simulator flow chart

2.2.1 The Multihead Weiger Knapsack

The knapsack problem can be defined as the problem that the combination control unit has to solve at each cycle, in order to discharge the "best setup" given by the 2^H possibile combinations. While there are different knapsack formulations that have been reported in the litrature (Section 1.3.1.1), most of them utilize linear objective function(s) and linear constraints. In this section, we assume that the machine has a built-in algorithm that solves, for each package, one of the two types of deterministic knapsack problem: the knapsack for the reject case and the knapsack for the rework case.

Rework

If a "non conforming" packages is reworkable, the knapsack problem minimize the absolute distance between the package weight ($\sum_{j=1}^{H} x_j w_j$) and the target value W_T. It is formulated as follows:

$$\text{objective:} \qquad \text{minimize} \quad \left| \sum_{j=1}^{H} x_j w_j - W_T \right| \tag{2.16}$$

where x_j is equal to 1 if the hopper is chosen in the hopper combination to open, otherwise x_j is set equal to 0. At each iteration, the algorithm selects the hopper combination whose weight is close as possibile to W_T. Note that the W_L constraint does not appear in this simulation. In fact, if the hopper combination selected has a weight lower than W_L this means that this combination has a weight difference lower than all the other combinations. In this case the package is reworked. It is implied that we are assuming that to rework a package costs less than to fill a package with a lot of products inside.

Reject

The knapsack for the reject case requires two different objective function and different constraints given that discard material implies additional costs. Thus the knapsack problem aims to fulfill the package with less material as possibile accordingly with the legislation limit W_L. Its formulation follows:

$$
\begin{cases}
\text{objective:} & \text{minimize } \sum_{j=1}^{H} x_j w_j & (2.17a) \\
\text{subject to:} & \sum_{j=1}^{H} x_j w_j \geq W_L & (2.17b) \\
 & x_j \in \{0,1\}\ j = 1,2,...,H & (2.17c) \\
\text{objective:} & \text{minimize } \sum_{j=1}^{H} x_j w_j & (2.17d) \\
\text{subject to:} & \sum_{j=1}^{H} x_j w_j < W_L & (2.17e) \\
 & H_e \geq 2 & (2.17f) \\
 & x_j \in \{0,1\}\ j = 1,2,...,H & (2.17g)
\end{cases}
$$

where H_e is the number of hoppers that are opened to produce the i-th package. If no hopper combination has a weight greather than W_L (Equation 2.17e), all the combinations involving the discarge of at least 2 hoppers are analyzed and then the one with minimum weight is selected to open (Equation 2.17d). The minimum of 2 hoppers gives a fair confidence to avoid to be stuck, for a few iterations, in combinations that may reduce the MHW performances. Otherwise, if at least one hopper combination is able to fullfill the constraint (Equation 2.17b), then the combination with minimum weight is selected and a conforming package will be produced.

According to the simulation input parameters displayed in Table (2.2), the comparison between the rework and the reject case (performed with a Brute-force[2] enumerative algorithm) is presented in Table (2.3). Table (2.3) shows that the expected values ($E(W^*)$) of the two approaches are far from to be the same. Specifically, the rework case has a $E(W^*)$ value near to the target value W_T respect to the reject. Since the discretization of the solution

[2]See Section 2.3 for a deep insight.

space is rough (coarsness = 10 g), the expected value of the reject case can not reach the lowest value W_L.

	Parameters					
Simulator	H	5		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	L	19		c_l	3.2	€¢/h
	H_e	2		c_f	7.34	€/p
	W_N	250	g	W_L	241	g

Table 2.2: Simulation parameters

The objective function values comply with the expected valued since the rework case exhibit an higher value caused by the higher $E(W^*)$ value. Although, the reject case, with a more complex knapsack problem needs more time to find optimal combination to be open.

	$\boldsymbol{Reject(\delta = 1)}$	$\boldsymbol{Rework(\delta = 0)}$
$E(W^*)$	246.0664	249.9504
$\sigma(W^*)$	4.8070	3.4557
$f(\boldsymbol{\mu^*})$	8.2267	8.3505
n_B	3000	2995
t_{tot}	$\sim$ 338" (5' 38")	$\sim$ 214 (3' 24")
$\boldsymbol{\mu}$	20 30 100 110 120	20 30 40 50 180

Table 2.3: Standard approach - Rework *v.* Reject

2.2.2 Iterative approach

To use the Economic objective function (see Section 2.1) iteratively during the Brute-Force search gives the opportunity to link the quantity delivered

in each package with the cost of that package. Therefore, evaluating production costs instead of weights has a more industrial meaning. Moreover, the iterative approach works on the continuous updating of n_B (Number of "conforming" packages) during the production batch.

Let is consider the flow chart in Figure (2.3), after that the algorithm has calculated the weight of all the hopper combinations, it splits the combination into two clusters: the combinations whose weights is greather than W_L is placed in the first cluster, otherwise in the second. Then the algorithm evaluates the objective function $f(\boldsymbol{\mu}) = {}^{C_T}/_{n_B}$ and selects the best combination from each cluster. The combination with the greatest of is picked. If the combination selected has a weight lower than W_L a "non conforming" package is filled, and the number of "conforming" is not updated. Therwise this number is increased by one.

According to the simulation parameters in Table (2.4), Fig. (2.4) presents the performance of the two methods, using a non-optimal machine setup to produce 3000 packages. As expected, given the iterative behavior and the ability of choosing the next package to discharge based on the previous one, the iterative approach outperforms respect to the standard in all the non optimal configurations.

	Parameters					
Simulator	H	5		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	L	19		c_l	3.2	€¢/h
	H_e	2		c_f	7.34	€/p
	W_N	250	g	W_L	241	g

Table 2.4: Simulation parameters

Table (2.5) shows the performances of both approaches (iterative and standard) when a optimal machine setup is selected. It can be noted, despite the conclusion drawn from the previous graph, the iterative approach, in an optimal setup configuration, performs slightly better thanks to a lower

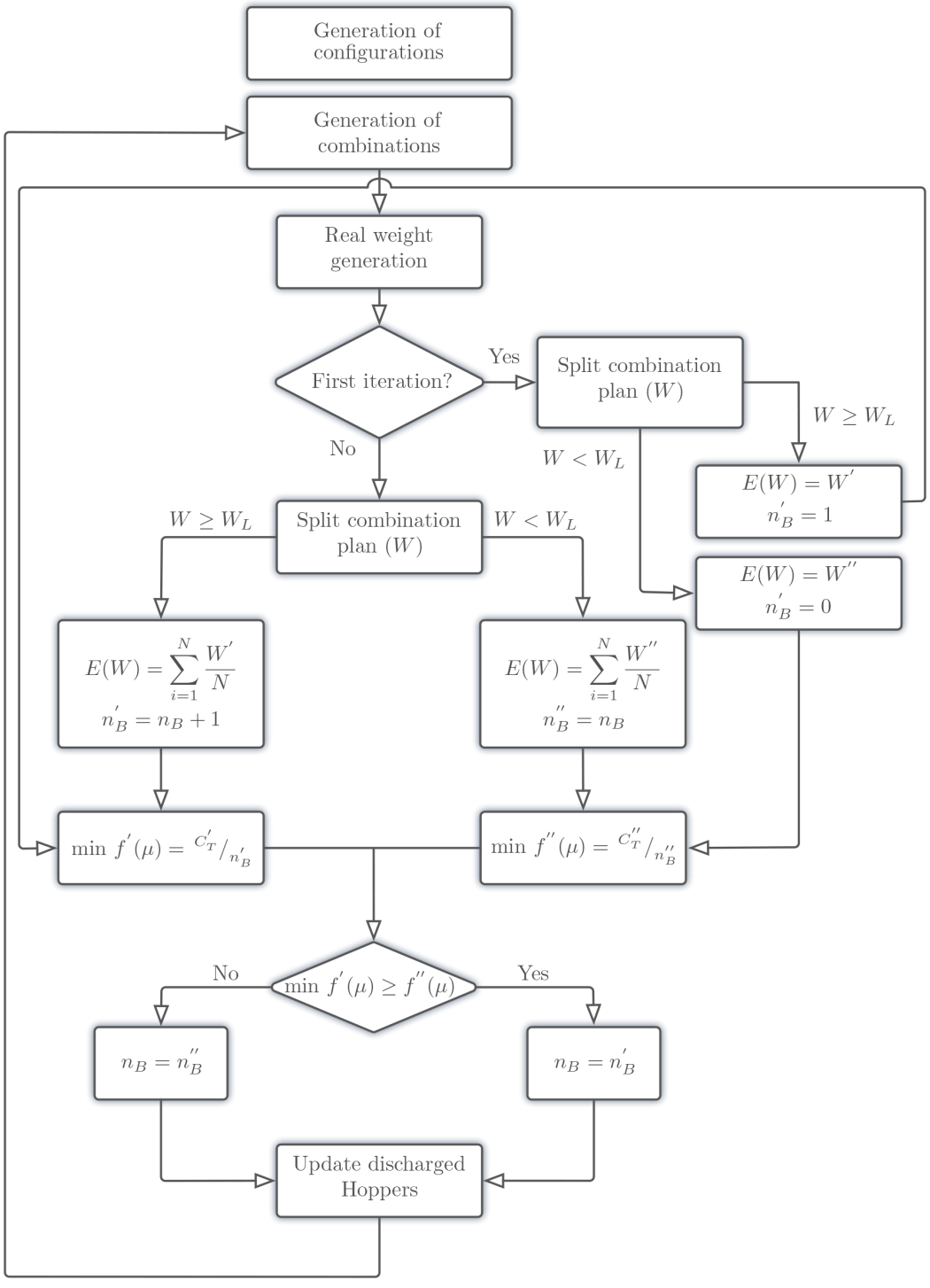


Figure 2.3: Iterative rules flow chart

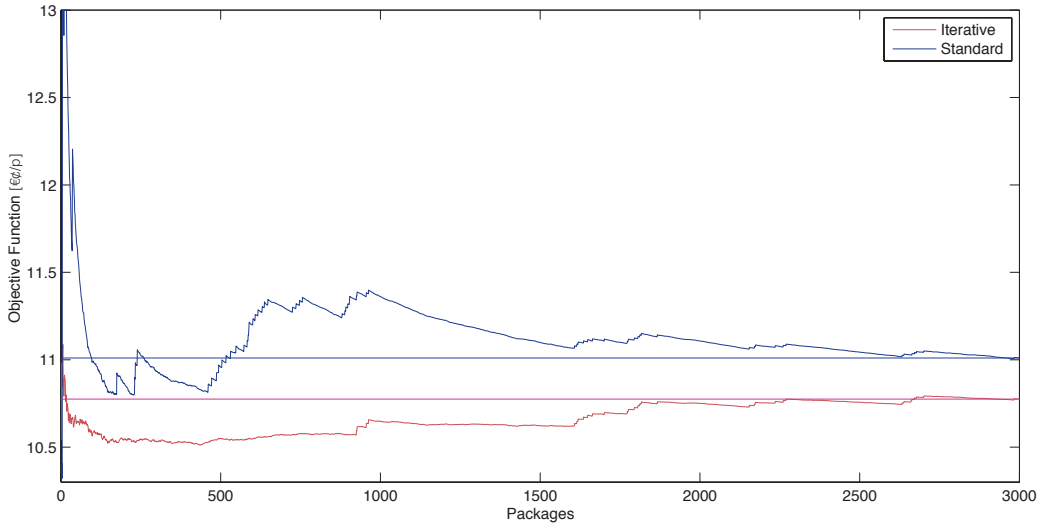


Figure 2.4: Iterative *v.* Standard approach in a non-optimal setup

$E(W^*)$ value. Unfortunately the outperformance comes with a computational time increase of four times.

	Standard	***Iterative***
$E(W^*)$	246.0664	246.0300
$\sigma(W^*)$	4.8070	4.7899
$f(\boldsymbol{\mu^*})$	8.2267	8.2256
n_B	3000	3000
t_{tot}	$\sim$ 338" (5' 38")	$\sim$ 1427 (23' 47")
$\boldsymbol{\mu}$	20 30 100 110 120	20 30 100 110 120

Table 2.5: Iterative *v.* Standard approach

According to these results and the trend noticeable in Figure (2.5), the iterative approach outperforms respect the standard approach but since the setup configuration consists in finding an optimal solution, the objective function gap is not enought to legitimize a computational time of four times.

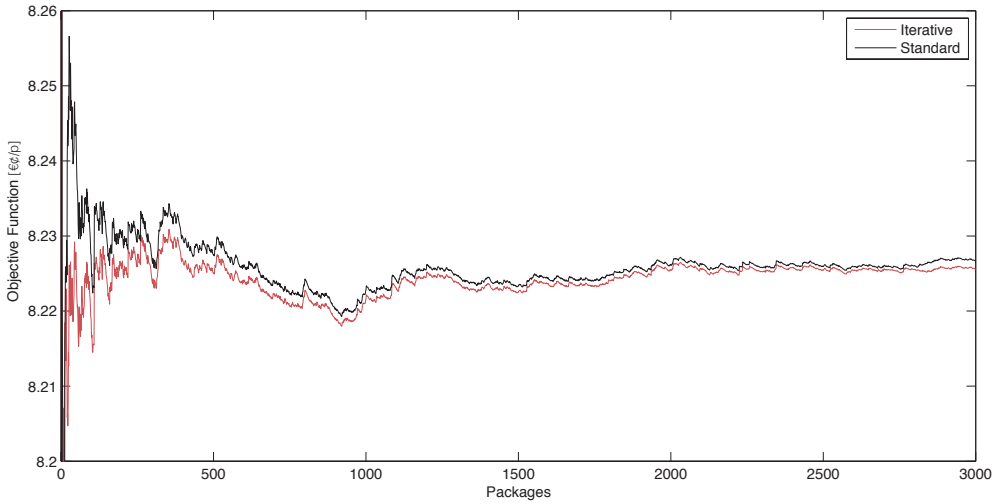


Figure 2.5: Iterative *v.* Standard approach in a optimal solution

2.2.3 Convergence & Oscillations

The oscillations and the convergence limit of W, quantity delivered in each package, has been discovered thanks to the brute-force search. Specifically, it was possible to gain knowledge about the number of packages P produced in a specific time interval that minimize the oscillations. The number of packages produced into a time interval, also called *production campaign*, affects the convergence both of weights W and objectiveive function $f(\boldsymbol{\mu})$. In order to find out the convergence point, it has been chose not to rely on the iterative approach that comport long computation time, chosing, instead, a more euristic method to simulate a pruduction campaign of 10 000 packages, to choose the best setup configuration discovered by solving the knapsack problems (see Section 2.2.1) and then to plot $E(W^*)$, $\sigma(W^*)$ and $f(\boldsymbol{\mu^*})$ was usefull to understand how those signals have unsettled trends. As viewable in Figure (2.6) the packages cumulative mean tends to a convergence value of 246.3. The empiric method consists in suppose a fair range to create a bandwitch around the signal and than discover from what point on the signal never goes cross the boundaries.

Using a narrow confidence interval, such as the 0.05% of the optimum average weight (W^*), that is roughly 0.123 g, the convergence point can be defined at 3000^{th} pakage (see Fig. (2.6)). At that point, the cumulative mean

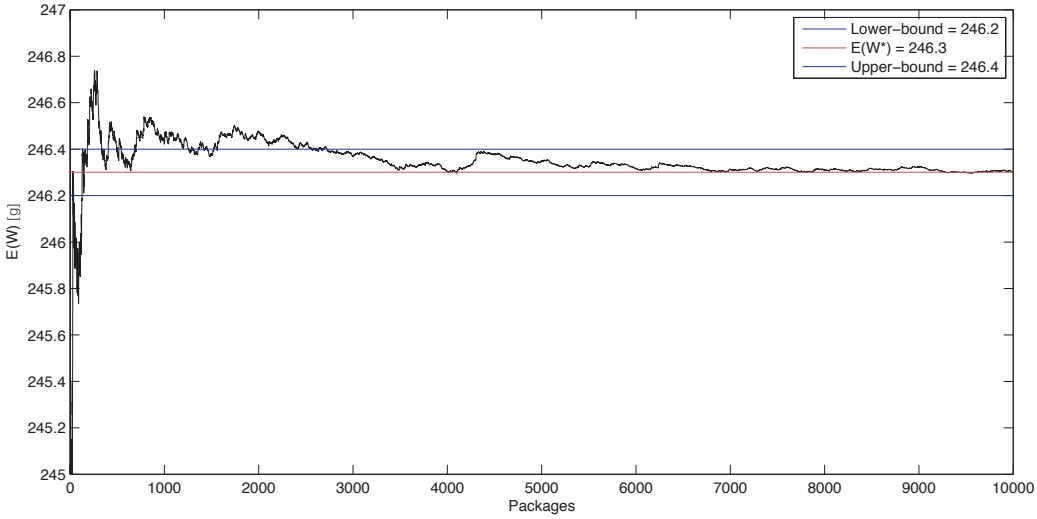


Figure 2.6: Packages cumulative Mean $E(W^*)$

lays inside the range and it never oversteps those boundaries.

$$246.2 \leq E(W^*) \leq 246.4$$

The same approach can also be used for the standard deviation proving not to be less effective given that the bandwidth of 2.545% renders a variation of 0.122 g, the same width of the mean displayed before (see Fig. (2.7)).

$$4.710 \leq \sigma(W^*) \leq 4.956$$

The Economic objective Function acts in the same manner of the packages cumulative mean only with a muffled effect thanks to the cost of the filled product c_u. The cost coefficient, being lower than one, dampens the unsettled trend of the average $E(W)$ resulting in a more stable cumulative evolution. This muffled effect is straightforward since the bandwidth of 0.05% (roughly 0.0040 €¢) of the objective function optimum ($f(\boldsymbol{\mu}^*)$) works well even though it is much narrower respect to the Mean bandwidth (see Fig. (2.8)).

$$8.0781 \leq f(\boldsymbol{\mu}^*) \leq 8.0861$$

Summing up, the convergence point seems to be 3000 packages since from that point on the cumulative mean, standard dev and objective function

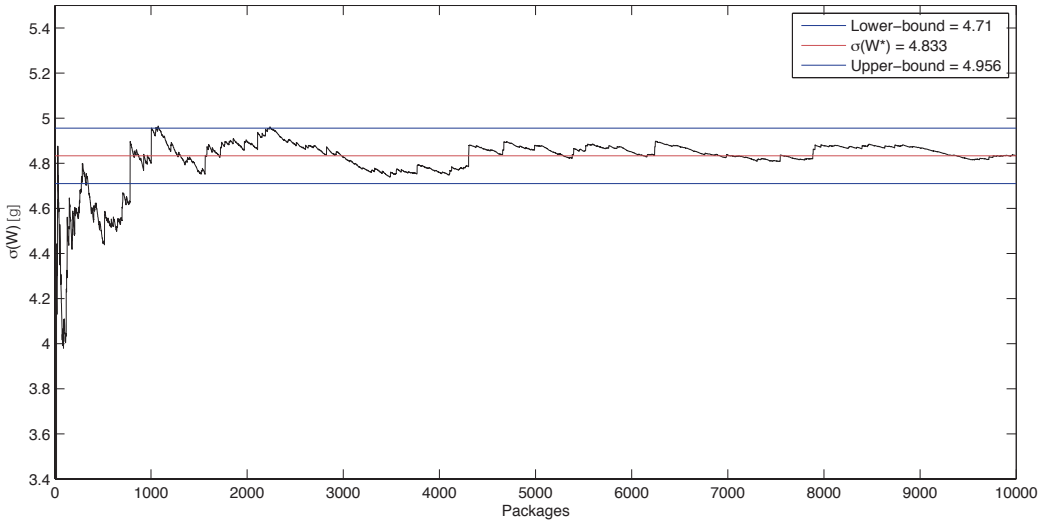


Figure 2.7: Packages cumulative Standard deviation $\sigma(W^*)$

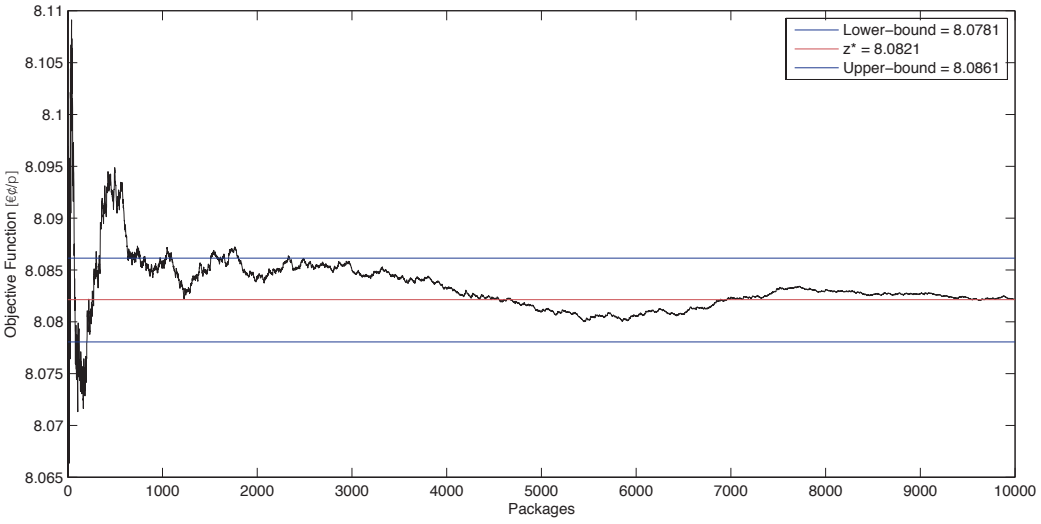


Figure 2.8: Packages cumulative Objective function $f(\boldsymbol{\mu}^*)$

never leave their bandwidth. With a roughly estimation it will be able to accept even 1000 as convergence point since the objective function, thanks to the damp coefficient C_u, exibits an highly unsettled trend only from 0 to 900 packages.

2.2.4 Uncertainty of the Loading Cell

The uncertainty of the Loading Cell affects the objective function performances. The more the measured weight shifts from the real one, the more the package weight differs from its real weight. The main feature of the Loading Cell was gathered from OIML R111 patent which evaluates the performance based on the relationship between the error and the maximum achievable target weight:

$$\varepsilon = \delta \cdot W_N$$

where ε is the maximum admissible error and δ is a Loading Cell coefficient. There are some different class of Loading Cell with different precision (see Table (2.6)). All simulations in this research are performed with a MHW E1 class.

Class	E1	E2	F1	F2	M1	M2	M3
δ	0.5×10^{-6}	1.5×10^{-6}	5×10^{-6}	15×10^{-6}	50×10^{-6}	150×10^{-6}	500×10^{-6}

Table 2.6: Loading Cell Class

2.3 Brute-Force Analysis

The Brute-Force search has been used to gain more knowledge about the problem and to discover some euristic rules. One of the main reasons about using brute-force is the need to analyze the solution space and to undestand how many solutions and where they are located. The MHW machine problem according to the Economic Object Function shown in Section 2.1, exhibits an high optimal region. The analisys of such region gives us the ability to find some heuristic rules[3] that can reduce the solution space and test only those solutions that can be valid fits.

[3]The K-Ratio Rules (stated in Section 3.2) is one of the primary rules that can reduce the number of the available solutions.

Since the problem dimension is grather than 3, the Euclidean distance has been used to "aggragate" the number of hoppers for each configuration in order to plot and display some significant behaviour. The Euclidean distance between configurations $\boldsymbol{\mu}_p$ and $\boldsymbol{\mu}_q$ is the length of the line segment connecting them ($\overline{pq}$). In Cartesian coordinates, if $\boldsymbol{\mu}_p = (\mu_{p1}, \mu_{p2}, \ldots, \mu_{pn})$ and $\boldsymbol{\mu}_q = (\mu_{q1}, \mu_{q2}, \ldots, \mu_{qn})$ are two points in the Euclidean n-space, then the distance from p to q, or from q to p is given by:

$$\begin{aligned} d(\boldsymbol{\mu}_p, \boldsymbol{\mu}_q) &= \sqrt{(\mu_{p1} - \mu_{q1})^2 + (\mu_{p2} - \mu_{q2})^2 + \ldots + (\mu_{pn} - \mu_{qn})^2} \\ &= \sqrt{\sum_{i=1}^{n} (\mu_{pi} - \mu_{qi})^2} \end{aligned} \tag{2.18}$$

To plot the Euclidean distance for all configurations against their objective function values gives us the opportunity to discover that a large amount of solutions are placed near the optimal solution. This means that very similar configurations can have very different performances. As viewable in Figure (2.10), there are stacks of configurations with the same distance from the optimum but with different objective function values. This difference becomes greater as the distance from the optimum become larger.

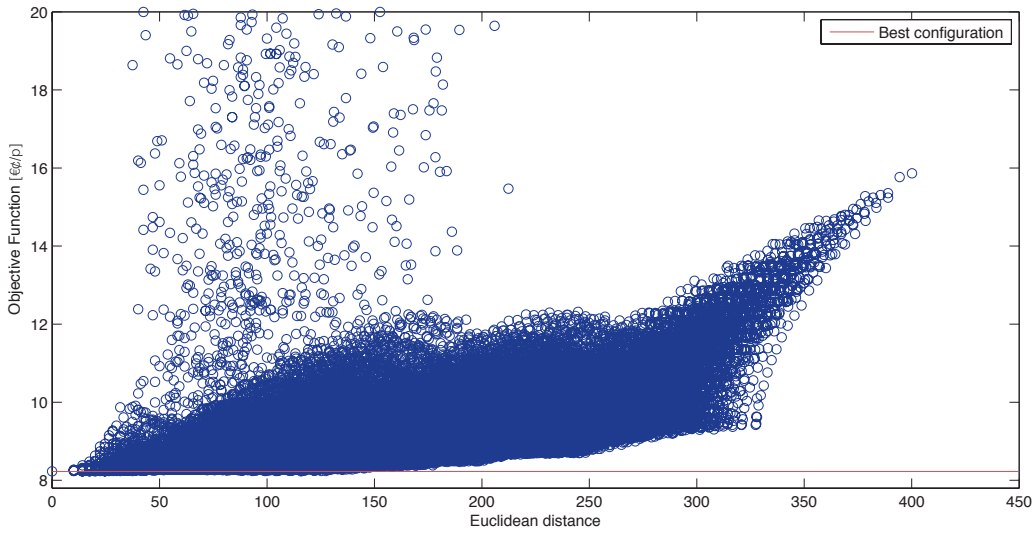


Figure 2.9: Euclidean distance

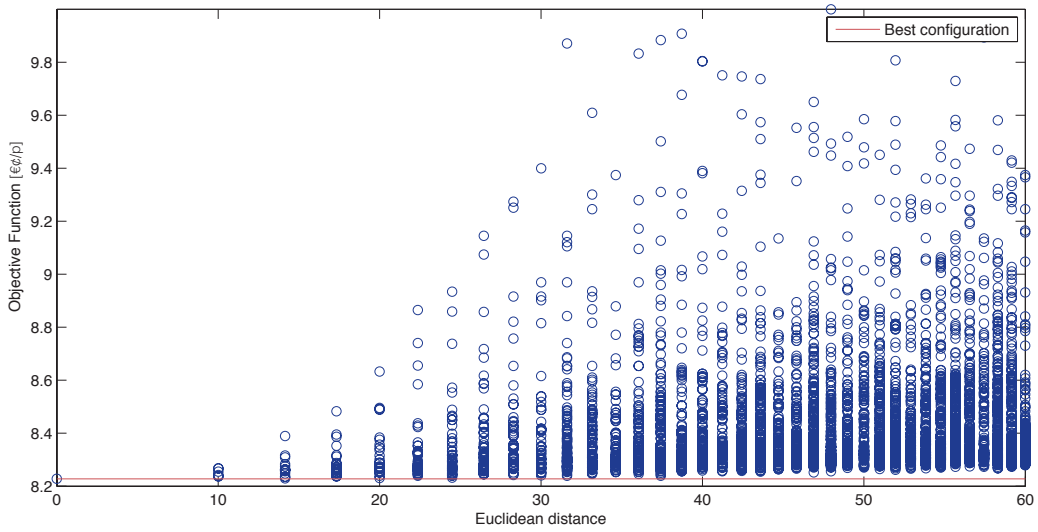


Figure 2.10: Euclidean distance - rescaled axis

Plotting the top 10% of the configurations ($142506 \cdot 10\% = 14251$), based on their objective function values, it is possibile to appreciate that the number of occurrences becomes higher very quickly with the increase of the objective function value (see Fig. (2.11)).

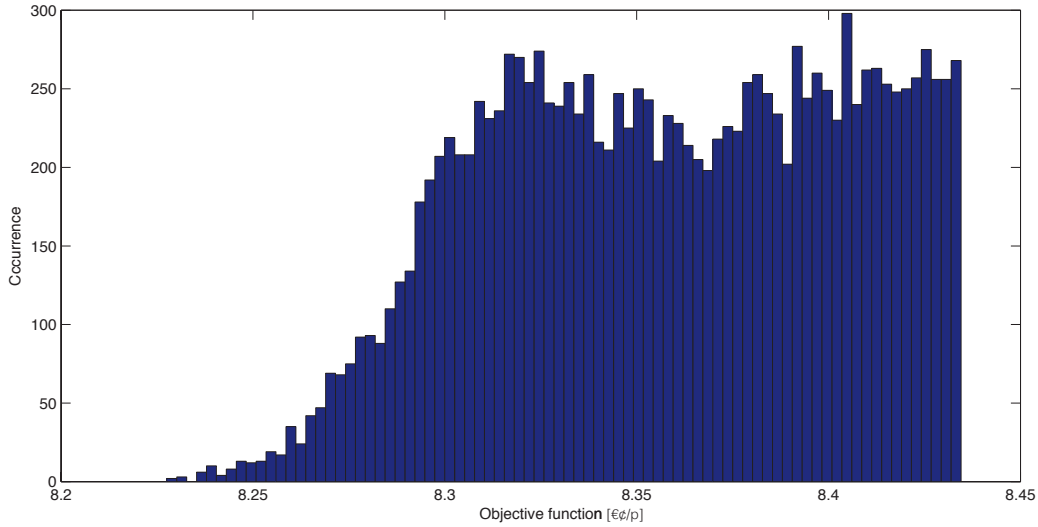


Figure 2.11: Objective function histogram plot

Chapter 3

Computational Methods

Three types of algorithms that can be applied to solve the setup problem (Section 1.3.2) will be proposed. Firstly, the heuristic K-Ratio is introduced, it consists in enumerating the whole solution space, reducing it with an heuristic rule and finding which combination performs better solving a knapsack problem. Secondly, the Response Surface Methodology (RSM) is explained. This methodology uses a factorial experiment or a fractional factorial design to estimate a first-order polynomial model and a second-order model to optimize.

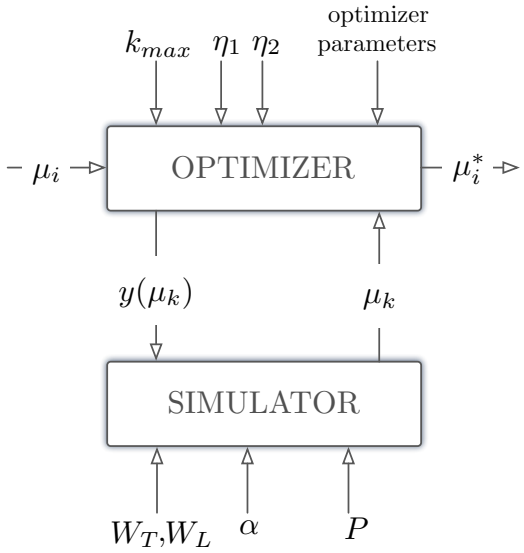


Figure 3.1: Simulator - Optimizator flow chart

Lastly, the Particle Swarm Optimization algorithm will be illustrate. Fig-

ure (3.1) explain the correlation between the simulator stated in Section 2.2 and the optimizators.

3.1 Stopping Rules.

The simulation-optimization routine requires some suitable stopping criteria to stop when a "good" solution is found. Let η_1 and η_2 be small positive numbers, k_{max} be the maximum number of the iterations, and $\widehat{\boldsymbol{\mu}}_k$ be the estimate of $\boldsymbol{\mu}_k$ at the k-th iteration. The algorithm is stopped when the first of the following three criteria is satisfied:

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

3.2 Heuristic

The number of setup configuration tackled in the several simulation is called *Cardinality* and its presented in Equation (3.1). Let H be the hoppers number and L be the level of discretization. The cardinality is defined as:

$$n_C = \frac{(H + L - 1)!}{H!(L - 1)!} \tag{3.1}$$

In particular the number of the steps between the lowest weight that can be filled up in the hoppers, and the greatest is called coarseness. As known, a discrete approach increases, drastically, the cardinality, and in this section two types of reduction rules are proposed to speed up the process without affecting the performances. Firstly, the sampling rule that consists, as the word suggests, in a random sampling of a fixed number of setup configurations. Secondly, the Hopper Volume Constraint, that is a constraint regarding the quantity delivered in each hopper. Lastly, the K-Ratio rule, a sample of the setup configurations according to their "goodness".

According to the Figure (3.2), the number of solutions n_C rose sharply from 5 hoppers and 25 levels.

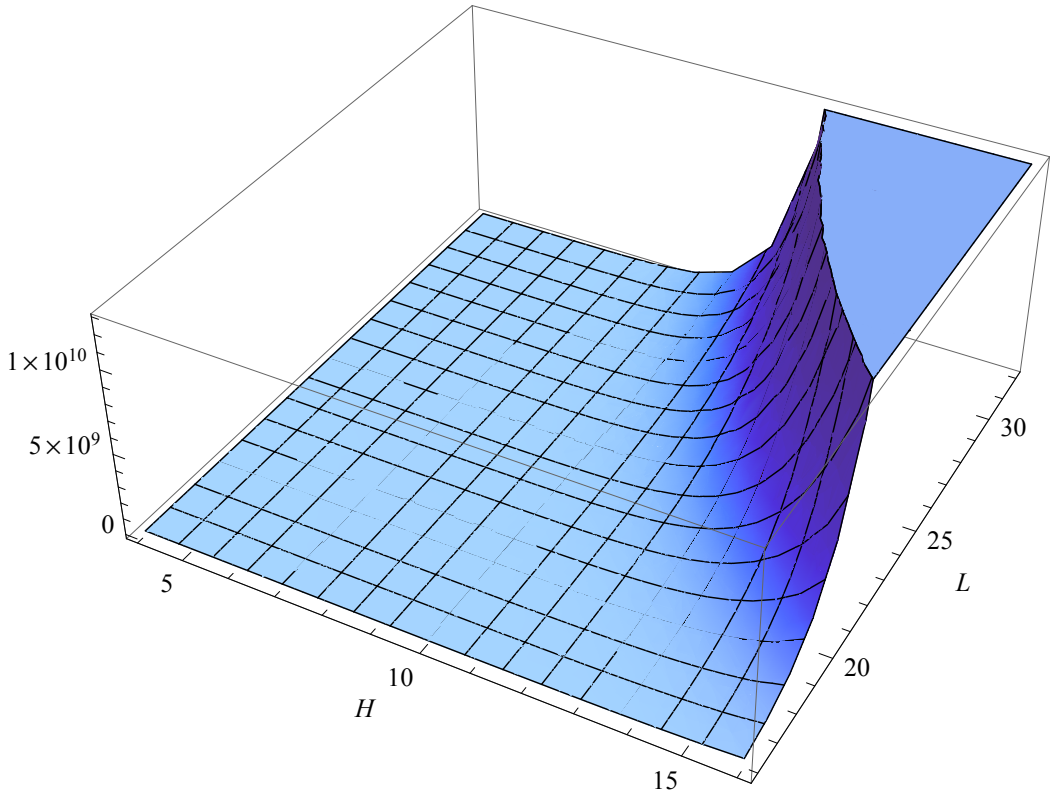


Figure 3.2: Cardinality plot

3.2.1 Sampling

The Sampling approach consists in a random sample of all solutions. The Equation (3.1) is modified as follows:

$$n_C = \frac{(H + L - 1)!}{H!(L - 1)!} \cdot S \tag{3.2}$$

where S is the sample percentage of the whole solutions space. The Sampling approach gives us a tuning handle that we can tight up or down according to the efficiency[1] required. According to the Simulation input parameters in Table (3.1), the effect of sampling is shown in Figure (3.3).

The shape of Figure (3.3) demonstrates that as the sample parameters in-

[1]Efficiency can be seen as the ratio between the objective function with reduction rules and the object function

creases, the value of the objective function decreases. A sampling parameter equal to $S = 50\%$ seems to be a good trade-off between the computational time and the sub-optimum objective function as pointed out by Table (3.2).

	Parameters					
Simulator	H	5		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	L	19		c_l	3.2	€¢/h
	H_e	2		c_f	7.34	€/p
	W_N	250	g	W_L	241	g

Table 3.1: Simulation parameters

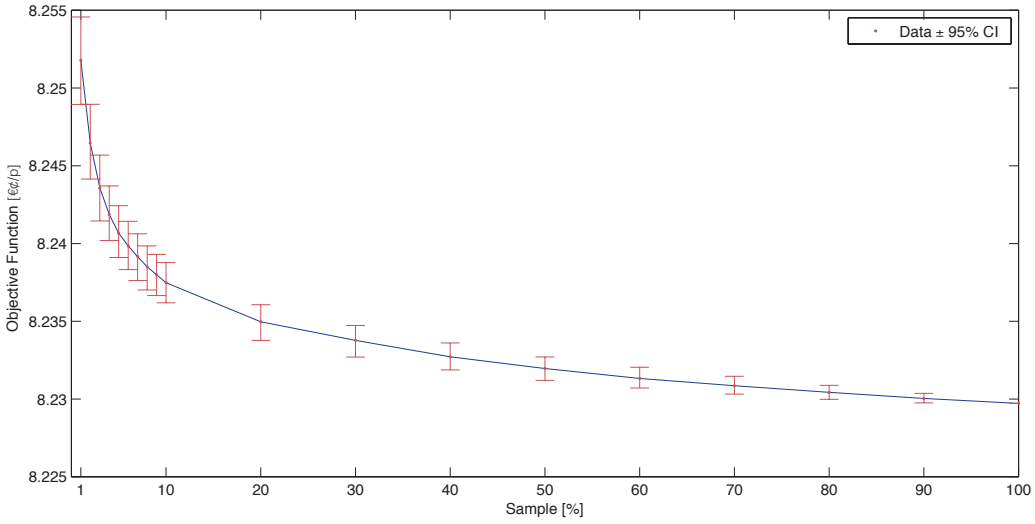


Figure 3.3: Objective function sampling effects

The data would seem to suggest that the differences between samples are small but, considering a large scale production, these differences can be meaningfull. In fact, let us consider a large production such as a 650M of packages per year, the difference between $S = 5\%$ (8.2406 €¢/p) and

	$\boldsymbol{S = 1\%}$	$\boldsymbol{S = 5\%}$	$\boldsymbol{S = 20\%}$	$\boldsymbol{S = 50\%}$	$\boldsymbol{Unsampled}$
$f(\boldsymbol{\mu}^*)$	8.2517	8.2406	8.2349	8.2319	8.2297
t_{tot}	215”	224”	234”	243”	262”

Table 3.2: Sampling effects

$S = 50\%$ (8.2319 €¢/p) is roughly 56 550 € per year. The difference between the unscaled and the $S = 50\%$ is more moderate, roughly 14 300 € per year.

3.2.2 Hopper Volume Constraint

Since the simulator allows to pinpoint both a lower bound (lb) and an upper bound (ub) to the quantity delivered in each hopper to find if these bounds need to be $lb = 0 \vee ub = W_T$ or can be different somehow, is worth. Each item weight w_j has a lower and upper bounds. The weight has to be greater than or equal to one and less than or equal to the target, otherwise the j-th hopper can not work.

$$1 \leq w_j \leq W_T \qquad j = 1, \ldots, H \tag{3.3}$$

In order to guarantee the compliance with this constraint during the simulation (at 99%), the weight of the hopper j has to be bound, according to the w_j distribution, as follows:

$$\mu_j - 3\sigma_j \leq w_j \leq \mu_j + 3\sigma_j \tag{3.4}$$

As a consequence, putting together the two constraints in Equation (3.3) and Equation (3.4), according to $\sigma_j = \alpha \cdot \mu_j$:

$$(\mu_j - 3\sigma_j \geq 1 \vee \mu_j + 3\sigma_j \leq W_T) \quad \rightarrow \quad \frac{1}{1 - 3\alpha} \leq \mu_j \leq \frac{T}{1 + 3\alpha}$$

The constraints do not affect the optimum setup configuration performance, as displayed in Table (3.3), although the computational time decreases of 75.2%. The unconstrained case, set with Table (2.4) parameters, evaluates

142 506 solutions compared to 33 649 solutions of the constrained case. All the results are performed with the Brute-force approach stated in Section 2.3.

	Unconstrained	***Constrained***
$E(W^*)$	245.9728	245.9728
$\sigma(W^*)$	4.6067	4.6067
$f(\boldsymbol{\mu^*})$	8.2239	8.2239
n_B	3000	3000
t_{tot}	$\sim$ 1582" (26' 22")	$\sim$ 392" (6' 32")
$\boldsymbol{\mu^*}$	20 30 100 110 120	20 30 100 110 120

Table 3.3: Hopper Volume Constraint Effect

Thanks to the constraints, the hoppers weight is bound from 10g to 190 g instead of 0 - 250 g (see Fig. (3.6) for a graphical example).

3.2.3 K-Ratio

The K-Ratio, unlike Sampling, uses the *problem parameters* to evaluate which are the best solutions in order to keep them and discard the rest. Empirically, it has been discovered that the ratio between the average of hopper weights and the target weight has an interesting pattern, that is called K-Ratio.

$$\text{K-Ratio} = \frac{\sum_{j=1}^{H} w_j x_j}{H} \cdot \frac{1}{W_N} \tag{3.5}$$

where

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

and w_j is quantity delivered in each pool hopper, it can be modeled as a random variable normally distributed with mean μ_j and standard deviation σ_j. The best setup configurations lay in the neighborhoods of a K-Ratio value of 0.3. A fair range to keep all "good" solutions is $[0.2, 0.5]$. Unfortunetly, very wide ranges do not pay back a massive time saving. Furthermore, the K

Ratio has a logical meaning: Considering $H \cdot W_T$ as the maximum capacity of the MHW machine[2], the K Ratio is a *capacity ratio*. Hence, all "good" solutions need to lay inside a capacity range, where the lower limit is the minimum weight that needs to be filled in all hoppers, and the upper limit is the maximum weight. All the solutions are sorted in a descending order according to the objective function value and plotted versus the K-Ratio to better understand where and how these reduction rules work (see Fig. (3.4)).

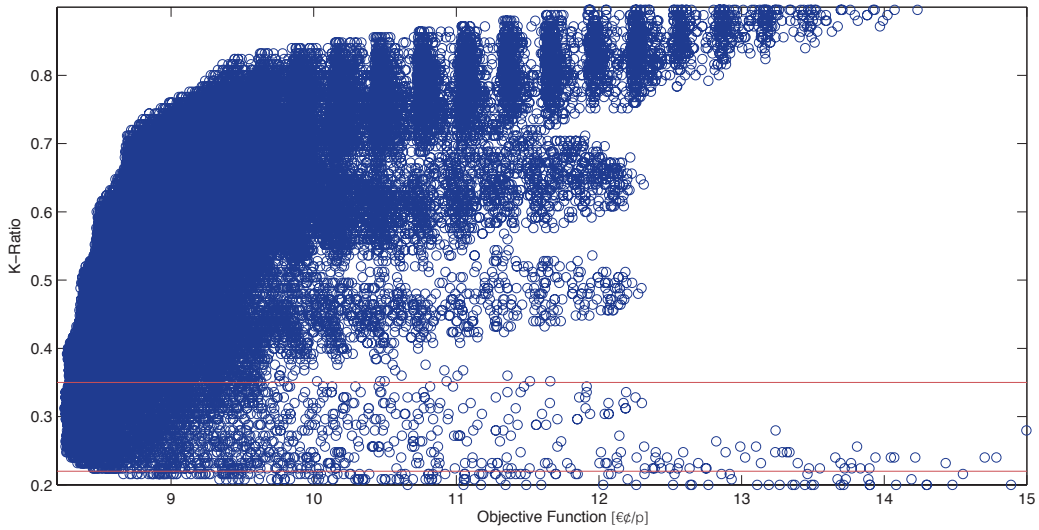


Figure 3.4: K-Ratio work zone

There is a clearly defined pattern in the Figure (3.4), and this can be mean that the K-Ratio shows two main peaks around 0.3 and 0.25. The Figure (3.5) reveals that the setup configurations with the lowest objective function values are those inside the K-Ratio bandwitch ($0.25 \rightarrow 0.3$). The number of solutions has been decreased of 92.89% without affects the two main peaks. The implication is a 87.2% computational time reduction. The results of the present study are obtained miming a 5 hoppers (H) MHW machine with 25 levels (L). This machine setup was chosen to keep the computational time short enough to compute several iteration of the same algorithm. Therefore, the total simulation time (t_{tot}) for a $H = 5, L = 25$ MHW machine, has decrease from 1589 to 204 seconds (see Table (3.4)).

[2]It is pointless fill up the hopper with more material than that one can be stored in the package since this will occur in a machine block.

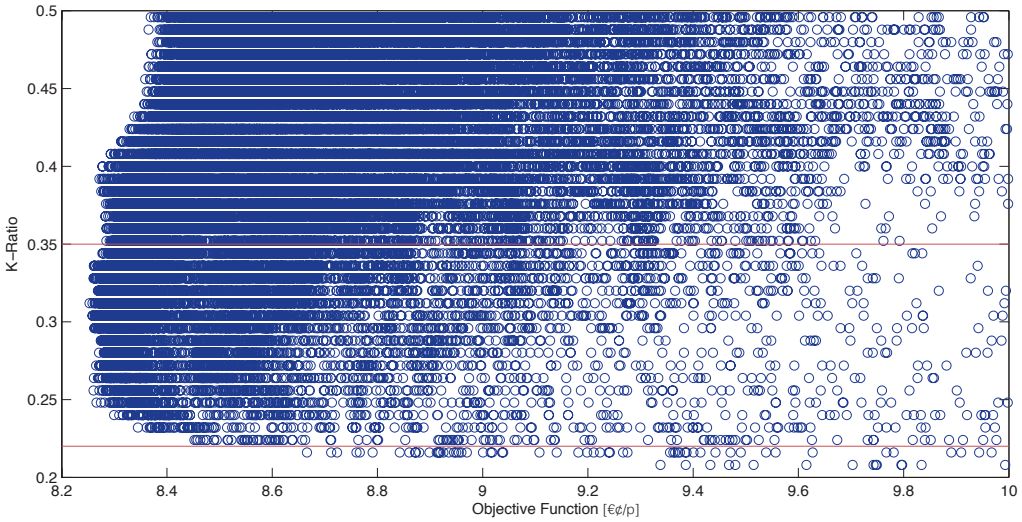


Figure 3.5: K-Ratio work zone - Rescaled axis

The average weight $E(W^*)$, the standard deviation $\sigma(W^*)$ and the objective function value $f(\boldsymbol{\mu}^*)$ of the discharged packages for the optimal setup configuration $\boldsymbol{\mu}^*$ do not change. The results show that the K-Ratio rule does not affect the best solutions but omits, from the brute-force evaluation, those setup configurations that are undoubtedly sub-optimal.

	Baseline	***K − Ratio***	***K − Ratio + HVC***
$E(W^*)$	245.8520	245.8520	245.8520
$\sigma(W^*)$	4.9485	4.9485	4.9485
$f(\boldsymbol{\mu}^*)$	8.2202	8.2202	8.2202
n_B	3000	3000	3000
t_{tot}	$\sim$ 1589” (26’ 29”)	$\sim$ 204” (1’ 48”)	$\sim$ 121” (1’ 21”)
$\boldsymbol{\mu}^*$	20 30 110 110 110	20 30 110 110 110	20 30 110 110 110

Table 3.4: K-Ratio effect

Using the K-Ratio approach paired with the Hopper Volume Constraint (see Section 3.2.2) can discard even more sub-optimal setup configurations since some solutions inside the K-Ratio bandwidth do not respect the Volume

constraint. The HVC reduction is far more restrained than the K-ratio rule but, the Table (3.4) demonstrates that excluding those sub-optimal solution can lead to a 41% computational time reduction. Figure (3.6), obtained by evaluating all the solutions with a Brufe-force aproach and sorting them by their objective function value and K-Ratio, demonstrates that the setup configurations that violate the hopper volume constraint (red dots) are unfeasible. Because those configurations are always outclassed by those that respect the HVC given that the hull of red dots is shifted to the right respect the hull of the blu circlutes. In other words, the setup configurations that violate the HVC constraint are always outperformed by those that respect it.

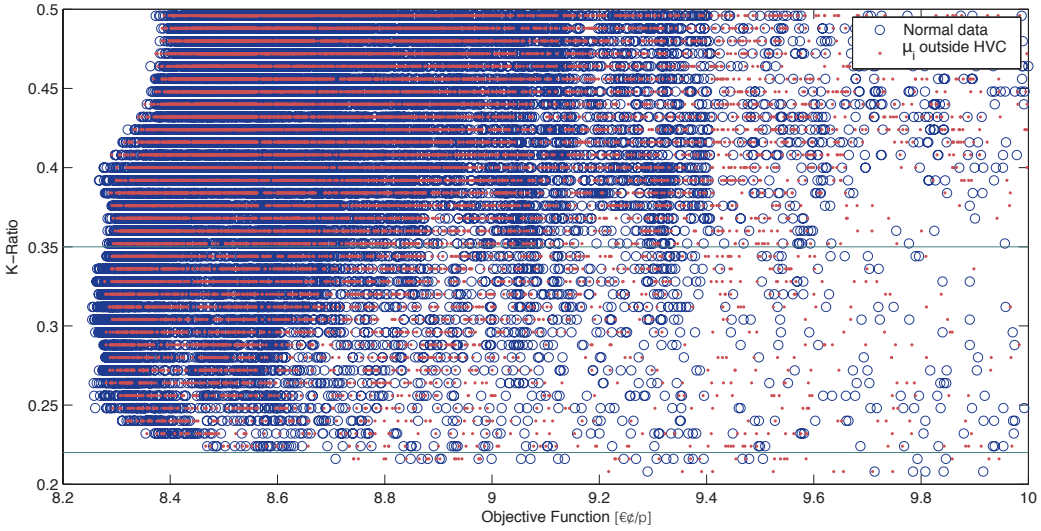


Figure 3.6: K-Ratio *&* Hopper Volume Constraint

3.3 The Response Surface Methodology

The Response Surface Methodology (RSM) is an optimization tool that was introduced in the fifties by Box & Wilson [1951] and examined in depth in [Box, Hunter & Hunter 1978, Box & Draper 1987, Myers, Montgomery & Anderson-Cook 2009, Khuri & Cornell 1996, Del Castillo 2007]. RSM is a collection of mathematical and statistical techniques that are useful to optimize a stochastic process. Usually, the form of the relationship between

the response and the factors is not known exactly. RSM, using designed experiments, estimates this relationship by means of simpler approximating functions that are valid on a small sub-region of the domain D. By moving the operating conditions of a process using a sequence of experimental designs, process improvement is achieved.

Let $\boldsymbol{\mu} = (\mu_1, ..., \mu_p)$ be the p-dimensional vector of the controllable factors and $y(\boldsymbol{\mu})$ be the response variable, called objective function, to minimize (without loss of generality). The response variable depends on the p factors by the relation $y(\boldsymbol{\mu}) = \eta(\boldsymbol{\mu}) + \varepsilon$, where $\eta(\boldsymbol{\mu}) = f(\mu_1, ..., \mu_p)$ is the true response variable, unobservable directly because of the process noise ε. The function f is unknown, directly unaccessible to us and its expected value should be approximated. The process noise ε is usually assumed distributed with mean zero and constant variance σ_ε^2. Usually, the $\boldsymbol{\mu}$ factors are coded, via the orthogonal convention, into a unitless value ($x_1..x_p$), typically on the -1 to $+1$ range.

Usually, an RSM procedure, depicted in Figure (3.7), comprises two phases. In the first phase, a first-order model is used to approximated the objective function in the region of interest (ROI) using an opportune design of experiment

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \varepsilon \tag{3.6}$$

where the constant β_0 and the linear coefficients β_i are estimated (b_0, b_i) by using ordinary least squares (OLS) applied to the data and $\varepsilon \sim N(0, \sigma_\varepsilon^2)$. When the first order model adequately describes the response behavior, a line search algorithm is applied to find a new ROI. The line search ends 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 model shows a significant lack-of-fit. Thus, the RSM moves to the second phase and additional experiments are conducted to obtain a more precise estimate of the response function. The objective function is now

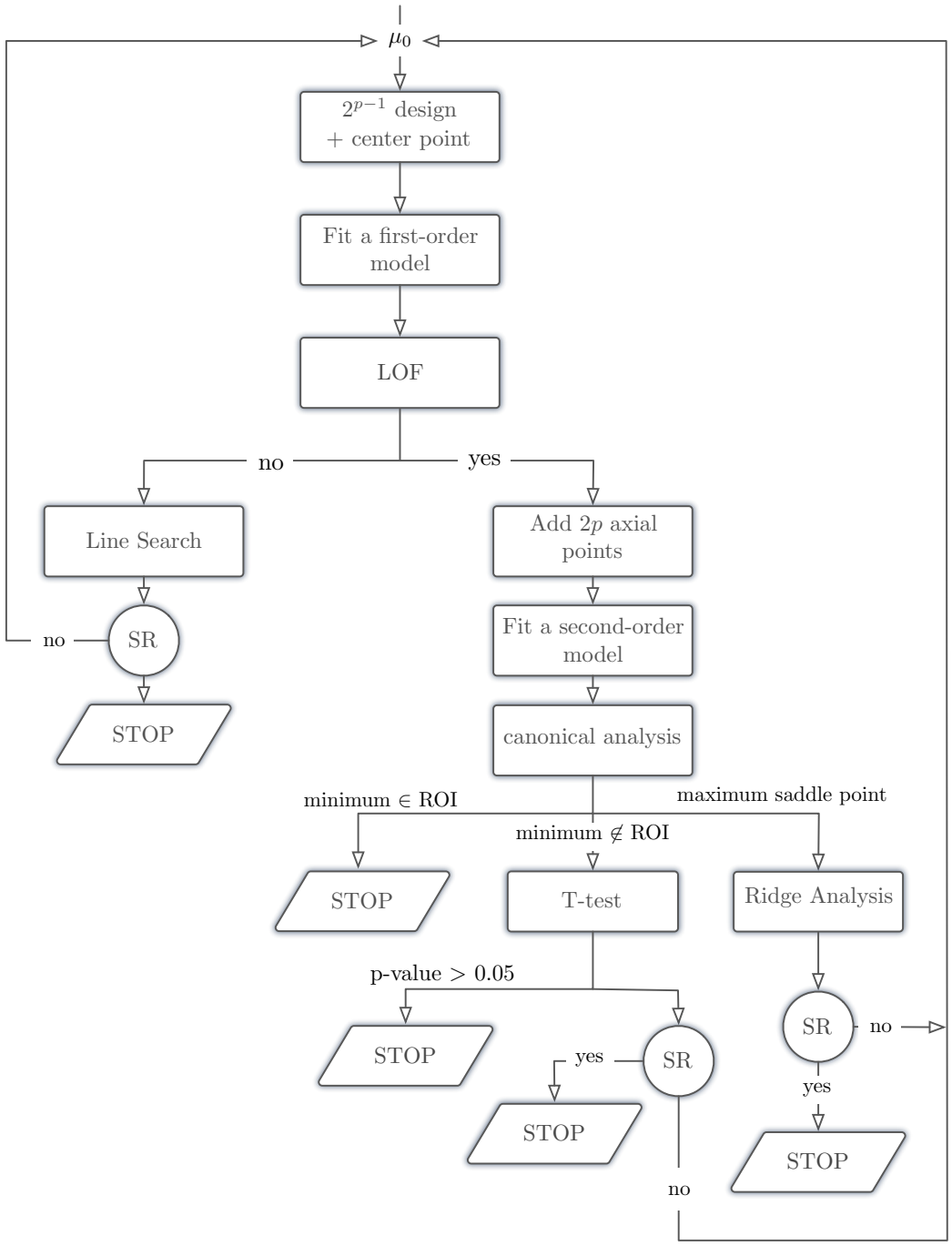


Figure 3.7: Flowchart of RSM procedure

approximated by a second-order polynomial:

$$y = \beta_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \beta_{ii} x_i^2 + \sum_{i=1}^{p} \sum_{j>i} \beta_{ij} x_i x_j + \varepsilon \tag{3.7}$$

where the regression coefficients are again determined by using OLS applied to the experimental data. Then, a canonical analysis is used to find the optimum point, if it exists.

In RSM literature, it is often proposed to end the algorithm when, after fitting the second order polynomial, a minimum point inside the ROI is found. Differently, in this thesis, some stopping criteria (see section 3.1) are added to end the RSM procedure if either a minimum point inside the ROI is found, if the estimated optimal response value or the the estimated optimal parameters do not improve sufficiently from one algorithm iteration to the next one, or if a fixed maximum number of algorithm iteration (k_{max}) have been performed. The diagram in Figure () depicts the RSM procedure, the stopping criteria are checked each time the "SR" circle is shown in the diagram.

The RSM is very popular in the manufacturing, chemical, biological sectors to improve process. Usually in this fields, the RSM is performed in a manual setting. Instead, in the area of computer experiment, where this thesis focuses the attention, an automated RSM procedure is necessary. Smith [1976] was the first to propose an automated RSM program, without explaining the choices that he made about the RSM procedure. Then, Joshi, Sherali & Tew [1998] described an enhanced algorithm for RSM comparing it with a standard RSM algorithm, unfortunately they did not explicate many details. Eventually, Neddermeijer, van Oortmarssen, Piersma & Dekker [2000] ([Nicolai, Dekker, Piersma & van Oortmarssen 2004, Nicolai & Dekker 2009]) proposed a detailed framework for a RSM algorithm in the simulation-optimization field. In particular, when an automated RSM procedure is required, there are number of settings that need to be implemented. Thus, the details of the procedure are now explained according to the step of the automated framework proposed by Neddermeijer et al. [2000].

3.3.1 The linear model

The objective function is approximated in the current ROI by a first-order model (Eq.3.6). To this end, the objective function is evaluated in the points of an experimental design. In literature, for instance see [Myers et al. 2009], many designs are available, such as fractional or full factorial, two-level or three levels designs. These designs can be augmented by center points that allow to test for curvature.

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 ($H = 8$ hoppers). This design allows 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 can easily be augmented to derive a second-order design.

Check the adequacy of the first-order model. Usually, a lack of fit (LOF) test is performed to check if the estimated first-order model adequately describes the response behavior in the current ROI. The LOF test requires that the design is not saturated, that is the total number of experiments should be larger than the number of regression coefficients, and that the center point is conveniently replicated.

What do to when the first-order model is adequate? If the first-order model does not exhibit lack of fit due to second order terms, a steepest descent procedure is applied to find a point of improved response. Then, this point is used as the center point of the design that will be used to fit a new first-order model.

How to solve first-order model inadequacy? If the first-order approximation is not accepted there is some evidence that the quadratic effects (i.e. curvature or interactions between factors) are significant. Consequently, the experimental design needs to be augmented to allow to fit a second order model which includes such terms.

3.3.2 The steepest descent

The fitted first-order model is adequate, thus it is used to define the direction of maximum improvement of the response. Along this direction, a line search is performed from the center point of the current ROI to find a point of improved response. As soon as a boundary of the domain D is crossed, the line search is continued along the projection of the search direction on this boundary ([Smith 1976]). Once the point of improved response is reached, a new first-order experiment should be performed to asses lack of fit.

Two main decisions have to be made: the steepest descent direction and the length of the step to move along this direction, and the stopping rule to end the line search when no further improvement is observed.

How to select the steepest descent direction and the step size? Usually, the steepest descent direction is given by $-\nabla\widehat{y}$. The step size Δ_i, according to Del Castillo [2007], is equal to the distance from the center point to the point of intersection of the steepest descent direction and the sphere given by $\sum_{i=1}^{p}\Delta_i^2 = 1$. An equivalent and simple approach [Myers et al. 2009] is to choose the most important factor by determining j such that $j = \arg\max_{i=1,...,p}|b_i|$. Then, the step size is set equal to $\Delta_i = \frac{-b_i}{|b_j|},\ i = 1, ..., p$. If the input variable are unitless coded, these two approaches are equivalent. Instead, Kleijnen, Den Hertog & Angün [2004] provided a suggestion not only of what step size to use, but also on which direction the search has to be conducted. This technique is useful when the design of experiment is not orthogonal.

How to stop the line search? Different heuristic stopping rules have been proposed in literature. The "classical" rule stops the algorithm when an observed value of the objective function is higher than the preceding observation. But a rise in the response may be due to "noise", not to a real increase of the mean response. As a consequences, the 2-in-a-row rule (i.e. 2 consecutive response rises) and the 3-in-a-row one (i.e. 3 consecutive response rises) have been introduced as a "classical" rule improvement. Del Castillo [1997] demonstrated that the "classical" rule and its improvements perform very poorly, stopping far before the optimum, especially when the noise level

was high and the optimum was far away from the starting point. Thus, three formal sequential stopping rules have also been proposed in the frequentist literature. Firstly, Myers & Khuri [1979] proposed a rule that involves a sequential hypothesis test to check if the response is still decreasing whenever a rise in the response is observed. However this rule requires a preliminary guess of the step number required to reach the optimum and, as expected, the procedure is rather sensitive to this parameter. Secondly, Del Castillo [1997] proposed the Recursive Parabolic Rule (RPR). This rule fits a second order model in the direction of steepest descent: $y(t) = \theta_0 + \theta_1 t + \theta_2 t^2 + \varepsilon$ where t is the search step counter, the θ_i are the model coefficients and ε is the random error in the observations. The rule estimates the intercept θ_0 and the first order coefficient θ_1 from the previous fitted first order model. Instead, the curvature parameter θ_2 is re-estimated sequentially at each step t using recursive least squares. The search is stopped when there is enough evidence to reject the hypothesis that the first derivative is strictly positive, since this assure that the average response is starting to increase. Lastly, Miró-Quesada & Del Castillo [2004] proposed the Enhanced Recursive Parabolic Rule, as an improvement of the previous one. In fact, the RPR becomes sensitive to non-quadratic behavior, consequently the intercept and the first order coefficient are now recursively updated in order to increase the robustness against non-quadratic behavior. In those days, in the bayesian word, Gilmour & Mead [1995] presented a stopping rule based on the estimate of the expected gain, i.e. the difference between the expected response at the i-th step and the response at the predicted i-th step. The suggested to end the experimentation when the expected gain mean is small or falls below a specified value.

In this thesis, the steepest descend direction and the line search step size are selected according to the [Myers et al. 2009] approach. Regarding the stopping rule, despite the possible drawbacks, the 2-in-a-row rule is picked for its simplicity of implementation.

3.3.3 The quadratic model

The objective function is now approximated in the current ROI by a second-order model (Eq.3.7). To this end, experimental design strategies require at least three distinct level in each coordinate. A popular second-order design

is the central composite design (CCD), proposed by Box & Wilson [1951] in their seminal paper. This design can be easily constructed by adding the $2p$ axial points to the first-order design. A useful recommendation is to make the CCD rotatable, so that the predicted response variance remains constant at all the points which are equidistant to the center point of the current ROI. Obvious alternatives to the CCD are 3^p designs, these designs are not use in practise because of the excessive number of experiments required, and the 3-level fractional factorials. A popular class of 3-level design is the Box-Behnken design [Box et al. 1978].

In this thesis, a CCD design is used to fit a second-order model. The CCD is composed of 2^{p-1} fractional factorial (used to fit the first-order model), that has at least a resolution V which allows us to fit all second order interactions without any aliasing of other 2-factor interactions and $2p$ axial points that allow to fit the quadratic terms.

The fitted second-order model can be written as:

$$y(\boldsymbol{x}) = \beta_0 + \boldsymbol{b'x} + \boldsymbol{x'Bx} \tag{3.8}$$

where $\boldsymbol{x}$ is a $p \times 1$ vector of the controllable factors, $\boldsymbol{b}$ is a $p \times 1$ vector that contains the main effect parameter estimates and $\boldsymbol{B}$ is a symmetric matrix that contains the b_{ii} parameters in the diagonals and $1/2b_{ij}$ in the (i, j) off-diagonal points.

Then the canonical analysis is performed to determine the location and the nature of the stationary point $\boldsymbol{x}_0$, that is determined by:

$$\boldsymbol{x_0} = \frac{1}{2}\boldsymbol{B^{-1}b} \tag{3.9}$$

Let $\boldsymbol{E}$ be the matrix of normalized eigenvectors of $\boldsymbol{B}$ and let $\lambda_1, ..., \lambda_p$ be the eigenvalues of $\boldsymbol{B}$. If all the eigenvalues λ_i are negative (positive), the stationary point $\boldsymbol{x_0}$ is a *minimum* (*maximum*). Instead if the eigenvalues have mixed sign, then $\boldsymbol{x_0}$ is a *saddle point*.

What to do if $\boldsymbol{x_0}$ is a *minimum* inside the ROI? The RSM procedure is ended and the stationary point $\boldsymbol{x_0}$ is the minimum point that optimize the objective function. Substituting the Eq.(3.9) into the fitted

second-order model (Eq.3.8), the predicted response at the stationary point is:

$$\widehat{y}(\boldsymbol{x_0}) = \widehat{\beta}_0 + \frac{1}{2}\boldsymbol{x_0'b}$$

What to do if $\boldsymbol{x_0}$ is a *minimum* outside the ROI? A sample t-test is performed to compare the response mean of the center point of the ROI and the response mean of the minimum point found outside the ROI. To perform the test, replications of the minimum point response are required and, in this thesis, the replication number is equal to center point number (3). If the test is refused, that is the response means of the two points are different, a new first-order design centered in this minimum point is performed. Instead, if there is no difference between the response means of the two points, the RSM procedure is ended and this point is returned as the minimum point that optimize the objective function.

What to do if $\boldsymbol{x_0}$ is a *maximum* or a *saddle point*? A Ridge Analysis (see [Myers et al. 2009]) is performed to look for a point of minimum response inside the current ROI since it is not correct to extrapolate the second-order model outside the current ROI. Once a minimum point is found, a new experiment is performed and a first-order model is fitted.

3.3.4 Parameters tuning

The key aspect of RSM algorithm is to find the optimal number of initial point, since each of these points represent a Multihead Weigher setup configurations. Figure (4.1) shows the Objective function values and the errorbars in order to find the optimal number of these points avoinding large computational time giving no improvements. Every algorithm setup, of the below chart, was iterate 100 times to better estimate the objective function value.

The statistics seem to show that the number of Initial Points affects the Objective function value but, having a closer look, it is straightforward that a gap of 1×10^{-5}€/p can not give any benefits even with a massive production. Taking into account the error bars[3] of Figure (4.1), we can surmise that the

[3]Using Bootstrap, the errorbars are obtained by resample the N data. Bootstrap uses

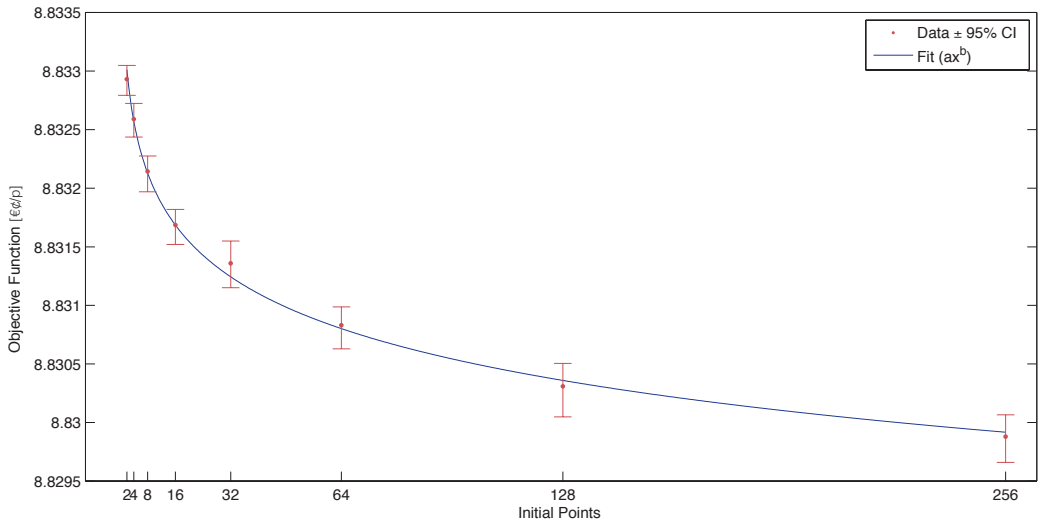


Figure 3.8: RSM Initial Points

optimal setup for the RSM approach is from 2 to 8 initial points. Moreover, all these possibilities garantee small computational times.

3.4 The Particle Swarm Optimization

Soft computing is an approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision. In an attempt to find out reasonably useful solutions. Soft computing techniques such as NN, GA, SA, ACO, and PSO have received a lot of attention of researchers due to their potentials to deal with highly nonlinear, multidimensional, and ill-behaved complex engineering problems. A brief overview of various soft computing techniques is presented here.

Neural Networks

Neural networks are systems that can acquire, store, and utilize knowledge gained from experience. An artificial neural network (ANN) is capable of

N_{boot} data sets each containing N points obtained by random (Monte Carlo) sampling of the original set of N points.

learning from an experimental data set to describe the nonlinear and interaction effects with great success. It consists of an input layer used to present data to the network, output layer to produce ANN's response, and one or more hidden layers in between. The input and output layers are exposed to the environment and hidden layers do not have any contact with the environment. ANNs are characterized by their topology, weight vectors, and activation function that are used in hidden and output layers of the network. A neural network is trained with a number of data and tested with other set of data to arrive at an optimum topology and weights. Once trained, the neural networks can be used for prediction. A multilayer perceptron (MLP) is a feedforward neural network with one or more hidden layers. A feedforward neural network has sequence of layers consisting of a number of neurons in each layer. The output of one layer becomes input to neurons in the succeeding layer.

Genetic Algorithm

GA mimics the process of natural evolution by incorporating the "survival of the fittest" philosophy [GE 1989]. In GA, a point in search space is represented by binary or decimal numbers, known as string or chromosome. Each chromosome is assigned a fitness value that indicates how closely it satisfies the desired objective. A set of chromosomes is called population. A population is operated by three fundamental operations, viz., reproduction (to replace the population with large number of good strings having high fitness values), crossover (for producing new chromosomes by combining the various pairs of chromosomes in the population), and mutation (for slight random modification of chromosomes). A sequence of these operations constitute one generation. The process repeats till the system converges to the required accuracy after many generations. The genetic algorithms have been found very powerful in finding out the global minima. Further, these algorithms do not require the derivatives of the objectives and constraints functions.

Simulated Annealing

SA is motivated by an analogy to annealing in solids. The idea of SA comes from a paper published by Metropolis [Metropolis, Rosenbluth, Rosenbluth,

Teller & Teller 1953]. The algorithm in this paper simulated the cooling of material in a heat bath. This is a process known as annealing. If you heat a solid past melting point and then cool it, the structural properties of the solid depend on the rate of cooling. If the liquid is cooled slowly enough, large crystals will be formed. However, if the liquid is cooled quickly (quenched) the crystals will contain imperfections. Metropolis's algorithm simulated the material as a system of particles. The algorithm simulates the cooling process by gradually lowering the temperature of the system until it converges to a steady, frozen state.

In 1982, Kirkpatrick et al [Kirkpatrick, Gelatt & Vecchi 1983] took the idea of the Metropolis algorithm and applied it to optimisation problems. The idea is to use simulated annealing to search for feasible solutions and converge to an optimal solution.

Ant Colony Optimization

ACO is the result of research on computational intelligence approaches to combinatorial optimization originally conducted by Dr. Marco Dorigo, in collaboration with Alberto Colorni and Vittorio Maniezzo [Dorigo, Birattari & Stutzle 2006]. The fundamental approach underlying ACO is an iterative process in which a population of simple agents repeatedly construct candidate solutions; this construction process is probabilistically guided by heuristic information on the given problem instance as well as by a shared memory containing experience gathered by the ants in previous iteration. ACO has been applied to a broad range of hard combinatorial problems. Problems are defined in terms of components and states, which are sequences of components. Ant Colony Optimization incrementally generates solutions paths in the space of such components, adding new components to a state.

Particle Swarm Optimization

PSO is a population-based stochastic optimization technique developed by Kennedy and Eberhart in 1995 and is inspired by the social behavior of bird flocking or fish schooling. Sometimes it is related to the Evolutionary Computation (EC) techniques, basically with Genetic Algorithms (GA) and Evolutionary Strategies (ES), but there are significant differences with those

techniques.
PSO is a metaheuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods. PSO can therefore also be used on optimization problems that are partially irregular, noisy, change over time, or high density of optimum. As the MHW machine problem present a large class of optimum values, the PSO will be used to search the global optimum in the continius space.

The PSO algorithm is population-based: a set of potential solutions evolves to approach a convenient solution (or set of solutions) for a problem. Being an optimization method, the aim is finding the global optimum of a real-valued function (fitness function) defined in a given space (search space). The social metaphor that led to this algorithm can be summarized as follows: the individuals that are part of a society hold an opinion that is part of a "belief space" (the search space) shared by every possible individual. Individuals may modify this "opinion state" based on three factors:

- The knowledge of the environment (its fitness value)
- The individual's previous history of states (its memory)
- The previous history of states of the individual's neighborhood

An individual's neighborhood may be defined in several ways, configuring somehow the "social network" of the individual. Several neighborhood topologies exist (full, ring, star, etc.) depending on whether an individual interacts with all, some, or only one of the rest of the population.
Following certain rules of interaction, the individuals in the population adapt their scheme of belief to the ones that are more successful among their social network. Over the time, a culture arises, in which the individuals hold opinions that are closely related.

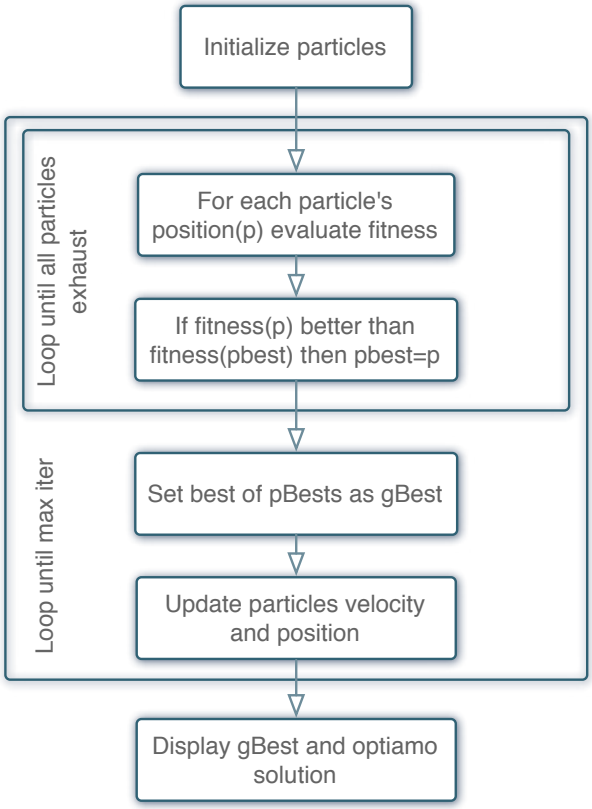


Figure 3.9: PSO Flow Chart

In the PSO algorithm each individual is called a "particle", and is subject to a movement in a multidimensional space that represents the belief space. Particles have memory, thus retaining part of their previous state. There is no restriction for particles to share the same point in belief space, but in any case their individuality is preserved. Each particle's movement is the composition of an initial random velocity and two randomly weighted influences: individuality, the tendency to return to the particle's best previous position, and sociality, the tendency to move towards the neighborhood's best previous position.
PSO is simple to implement and needs less parameters to adjust. In many cases, it has been reported to be more efficient than GA.

3.4.1 Problem Formulation

The goal of any optimization problem is to maximize or minimize an objective function $f(\vec{x})$ where x is the decision vector consisting of n dimensions or decision variables consisting in real numbers. Since maximization of any function $f(\vec{x})$ is equivalent to minimization of $-f(\vec{x})$ without loss of generality, the literature generally focuses on minimization. Solution $\vec{x}^*$ is a global minimizer of $f(\vec{x})$ if and only if $f(\vec{x}^*) \leq f(\vec{x})$ for all $\vec{x}$ in the domain of $f(\vec{x})$. The uncostrained minimization problem of consideration here can be formulated as

$$\begin{aligned} &\min \quad f(\vec{x}) \\ &\text{where} \quad f : R^n \to R \end{aligned} \tag{3.10}$$

The basic idea of particles searching individually while communicating with each other concerning the global best in order to produce a more capable collective search applies to all forms of PSO from the originally conceived algorithm through the more capable models available today.
Particle swarm, as originally published [Kennedy & Eberhart 1995], consisted of a swarm of particles each moving or flying through the search space according to velocity update equation

$$\begin{aligned} \vec{v}_i(k+1) = \vec{v}_i(k) + c_1 \vec{r}_{1_i}(k) \cdot (\vec{p}_i(k) - \vec{x}_i(k)) + \\ + c_2 \vec{r}_{2_i}(k) \cdot (\vec{g}(k) - \vec{x}_i(k)) \end{aligned} \tag{3.11}$$

where

$\vec{v}_i(k)$ is the velocity vector of particle i at iteration k,

$\vec{x}_i(k)$ is the position vector of particle i at iteration k,

$\vec{p}_i(k)$ is the n-dimensional personal best of particle i found from initialization through iteration k,

$\vec{g}(k)$ is the n-dimensional global best of the swarm found from initialization through iteration k,

c_1 is the cognitive acceleration coefficient so named for its term's use of the personal best, which can be thought of as a cognitive process whereby a particle remembers the best location it has encountered and tends to return to that state,

c_2 is the social acceleration coefficient so named for its term's use of the global best which attracts all particle simulation social communication,

$\vec{r}_{1_i}(k)$ and $\vec{r}_{2_i}(k)$ are vectors of pseudo.random numbers with components selected from uniform distribution U(0,1) at iteration k, and $\cdot$ is the Hadamard operator representing element-wise multiplication.

The farther a particle is from its personal best, the larger $||\vec{p}_i - \vec{x}_i||$ is and the stronger the acceleration toward that point is expected to be. Notice that if a particular dimension of the current position is greater than the same dimension of the personal best, the acceleration on that dimension is negative, which means that the particle is pulled back toward that location on that dimension. Of course, this implies that when the personal best lies ahead of the current position, the particle will accelerate in the positive direction toward the personal best so that each particle is always pulled toward its personal best on each dimension.

Similarly, the farther a particle is from its global best, the larger $||\vec{g}_i - \vec{x}_i||$ is and the stronger the acceleration is toward that point. The social and cognitive acceleration coefficient, c_1 and c_2 determine the respective strenghts of those pulls and relative importances of each best.
When each dimension of the social and cognitive terms is multiplied by a

random number, the acceleration is not necessarily directed straight toward the best. Were the same random number used on all dimension, each pull will be straight toward its best. Either way, particles are accelerated in two different directions at once so that they do not actually accelerate straight toward either best.

At each iteration, the previous velocity is reduced by the inertia weight and altered by both accelerations in order to produce the velocity of the next iteration.
Treating each iteration as a unit time step, a position update equation can be stated

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \qquad \text{for } i = 1, 2, \ldots, s \tag{3.12}$$

Static Inertia Weight

There was a weakness inherent in velocity update Equation (3.11) that was fixed by the introduction of an inertia weight. As prroved in [Evers 2009], by mathematical induction, it can be seen that

$$\begin{aligned} \vec{v}_i(k+1) = \vec{v}_i(0) + c_1 \sum_{a=1}^{k} \vec{r}_{1_i}(a) \cdot [\vec{a}_i(k) - \vec{x}_i(a)] + \\ + c_2 \sum_{a=0}^{k} \vec{r}_{2_i}(a) \cdot [\vec{g}(a) - \vec{x}_i(a)] \end{aligned} \tag{3.13}$$

Because the personal bests and global best can only improve over time, $\vec{v}_i(k+1)$ should rely more heavily upon recent bests than upon early values. Yet Equation (3.13) shows that early information in $\vec{p}_i(a \ll k)$ and $\vec{g}_i(a \ll k)$ is given just as much opportunity to affect $\vec{v}_i(k+1)$ as is the higher quality information of later interations since the information fo all iterations is summed without any weighting scheme by which to increase the relative importance of the higher quality information of later iterations. This problem is remedied by introduction and intertia weight [Eberhart & Shi 2000, Shi & Eberhart 1998], $\omega \in (0, 1)$, into velocity update Equation (3.11) according to

$$\vec{v}_i(k+1) = \omega^{k+1}\vec{v}_i(k) + c_1 \sum_{a=1}^{k} \omega^{k-a}\vec{r}_{1_i}(a) \cdot [\vec{p}_i(a) - \vec{x}_i(a)) + \\ + c_2 \sum_{a=0}^{k} \omega^{k-a}\vec{r}_{2_i}(a) \cdot [\vec{g}(a) - \vec{x}_i(a)] \tag{3.14}$$

So long as the inertia weight has a magnitude less than one, (see Equation (3.14)) shows that past personal bests are expected to have less effect on a particle's velocity at iteration $k+1$ than more recent personal bests due to the effect of multiplication at each iteration by the inertia weight, ω. This makes sense conceptually since recent bests, both global and personal, are expected to be of higher quality than past bests.

Velocity Clamping

Eberhart and Kennedy introduced velocity clamping, which helps particles take reasonably sized steps in order to comb through the search space rather than bouncing about excessively [Eberhart & Kennedy 1995].
Velocity clamping is done by first calculating the range of the search space on each dimension, which is done by subtracting the lower bound from the upper bound.

For example, if each dimension of the search space is defined by lower and upper bounds $[100, 100]$, the range of the search space is 200 per dimension. Velocities are then clamped to a percentage of that range according to

$$v_j^{max} = \lambda \cdot range(\Omega), \qquad \lambda \in (0, 1] \tag{3.15}$$

where $range(\Omega) = x_j^U - x_j^L$, search space $\Omega = [x_1^L, x_1^U] \times [x_2^L, x_2^U] \times \ldots \times [x_n^L, x_n^U] \subset R^n$ and x_j^L and x_j^U are, respectivley, the lower and upper bounds of the search space.

3.4.2 Initializations

For a particle swarm of size s, each particle, $\vec{x}_i = [x_{i_1}, x_{i_2}, \ldots, x_{i_n}]$, represents a potential solution to the optimization problem where particles are indexed as $i = 1, 2, \ldots, s$. The swarms is initialized by randomizing partiles' positions

about the center, $\overrightarrow{\text{center}}$, of the search space, Ω, using random numbers drawn from a uniform distribution so that no portion of the aearch space is preferred over any other as would results from random numbers being drawn from a normal distribution. This can be done according to Equation (3.16) below.

$$\begin{aligned} \vec{x}_i(k=0) &= \overrightarrow{\text{center}} + \vec{r}_i \cdot \overrightarrow{\text{range}}(\Omega) - \frac{1}{2} \cdot \overrightarrow{\text{range}}(\Omega) \\ \overrightarrow{center} &= [\frac{x_1^L + x_1^U}{2}, \frac{x_2^L + x_2^U}{2}, \ldots, \frac{x_n^L + x_n^U}{2}] \end{aligned} \tag{3.16}$$

where $\vec{r}_i = [r_{1_i}, r_{2_i}, \ldots, r_{n_i}]$ with $r_{j_i} \sim U(0,1)$ and $\overrightarrow{\text{center}}$ is usually known in advance rather than calculated.
The personal bests are then initialized to be the same as the initial positions since there are no past positions with which to compare:

$$\vec{p}_i(k=0) = \vec{x}_i(k=0) \tag{3.17}$$

Let $P(k) = \{\vec{p}_1(k), \vec{p}_2(k), \ldots, \vec{p}_s(k)$ be the set of all personal bests at iteration k. In *Gbest PSO*, the global best is initialized and iteratively updated to be the best of all personal bests according to

$$\vec{g}_i(k) = \underset{p_i(k) \in P(k)}{\arg\min} f(\vec{p}_i(k)) \tag{3.18}$$

The value of each particle's velocity along dimension j is initially randomized to lie within $[-v_j^{max}, v_j^{max}]$ according to Equation (3.19) and subsequently clamped to lie within the same range since particles should only need to step through some maximum percentage of the search space per iteration.

$$\vec{v}_i(k) = 2\vec{r}_i \cdot \vec{v}^{max} - \vec{v}^{max} \tag{3.19}$$

3.4.3 Iterative Swarm Motion

Iteratively, particle i moves from its current position to a new position along velocity vector, $\vec{v}_i = [v_{i_1}, v_{i_2}, \ldots, v_{i_n}]$, according to position update Equation (3.12), where the rate of velocity may be thought of as being multiplied by a unit time step of one iteration.
The velocity is first calculated according to velocity update Equation (3.20)

$$\vec{v}_i(k+1) = \omega \vec{v}_i(k) + c_1 \vec{r}_{1_i}(k) \cdot (\vec{p}_i(k) - \vec{x}_i(k)) + \\ + c_2 \vec{r}_{2_i}(k) \cdot (\vec{g}(k) - \vec{x}_i(k)) \tag{3.20}$$

As the stepping process according to velocity and position update Equation (3.12) and Equation (3.20) continues, particles update their personal bests as they encounter better positions than encountered previously. Particles eventually converge, via their communication of the global best and collective movement toward it, to the one best position they have found.
The algorithm can be allowed to run either for a number of iterations expected to produce a good solution or until a user-specified criterion or threshold is reached.

In *Gbest PSO* the *global best*, $\vec{g}(k)$, is iteratively updated according to the same Equation (3.18) by which it was initialized. It is then "communicated" via shared computer memory to all particles for consideration.

Stagnation & Regrouping PSO

The swarm is said to have converged prematurely when the proposed solution approximates a local, rather than global, minimizer and when progress toward a global minimizer has ceased so that continued activity could only hope to refine a local minimizer. Stagnation is a result of premature convergence. Once particles have converged prematurely, they continue converging to within extremely close proximity of one another so that the global best and all personal bests are within one miniscule region of the search space. Since particles are continually attracted to the bests in that same small vicinity, particles stagnate as the momentum from their previous velocities vanishes. While particles are technically always moving, stagnation can be thought of as a lack of movement discernable on the large scale, from which perspective the stagnated swarm will appear as one static dot.

In order for swarm regrouping to be triggered, premature convergence has first to be detected. If none of initialized particles encounter a better solutions (i.e a new global best) over a period of time , they will continue moving closer to the unchanged global best until the all occupy pratically the same space location. It is usefull to measure how near particles are to each

other so that an effective action can be taken once they have converged to the same region. G. Evers proposes [Evers & Ben Ghalia 2009], using Van de Bergh *Maximum Swarm Radius* approach [Van Den Bergh 2002], that once premature convergence is detected the swarm be regrouped in a new search space centered at the global best as follows.

At each iteration, k, *the swarm radius*, $\delta(k)$, is taken to be the maximum Euclidean distance, in n-dimensional space, of any particle from the global best as follows.

$$\delta(k) = \max_{i \in \{1,\ldots,s\}} ||\vec{x}_i(k) - \vec{g}(k)|| \tag{3.21}$$

where $||\cdot||$ denotes the Euclidean norm. Let $\text{diam}(\Omega) = ||\text{range}(\Omega)||$ be the diameter of the search space. Particles are considered to be too close to each others and regrouping is triggered when the *normalized swarm radius*, δ_{norm}, satisfies the *premature convergence condition* defines as

$$\delta_{norm} = \frac{\delta(k)}{\text{diam}(\Omega)} < \varepsilon \tag{3.22}$$

where ε, called *stagnation threshold*, is a positive scalar value to be specified by the user. An empirical study, conducted by Evers, shows that selecting $\varepsilon = 1.1 \times 10^{-4}$ works well with the RegPSO because it allows enough time to pass so that the amount of deviation from global best seen on each dimension can successfully be used as an indicator of the swarm's uncertainty on that dimension.

When premature convergence is detected as given by the condition stated in Equation (3.22), the swarm is regrouped in a search space centered about the global best. The regrouping factor

$$\rho = \frac{6}{5\varepsilon} \tag{3.23}$$

stated by Evers works well even in the MHW machine case. Upon detection of premature convergence, the range in which particles are to be regrouped about the global best is calculated by dimension as the maximum of (i) the original range of the search space on dimension j and (ii) the product of the regrouping factor with the maximum distance along dimension j of any particle from global best:

$$\text{range}_j(\Omega) = \max\left(\text{range}_j(\Omega^0), \rho \max_{i\in\{1,\ldots,s\}} |x_{i,j}^{r-1} - g_j^{r-1}|\right) \tag{3.24}$$

The swarm is then regrouped by re-inizializing particles' position as

$$\vec{x}_i = \vec{g}^{r-1} + \vec{r}' \circ \text{range}(\Omega^r) - \frac{1}{2}\text{range})(\Omega^r) \tag{3.25}$$

where $\text{range}(\Omega^r) = [\text{range}_1(\Omega^r), \ldots, \text{range}_n(\Omega^r)]$, which utilizes a random vector $\vec{r}'$ to randomize particles within the implicitly defined search space (Equation (3.26)) with respective lower and upper bounds (Equation (3.27)).

$$\Omega^r = [x_1^{L,r}, x_1^{U,r}] \times [x_2^{L,r}, x_2^{U,r}] \times \ldots \times [x_n^{L,r}, x_n^{U,r}] \tag{3.26}$$

$$x_j^{L,r} = g_j^{r-1} - \frac{1}{2}\text{range}_j(\Omega^r) \tag{3.27a}$$

$$x_j^{U,r} = g_j^{r-1} + \frac{1}{2}\text{range}_j(\Omega^r) \tag{3.27b}$$

The swarm regrouping index, r, begins with 0 prior to the occurence of any regrouping and increments by one with each consecutive regrouping. Vector $\vec{g}^{r-1}$ is the global best at the last iteration of the previous grouping, and $\vec{x}_i^{r-1}$ is the position of particle $i$at the last iteration of the prior regrouping.
Since the PSO algorithm works well initially, the new RegPSO algorithm is not intended to make changes to the original position and velocity update equations, but merely to liberate the swarm from the well in which it has prematurely converged using an automatic regrouping mechanism.

3.4.4 Parameters tuning

In order to find the best parameters to achieve the optimal solution, without overdo in terms of computational time, all algorithms need a tuning phase. Specifically, the PSO algorithm, has two main topic to cover: particles & iterations. Particles are the solutions $\boldsymbol{\mu}_j$ that need to be evaluate through the Simulator (see Section 2.2). Granted that an higher number of particles means a better estimation of the objective function, augmenting drastacally

the particles number does not lead to outperform the results.

Iterations are instead the number of steps that the particles can do before stopping the algorithm. Since particles, iterations and objective functions are all correlated, it is straightforward to find a connection between these factors. The product between particles and iterations is the number of calls to the objective function, meaning how many times the objective function is "called" by the algorithm to evaluate a setup configuration.

$$\text{Particles}: \quad \eta = \{10, 50, 100, 200\}$$
$$\text{Iterations}: \quad \nu = \{15, 35, 60, 120\}$$

To locate the trade-off between particles and iterations, it has been developed a factorial experiment (2^k, k=4) with two factors (particles, iterations) and four levels for each factor. The sixteen experiments were replicate ten times to achieve better results.

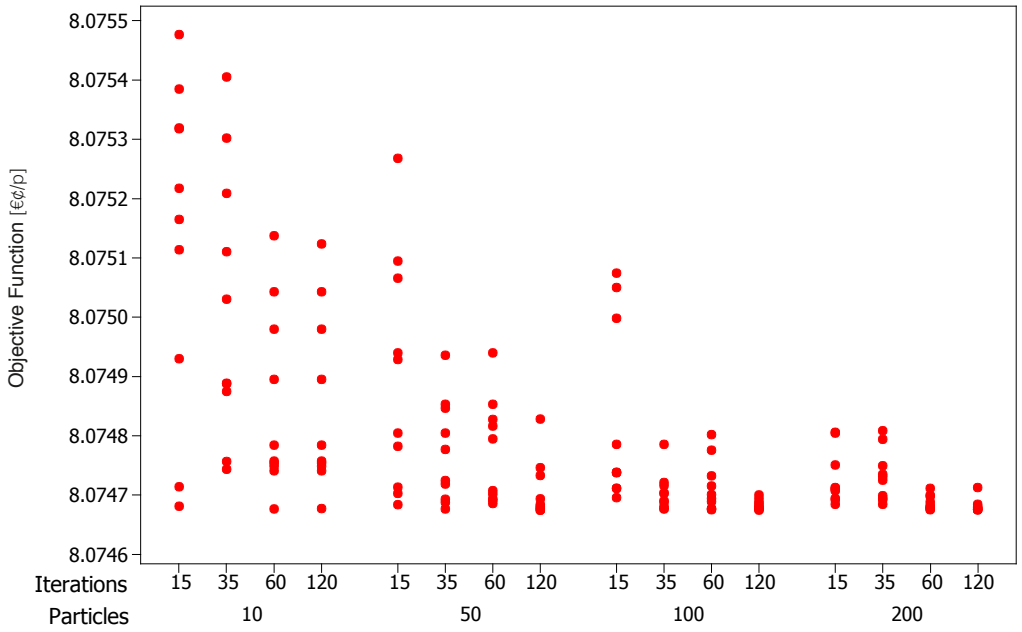


Figure 3.10: Individual Value Plot

According to the individual value plot in Figure (3.10), all experiments involving 15 iteration exhibit an extreamly high dispersion, expect those with 200 particles. This is consistent with a poor analisys of the solution space,

since 10 particles or 15 iterations are way inadequate to search the whole solution space. Thus, increase either the number or particles or the number of iterations give a better estimation of the objective function.

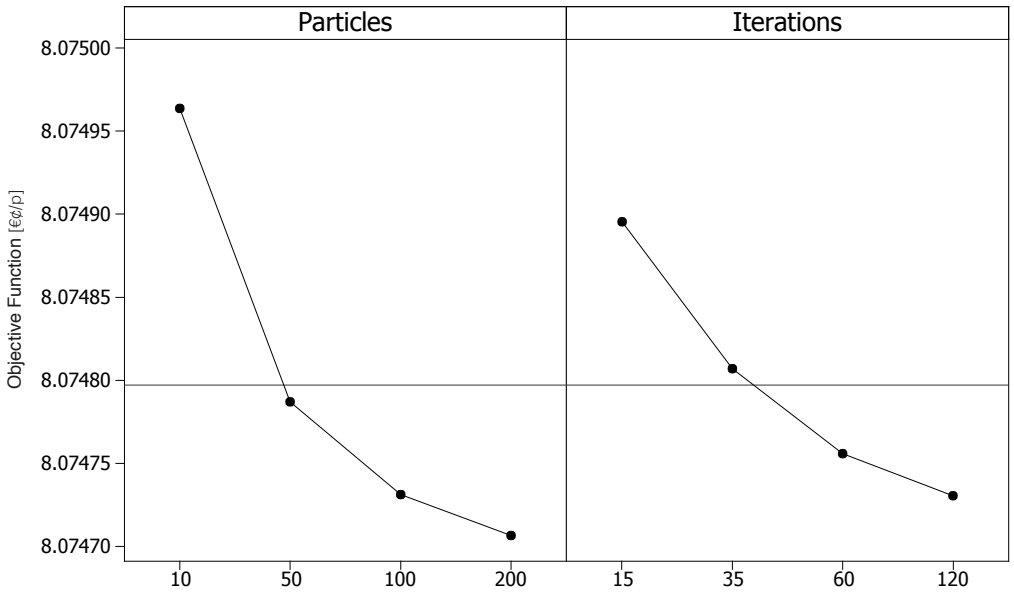


Figure 3.11: Main Effects Plot

The *Main Effects Plot* in Figure (3.11) exbihit that the optimal number of particles, in order to reach low values of objective function, seems to be $\eta = 50$ or $\eta = 100$. Even if the different levels of iterations number exibit more various gaps, the optimal number is 60. Even though, from the *Main Effects Plot* the correct number of iterations seems to be 60, the individual value plot leads us to $\nu = 35$. It is important to not forget that an higer number of iterations means higher computational times. The General Regression confirm the importance of the two parameters and suggest a curvature of the particles number. The regression equation below:

$$\begin{aligned} f(\boldsymbol{\mu}) = 8.07512 - 4.7504 \times 10^{-6} \cdot \eta - \\ + 2.2787 \times 10^{-6} \cdot \nu + 1.38829 \times 10^{-8} \cdot \eta^2 + \\ + 9.85766 \times 10^{-9} \cdot \eta \cdot \nu \end{aligned} \tag{3.29}$$

The results of the present study demonstrate that particles (η), iterations

(ν) and the square of the particles (η^2) are all significative. The model has: S = 0.000132167, R-Sq(adj) = 42.95%, PRESS = 2.878455×10^{-6} and R-Sq(pred) = 40.87%. The Analysis of Variance in Table (3.6) shows the same results, statistical significance for all factors except the interation ($\eta\cdot\nu$). The Lack of Fit has not significance, meaning that the model fits well.

Coefficients				
Term	**Coef**	**SE Coef**	**T**	**P**
Constant	8.07512	0.0000345	234094	0.000
η	-0.00000	0.0000006	-7	0.000
ν	-0.00000	0.0000004	-5	0.000
$\eta\cdot\nu$	0.00000	0.0000000	3	0.009
η^2	0.00000	0.0000000	5	0.000

Table 3.5: Multilevel Design

Analysis of Variance						
Source	**DF**	**Seq SS**	**Adj SS**	**Adj MS**	**F**	**P**
Regression	4	0.000022	0.0000022	0.000005	30.9260	0.0000000
η	1	0.000011	0.0000010	0.000010	54.5444	0.0000000
ν	1	0.000005	0.0000005	0.000005	28.4246	0.0000003
$\eta\cdot\nu$	1	0.000001	0.0000001	0.000001	6.9951	0.0090147
η^2	1	0.000005	0.0000005	0.000005	26.3202	0.0000009
Error	155	0.000027	0.0000027	0.000000		
Lack of Fit	11	0.000004	0.0000004	0.000000	1.9779	0.0346058
Pure Error	144	0.000024	0.0000024	0.000000		
Total	159	0.000049				

Table 3.6: Analysis of Variance

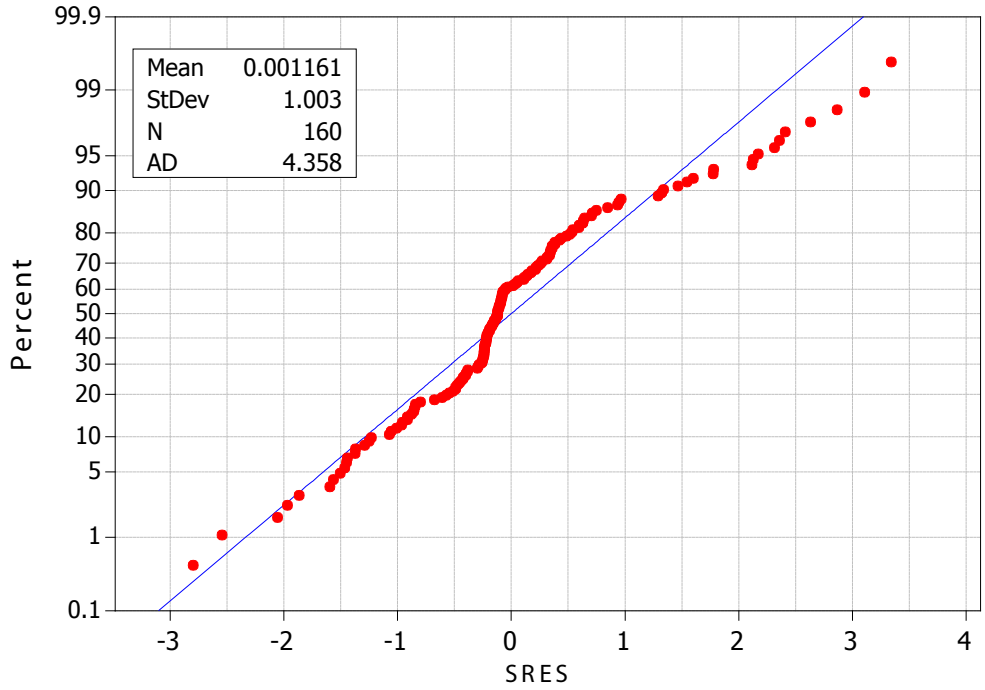


Figure 3.12: Probability Plot

Summing up, the best trade-off seems to be particles $\eta = 100$ and iterations $\nu = 35$. The *Probability Plot* in Figure (3.12) shows that there the standardized residuals are normal.

Chapter 4

Results

In the following, several noteworthy results are presented. In Section 4.1 a full comparison between three different approaches: Brute-force, RSM and PSO with two different policies: rework and reject, is displayed . Hence, a peculiar case, with the discretization of the two continuous approaches, is analyzed (see Section 4.2). The effect over the objective function value of the MHW machine hopper number is presented in Section 4.3.1 and a comparison between the Brute-force and the PSO, with the same cardinality is exhibited in Section 4.3.2. Furthermore, Section 4.3.3 presents a detailed sensitivity analysis of the objective function in order to understand the impact of the cost coefficients over the expected value $E(W)$ when a iterative knapsack problem is chosen (Section 2.2.2).

Moreover, Section 4.4 debates over the comparison between the three techniques in four different cardinality cases. Lastly, Section 4.5 presents the effect of PSO on current approaches to the problem, already implemented by packagers and, if a MHW machine, optimized by $dPSO$, with different hoppers, can lead to different performances. The RSM approach, state in literature [Beretta 2010], is used as baseline to perform comparisons between the other two methods.

4.1 Rework & Reject full comparison

As stated in Section 3.3.4 and Section 3.4.4, different setup parameters can deeply affect optimization performances, therefore all the simulations stated below are obtained according to the parameters in Table (4.1). RSM and PSO are developed both with reject and rework polices but, since RSM has an higher number of calls to the objective function, the iterative approach is not a viable way.

	Parameters					
Simulator	H	8		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	W_N	250	g	c_l	3.2	€¢/h
	W_L	241	g	c_f	7.34	€/p
RSM	M	8		ε	0.001	
PSO	η	100		ν	35	

Table 4.1: Simulation parameters

Table (4.2) shows the results for all three optimization methods and for both policies. The results of the rework case demonstrate that PSO has an extreamly superiority over the other two approaches. Regardless of the higher computational time t_{tot} of the Brufe-force approach, this heuristic technique performs a better objective function value than the grandient based method (RSM). On the other hand, with a reject policy, the gap between PSO and the other two methods drops significantly. The PSO stands as the best optimization method but, the Brute-force now performs slightly worse than the RSM.

Since both RSM and PSO algorithms works in a continuous solutions space, the $\boldsymbol{\mu}^*$ setup configurations stated in Table (4.2) are rounded to the

		$E(W^*)$	$\sigma(W^*)$	$f(\boldsymbol{\mu}^*)$	t_{tot}	
Rework	***BF***	249.6642	2.5014	8.3360	$\sim$ 5513”	(1h 31’ 53”)
	RSM	249.9350	1.7165	8.3411	$\sim$ 67”	(1’ 7”)
	PSO	241.0001	1.4×10^{-11}	8.0747	$\sim$ 80”	(1’ 20”)
Reject	***BF***	241.8108	0.8659	8.0990	$\sim$ 9708”	(2h 41’ 48”)
	RSM	241.8128	0.8254	8.0981	$\sim$ 188”	(3’ 8”)
	PSO	241.0019	9.9×10^{-12}	8.0747	$\sim$ 194”	(3’ 14”)
			$\boldsymbol{\mu}^*$			
Rework	***BF***	20 20 20 30 30 110 160 250				
	RSM	(17 21 31 34 37 43 51 60)				
	PSO	(45 85 108 133 171 173 189 193)				
Reject	***BF***	10 20 20 30 90 100 100 100				
	RSM	(6 18 27 58 66 72 78 84)				
	PSO	(35 45 62 88 91 117 148 150)				

Table 4.2: PSO, RSM & Brute-force comparison

nearest integer. In the following the continous solutions are presented:

$$
\begin{aligned}
RSM\ Rework:\ \boldsymbol{\mu}^* &= [17.0716\ 21.3746\ 31.2462\ 34.2847\ 37.0049\ 43.2856\ 50.8193\ 59.7093] \\
RSM\ Reject:\ \boldsymbol{\mu}^* &= [5.5147\ 17.6741\ 27.3775\ 58.0904\ 65.5691\ 72.0947\ 78.4950\ 83.5834] \\
PSO\ Rework:\ \boldsymbol{\mu}^* &= [44.938\ 85.28\ 107.6050\ 133.395\ 170.5452\ 172.6891\ 188.599\ 192.9711] \\
PSO\ Reject:\ \boldsymbol{\mu}^* &= [35.1253\ 45.2711\ 62.081\ 87.871\ 91.0492\ 117.3270\ 147.767\ 150.1175]
\end{aligned}
$$

Let us consider the Bariall Pedrignano plant as example to better understand the performances of PSO. In particular, the plant works on 3 shifts, performing a production of 600 tons a day, equal to 650 000 000 packages per year. Considering the rework case, the saving that can be performed, is roughly 0.2664 €¢ per package, leading to a massive saving of 1 731 600 € per year.

Instead for the reject policy, the gap between RSM and PSO becomes smaller, indeed the saving of 15 210 € can be seen as relevant in a statistical

way but, since these methods will be implemented in firms that weight performances thoughout money, the difference is no relevant

It is now clear that, in a continuous solution space, the PSO approach performs better with both policies. Let now turn the attention to the discrete case. The Brufe-force approach (BF), evaluating a number of solution of six order higer, leads to computational times that are not comparable to the other two methods but, exploiting a different case as comparison, the performances of BF can change.

4.2 BF, RSM & PSO in a low cardinality case

Let us consider a particular case[1] with a very low cardinality (see Section 3.2), such as where the target value is $W_T = 30$g and the raw material single unit weight is 2g, the Brufe-force has better performances.

	Parameters					
Simulator	H	8		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	W_N	30	g	c_l	3.2	€¢/h
	W_L	29	g	c_f	7.34	€/p
RSM	M	64		ε	0.001	
PSO	η	50		ν	60	

Table 4.3: Simulation parameters

According to the parameter presented in Table (4.3), the comparison between the discrete approach and the two continuous space-based method are stated in the Table (4.4).

where the real RSM setup configuration, obtained with 489 experiments,

[1] All data show in Table (4.4) are obtained simulating the MHW in the rework case.

	$E(W^*)$	$\sigma(W^*)$	$f(\boldsymbol{\mu}^*)$	t_{tot}	$\boldsymbol{\mu}^*$
BF	29.1084	0.1233	1.7179	$\sim$ 224"	2 4 8 8 8 8 8 8
RSM	29.9958	0.1347	1.7445	$\sim$ 112"	(1 2 3 4 5 5 6 10)
$\boldsymbol{RSM_{dscr}}$	29.9975	0.1364	1.7546	-	1 2 3 4 5 5 6 10
PSO	29.0004	5.8×10^{-13}	1.7147	$\sim$ 176"	(6 6 7 8 17 17 23 23)
$\boldsymbol{PSO_{dscr}}$	29.8114	0.8618	1.7390	-	6 6 7 8 17 17 23 23)

Table 4.4: PSO, RSM *&* Brute-force comparison in low cardinality case

and the PSO one are:

$$\begin{aligned} RSM: \boldsymbol{\mu}^* &= [\,0.7422\ 2.1513\ 2.7044\ 4.2598\ 4.8813\ 5.2139\ 6.4160\ 10.2321\,] \\ PSO: \boldsymbol{\mu}^* &= [\,5.5147\ 6.2741\ 7.3775\ 8.0904\ 16.5691\ 17.0947\ 23.4950\ 22.5834\,] \end{aligned}$$

Since RSM and PSO work in a continuos solutions space is fair to assume that these types of algorithms do not mimic well enough a real production campaign, above all in the cases where the raw material unit weight is remarkable high compared to the nominal package weight W_T. Furthermore, the approach is not a real case because the weight of a single unit of raw material is 2 g. Discretizing and analyzing the configurations $\boldsymbol{\mu}^*$ obtained by PSO and RSM through the simulator (called RSM_{dscr} and PSO_{dscr}) it is possibile obtain the real values of $E(W^*)$, $\sigma(W^*)$ and $f(\boldsymbol{\mu}^*)$.

As it was fair to assume, the discretized solutions, obtained by RSM and PSO, get worse. Interestingly are the gaps between the continuous cases and the discretized ones are interesting. The gap between the objective function values obtained with PSO and its discretization PSO_{dscr} is 0.0243 €¢/p. This difference produces a loss of 157 950 €/year for a firm[2] with 650 000 000 packages per year. Indeed, in low cardinality cases, as stated by Table (4.4), a Brute-force approach (BF) performs better, saving 0.0211 €¢/p per package that can be converted in 137 150 €/year.

[2]The Barilla pedrignano plant have a production rate of 600 tons per day.

4.3 PSO results

4.3.1 Hoppers trend on performances

Since the cardinal principle of the MHW machine is the multihead part, it is simple to understand that a change in the number of hoppers, can affects the performances as the objective function value and the computational time. Let us consider the PSO with a knapsack reject approach, a number of packages equal to 1000 and PSO parameters found in Section 3.4.4. The Figure (4.1) shows that the computational time increases as the number of hoppers increases, but the objective function value decreases exponentially[3] with an higher number of hoppers.

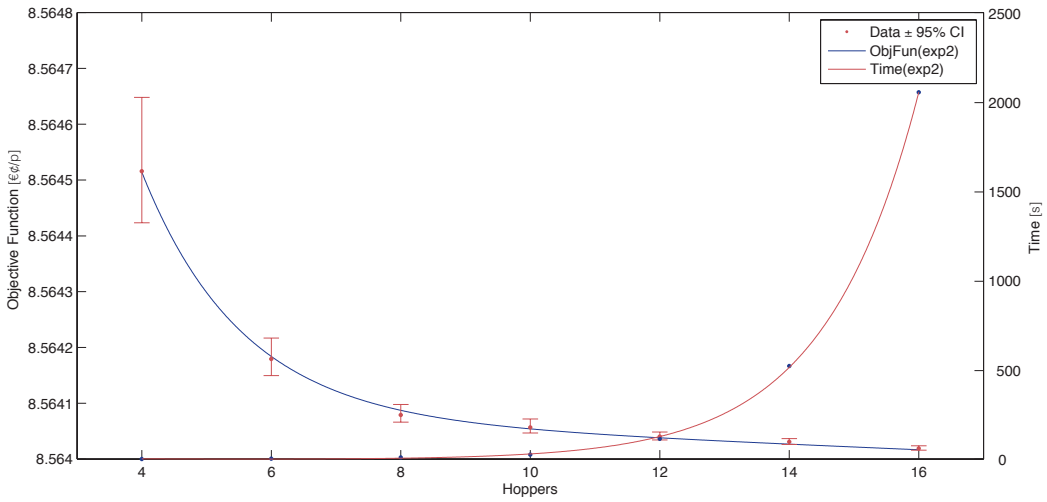


Figure 4.1: PSO Objective function *v.* Hoppers

In order to achieve a good estimation of the objective function standard deviation, the simulation was iterated 100 times. Therefore, the error bars, also known as confidence interval at 95%, are based on the *bootci* *Matlab*® function that computes the 95% bootstrap confidence intervals of the statistic computed by the function bootfun.

[3] A two decay modes was used to fit data $y = ae^{bx} + ce^{dx}$.

Since increasing the number of hoppers increases the number of combination computed (2^H), simulations with 14 or 16 hoppers has very large computational time for instance 2058 seconds for the $H = 16$ case. Moreover, the impact of hopper number on the objective function, using the PSO algorithm, is far from to be important since the gap from a 4 hopper case to a 16 hopper case is only 4.97×10^{-4}. A such small gap can gain importance with large pruduction rate, like 650 000 000 packages per year. A $H = 16$ setup, with this production rate, can save 32 464 € per year to detriment of 2048s instead of 5s (see Table (4.5)). Those differences are small thanks to a very good parameters settings made in advance. Thus, an higher number of hoppers can be usefull only in cases with a poor setup configuration.

		$E(W^*)$	$\sigma(W^*)$	$f(\mu^*)$	$\sigma(f(\mu^*))$	$Time$
	4	241.014	1.32×10^{-2}	8.56451	5.75×10^{-4}	5
	6	241.005	4.65×10^{-3}	8.56417	1.65×10^{-4}	7
	8	241.002	2.18×10^{-3}	8.56407	8.03×10^{-5}	13
H	10	241.001	8.61×10^{-4}	8.56405	6.20×10^{-5}	29
	12	241.001	7.21×10^{-4}	8.56404	3.61×10^{-5}	118
	14	241.001	4.21×10^{-4}	8.56403	2.68×10^{-5}	525
	16	241.000	2.23×10^{-4}	8.56401	1.93×10^{-5}	2058

Table 4.5: PSO - Hopper trend on performances computed on a Intel Xeon E5-2650 v2 PC computer running Matlab 2013a

4.3.2 PSO & Cardinality

Since the PSO has been faster and more precise in finding the best configuration available, a comparison between the Brute-force, using all type of reduction rules (K-Ratio and Hopper Volume Constraint) and PSO with the same computational load[4] is presented in Table (4.6). Thanks to the heurisitc rules stated in Section 3.2 the simulation cardinality from 1.05183×10^7

[4]Computational load means the number of calls to the objective function made by the optimizatior.

is reduced to 440 704. The K-ratio is set from 0.22 to 0.35 and the Hopper Volume Constraint from 10 to 190 g. As told, the PSO needs to evaluate the same amount of solution but, since the cardinality of this method can be adjusted using two different parameters (η and ν), two different cases are now stated:

PSO_η This approach increases the number of particles without change the number of iterations. Using $\eta = 4407$ and $\nu = 60$, the cardinality is 440 700.

PSO_ν Instead, this approach increases the iterations without change the particle number. Using $\eta = 50$ and $\nu = 12591$, the cardinality is 440 700.

	$\boldsymbol{Brute-force}$	$\boldsymbol{PSO_\eta}$	$\boldsymbol{PSO_\nu}$
$E(W^*)$	241.7986	241.0002	241.0002
$\sigma(W^*)$	0.9310	4.57×10^{-12}	5.57×10^{-12}
$f(\boldsymbol{\mu^*})$	8.0986	8.0747	8.0747
t_{tot}	$\sim$ 5882”	$\sim$ 6643”	$\sim$ 10865”

Table 4.6: PSO *v.* Brute-force - equal cardinality in a reject case

As pointed out by Table (4.6), the Brufe-force search has a higher objective function value, due to its discrete solution space and its 10g coarsness but, it can evaluate the same amount of configurations in a slightly lower time thanks to a neat and loop-free code. Moreover, it is now clear that the number of iterations ν affects deeply the computational time respect to the number of particles η. This behavior is granted by the MHW simulator nature that base itself only on vector calculus exploiting more deeply the computational power.

4.3.3 Objective Function Sensitivity analysis

The discharge combination is affected, directly, by the real time rules. Hence, it is very important to understand how the objective function, used instead

of a knapsack problem, can affect those discharges. A Design of Experiment full factorial plan with 3 factors (c_u, c_p and c_f), 3 levels for each factor (see Table (4.7)) and 10 replicates is used to estimate a possibile effect, of those three factors, onto the response $E(W^*)$. The cost coefficient c_l is omitted from the study since in optimal solution it is not relevant.

Factor	**Type**	**Levels**	**Values**		
c_u	fixed	3	0.005	0.03	0.1
c_f	fixed	3	73.4	734	7340
c_p	fixed	3	0.1	0.6	1

Table 4.7: Multilevel Factorial Design

From the Individual Value Plot in Figure (4.2) it is easy to understand how little is the impact over the response.

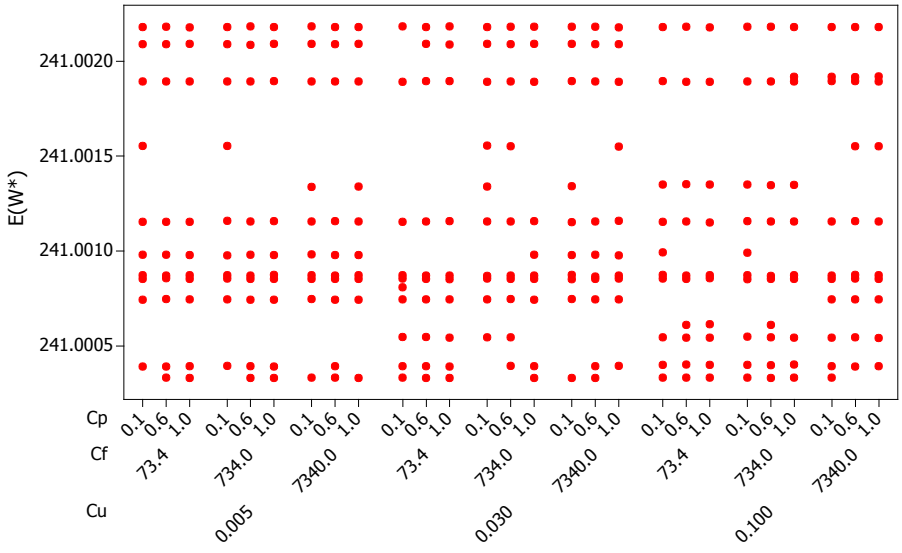
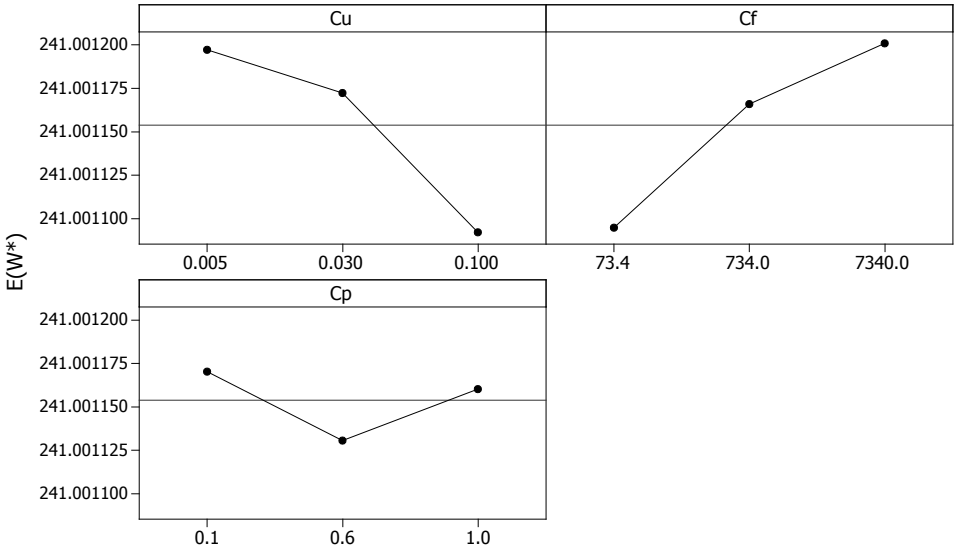


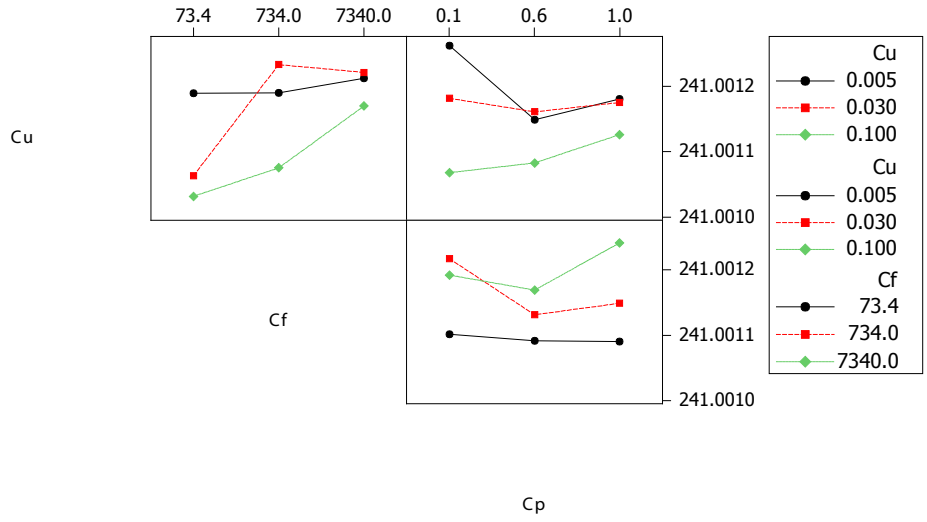
Figure 4.2: Individual Value Plot

The Main Effects plot seems show some differences but since the y-axis scale is around the third decimal point these differences will show as ininfluent on the ANOVA table. The Interaction plot in Figure (4.3) does not

display any important interaction.



(a) Main Effects Plot



(b) Interaction Plot

Figure 4.3: Main Effects & Interaction Plot

As pointed out before, the ANOVA in Table (4.8) shows no significance at all, except to the cost of the raw material c_u with a P-value of 0.001. The

standardized residuals are far from normality as marked by Figure (4.4) but they do not exhibit any strange behavior in the scatterplot in Figure (4.5).

Source	DF	Seq SS	Adj SS	Adj MS	F	P
c_u	2	0.0000312	0.0000312	0.0000156	7.44	0.001
c_f	2	0.0000003	0.0000003	0.0000002	0.07	0.930
c_p	2	0.0000000	0.0000000	0.0000000	0.01	0.988
$c_u\, c_f$	4	0.0000001	0.0000001	0.0000000	0.01	1.000
$c_u\, c_p$	4	0.0000000	0.0000000	0.0000000	0.00	1.000
$c_f\, c_p$	4	0.0000001	0.0000001	0.0000000	0.01	1.000
$c_u\, c_f\, c_p$	8	0.0000002	0.0000002	0.0000000	0.01	1.000
Error	243	0.0005100	0.0005100	0.0000021		
Total	269	0.0005420				

Table 4.8: Multilevel Factorial Design

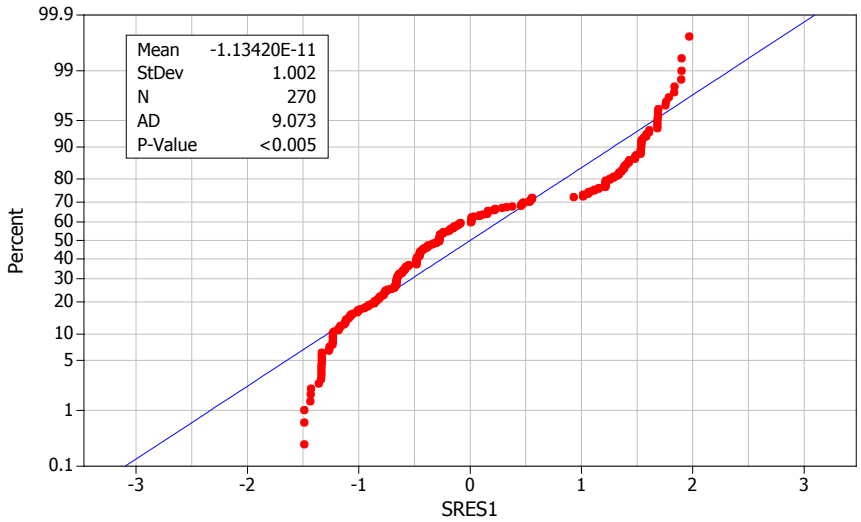


Figure 4.4: Probability Plot

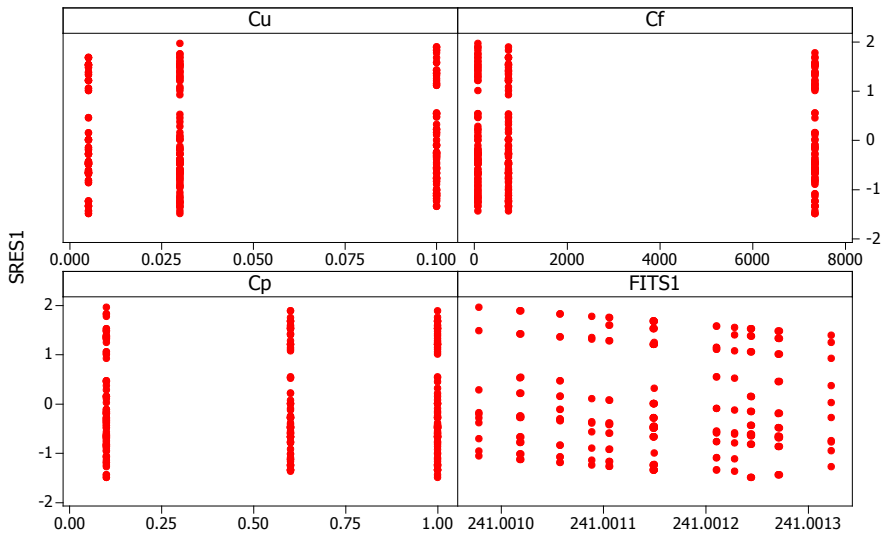


Figure 4.5: Scatter Plot

4.3.4 Discrete PSO

As stated before, PSO works in a continuous solutions space, therefore, the performance are much better than a discrete case since there are more shades, and more levels, that can be used to hit the target weight. The algorithm can be switched to a discreate case only by rounding up the spatial points of particles before the simulation begins.

	Parameters					
Simulator	H	8		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	W_N	250	g	c_l	3.2	€¢/h
	W_L	241	g	c_f	7.34	€/p
PSO	η	100		ν	35	

Table 4.9: PSO simulation input parameters

Hence, the performances decrease because the combinations available for the discharged package are less. According to Table (4.9), where are presented the parameters of the simulation/optimization, the performances of the

discrete approach are presented in Table (4.10).

	PSO	***dPSO***
$E(W^*)$	241.0019	241.1203
$\sigma(W^*)$	9.9493×10^{-12}	0.3254
$f(\boldsymbol{\mu}^*)$	8.0747	8.0783
t_{tot}	$\sim$ 74" (1' 14")	$\sim$ 77" (1' 17")
$\boldsymbol{\mu}^*$	(35 45 62 88 91 117 148 150)	33 83 99 105 106 109 135 136

Table 4.10: PSO *v.* dPSO

The discrete PSO (*dPSO*), besides finding a different configuration $\boldsymbol{\mu}^*$, performs worse that the continuous one. These differences do not need to be seen as negative since the dicrete case is more realistic than the standard PSO. Therefore, the *dPSO* combines the performances of standard PSO, such as the computational speed and a lower objective function value, and the realism of the Brute-force search. This appraoch can be appointed as the best solution. Depending on the raw material unit weight the discretized PSO can give a more accurate objective function value.

4.4 Methodologies Comparison

As pointed out in Section 4.1, the discrete approach of an exaustive search does not pay out good performances in reject and rework policies both since the two optimizators, RSM and PSO, have exibhit a more accurate behavior besides a lower computational time. Then, in Section 4.2 we saw that, evaluating the three methods with the same cardinality can lead to a different outcome. In the following, in disagreement with Section 3.3.4 and Section 3.4.4 where are stated the parameters tuning for both RSM and PSO, a comparison between the three methodologies with the same cardinality is proposed.

	Parameters					
Simulator	H	8		c_u	0.03	€¢/g
	P	3000		c_p	0.6	€¢/p
	W_N	250	g	c_l	3.2	€¢/h
	W_L	241	g	c_f	7.34	€/p
RSM	M	$2 \rightarrow 8$		ε	0.001	
PSO	η	$18 \rightarrow 80$		ν	$35 \rightarrow 49$	

Table 4.11: Simulation parameters

In order to grant a correct comparison between the three methods, all the comparisons are made with the same number of objective function calls, a good proxy of the computational load. The RSM is used as baseline, meaning that the number of experiments equals to the number of calls to the objective function ($\#f(\boldsymbol{\mu}) = Exp$) made by this optimizator, is be met by the PSO multiplying the particles and the iteration number ($\#f(\boldsymbol{\mu}) = \eta \cdot \nu$) and by Brute-force by sampling the total amount of solutions. Four different cases are then outlined and the results are shown in Figure (4.6):

Case 1: RSM works with $M = 2$ calling ~ 600 times the objective function. The PSO parameters to met the same amount are $\eta = 18$ and $\nu = 35$ and the sampling factor for the BF approch is 0.0384%.

Case 2: RSM works with $M = 4$ calling ~ 1600 times the objective function. The PSO parameters to met the same amount are $\eta = 45$ and $\nu = 35$ and the sampling factor for the BF approch is 0.1024%.

Case 3: RSM works with $M = 6$ calling ~ 2280 times the objective function. The PSO parameters to met the same amount are $\eta = 60$ and $\nu = 38$ and the sampling factor for the BF approch is 0.1459%.

Case 4: RSM works with $M = 8$ calling ~ 3920 times the objective function. The PSO parameters to met the same amount are $\eta = 80$ and $\nu = 49$ and the sampling factor for the BF approch is 0.2509%.

The shape of the graph in Figure (4.6) shows that there is huge difference between PSO and the other two methods. The grandient-based approach as like as the Brute-force, has a strong apathy to the parameter shift given that the difference between the objective function calls can be accounted as less than a third decimal gap. On the other hand, PSO shows a wide gap between case 1 and case 2, shifting from a ~ 8.084€¢/p to a ~ 8.078€¢/p, but the cases 3 and 4 exhibith a downward trend only for the standard deviation.

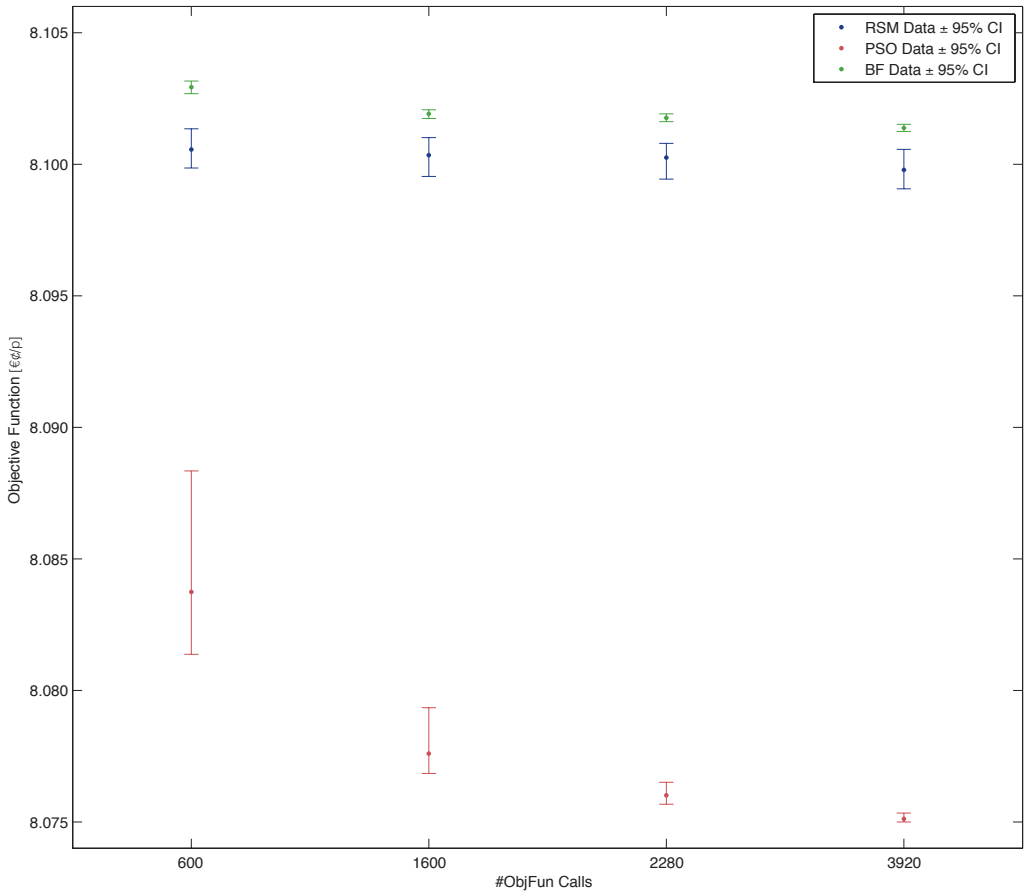


Figure 4.6: PSO *v.* RSM and BF with equal objective function calls

According to Table (4.12), the objective function values reached by PSO are lower than BF and than RSM in all four cases. On the contrary, RSM always performs better regarding the computational times thanks to the parallel computing that grant a good CPU saturation. Re-coding the PSO with

		RSM	***BF***	***PSO***
Case 1	$f(\boldsymbol{\mu}^*)$	8.1006	8.1029	8.0837
	t_{tot}	$\sim$ 29”	$\sim$ 43”	$\sim$ 79”
Case 2	$f(\boldsymbol{\mu}^*)$	8.1003	8.1019	8.0776
	t_{tot}	$\sim$ 38”	$\sim$ 60”	$\sim$ 119”
Case 3	$f(\boldsymbol{\mu}^*)$	8.1003	8.1018	8.0760
	t_{tot}	$\sim$ 46”	$\sim$ 70”	$\sim$ 133”
Case 4	$f(\boldsymbol{\mu}^*)$	8.0998	8.1014	8.0751
	t_{tot}	$\sim$ 53”	$\sim$ 93”	$\sim$ 180”

Table 4.12: Full comparison with equal objective function calls

the same approach can possibily lead to met the same computational times. The objective function gap, between PSO and the other two techniques, becomes greater shifting from case 1 to case 4, upholding that PSO keeps performing better with more particles and iterations.

Those gaps becomes more meaningfull in large scale production. Implementing PSO instead of RSM can lead, in a 650 000 000 package per year production, to save 160 550 € per year, roughly the 0.31% of the total production costs. Therefore, PSO are the best approach to solve the setup configuration problem.

4.5 PSO & current solutions

As mentioned in the first chapter, the setup configuration problem for a MHW machine is relatively new therefore the firms that have to tackle with this problem oftenly choose euristic or operator experience based setup. In the following, the PSO algorithm is compared with two actual cases in order to understand the economic impact of a more complex method on the total production costs.

Case 1: The number of discharging hoppers is set and the setup configur-

ation $\boldsymbol{\mu}^*$ is obtained dividing the target package weight W_T for the number of discharging hoppers. Usually the experience has lead to fix the number of the hopper to be open to 4.

$$\boldsymbol{\mu}^* = \left[\frac{250}{4} \frac{250}{4} \frac{250}{4} \frac{250}{4} \frac{250}{4} \frac{250}{4} \frac{250}{4} \frac{250}{4} \right]$$

Case 2: The second case consists to a Brute-force search with a very high sampling factor S, to reduce the computational times. Since a MHW machine with $H = 8$, $L = 25$ presents a cardinality of 1.05183×10^7 the sampling factor is set to 0.01%. The solutions considered are 1052.

While the first case have a fixed setup configuration, the second case and the PSO have to firstly find the setup configuration and then evaluate it.

	Case 1	Case 2	***PSO***
$E(W^*)$	249.1788	244.0302	241.0007
$\sigma(W^*)$	8.7903	2.9245	5.05×10^{-12}
$f(\boldsymbol{\mu}^*)$	8.3239	8.1656	8.0747
t_{tot}	~ 11”	~ 20”	~ 170”

Table 4.13: PSO *v.* Current solutions

The setup configuration obtained by the algorithms are:

$$\begin{aligned}
\boldsymbol{\mu}^*_{Case\,1} &= [\,62.5\ 62.5\ 62.5\ 62.5\ 62.5\ 62.5\ 62.5\ 62.5\,] \\
\boldsymbol{\mu}^*_{Case\,2} &= [\,10\ 20\ 40\ 60\ 90\ 100\ 120\ 120\,] \\
\boldsymbol{\mu}^*_{PSO} &= [\,36.6631\ 45.3232\ 45.5693\ 78.7455\ 85.9103\ 143.1920\ 145.3615\ 244.2853\,]
\end{aligned}$$

The PSO performances are straighforword respect to both current approaches. Since enumeration with an highly sampling factor performs better than the “experience” method and given that it is not so well known, we advise to implement this approach. A firm with an high production rate, such as 650 000 000 package per year, can reach a 1 028 950 €/year saving adopting the Case 2. Firms with a more statistical approach and with the ability to code and implement the PSO approach can perform, respect case 1, a 1 619 800 €/year saving, roughly the 3% of the total production costs.

4.5.1 Effect of more hoppers

The previous section shows that using PSO to find an optimal setup configuration $\boldsymbol{\mu}$ leads to a 3% saving on the total production costs. Since to fix the number of the hoppers to be open and the weights is a current solution, the producers of packaging machines tend to add more hoppers in order to increase the performances. This tendency can no longer be needed since the boost of hoppers, leads to no improvement at all (see Fig. (4.7)).

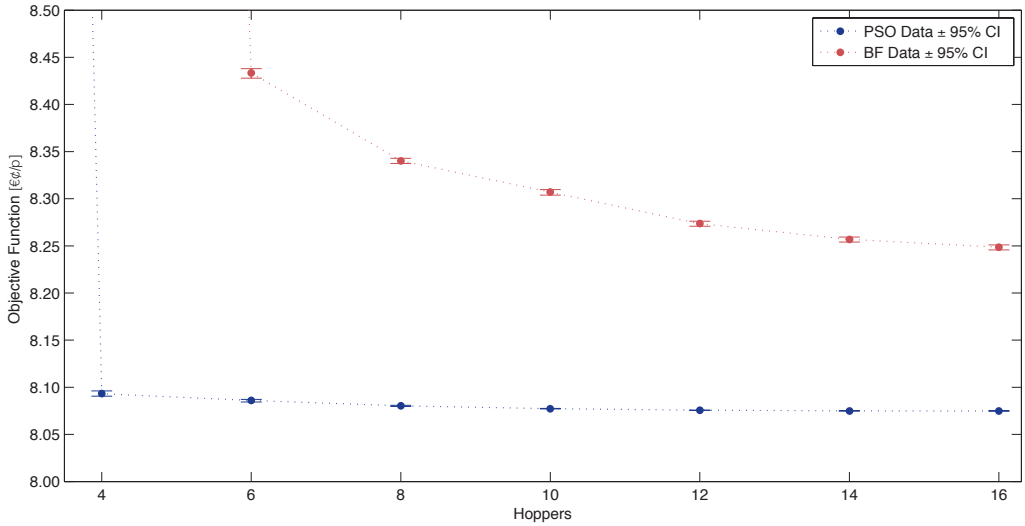


Figure 4.7: Effect of more hoppers on the objective function

According to Figure (4.7), the objective function value rose sharply below 6 hoppers for the "Case 1" and below 4 hoppers for the PSO, before plunging down. The non optimal solution used in industry take advantage by rise the number of hoppers. On the other hand, PSO, finding the best setup configuration $\boldsymbol{\mu}^*$, shows an aphaty to the number of hoppers. Hence, chosing PSO instead of an experience based method other than the saving produced by a low production cost per package, gives the opportunity to exploit old MHW machine, with less hoppers, saving more fixed costs.

Conclusions

The Multihead Weigher machine, a computer-controlled machine used to fill a package of products at its target weight, has a wide range of applications in the food industry for dried foods such as pasta, pet foods, coffee, cereals, or fresh foods such as salad, spare ribs, etc. Its applications cover also the non-food areas, for instance the paper clip packages are filled using this machine. Despite its worldwide spread and the high number of manufacturers, scientific studies aimed at studying its characteristics in order to reach optimal performance are few and most of the literature on this topic is mainly commercial (e.g. patents, machine brochures).

The main aim of this thesis was to exploit the logic structures of the machine in order to solve one of the two main problem: the operation and the setup. The operation problem has as goal to select the best hopper subset to open according to the product, the time constraints and the objective function. This is a kind of knapsack problem. Instead, the setup problem deals with the definition of the average quantity of the product delivered by the radial feeders to each hopper at each cycle. Since an improper machine setup turns into a lack of machine efficiency, and since nowadays this relies on the operator's skills and experience this thesis focused on the last problem. The selection of the setpoints of a MHW is critically important since it directly affects the quantity of product delivered in each package, and consequently the production costs for the producer, and the loss to the costumer due to the deviation of the package weight from specification. The importance of the production costs, and the need to evaluate the performance of a machine setup, leads to formalize an economic objective function based on the production costs (Section 2.1). The search of the optimal setup configuration,

due to the higly number of solutions and the computational expense of the analytic way, has been tackled with a simulator thats mimics the behaviour of a simple MHW machine (Section 2.2) and an optimizer, that helps us to look for the best setup configuration. The analysis of the solution space via Brute-force gave us the opportinity to find three different heuristic rules: sapling, Hopper Volume Constraint and K-Ratio. These rules, above all the K-Ratio, proved to be excellent to reduce the number of solutions and compute the setup problem faster (Chapter 3). Despite the number of solutions was reduced by 70-90%, larger problem can not be solved via an enumerative way. Therefore, the optimization of the simulation output was commited to the Particle Swarm Optimization that proves to be excellent managing problems with high density of optimum.

The comparison between optimizators, in terms of performances, shows that the Brute-force with heurisitc reduction perform better in problems with a small subset of solutions but, with an higher number of setup configurations to be evaluate the PSO approach, performs better than both Brute-force and the previous method stated in literature [Beretta & Semeraro 2012]. Hence, dealing the setup configuration problem of a MHW machine has become more convenient, in terms of production costs, with the approaches recommended by this work. Above all, using a discrete PSO approach, respect the current solutions exploited by packagers, to solve the setup configuration problem can lead to save around the 3% of the total production costs that, for an highly rate production site, like Barilla Pedrignano plant, can become around 1 619 800 € per year.

Because of the limits on the size of the problem and time, we did not consider a situation more wide (MHW machine with more hoppers) or more complex, however we provided a proof that, with an optimal setup configuration, a MHW machine with more hoppers do not perform better (Section 4.5.1).

Bibliography

Al Sultan, K. S. [1994], 'An algorithm for determination of the optimal terget values for two machines in series with quality sampling plans', *International Journal of Production Research* **32**, 37–45.

Al Sultan, K. S. & M. A. Al-Fawzan [1997], 'Variance reduction in a process with random linear drift', *International Journal of Production Research* **35**(6), 1523–1533.

Al Sultan, K. S. & M. Pulak [1997], 'Process improvement by variance reduction for a single filling operation with rectifying inspection', *Production Planning and Control* **8**(5), 431–436.

Beretta, A. [2010], Procedura RSM per la configurazione di una pesatrice multitesta (in italian), Master's thesis, Politecnico di Milano, Milano, Italy.

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

Bettes, D.C. [1962], 'Finding an optimal target value in relation to a fixed lower limit and an arbitrary upper limit', *Applied Statistics* **11**(3), 202–210.

Bisgaard, S., W. G. Hunter & L. Pallessen [1984], 'Economic selection of quality of manufactured product', *Technometrics* **26**, 9–18.

Boucher, T. O. & M. A. Jafari [1991], 'The optimum target value for single filling operations with quality sampling plans', *Journal of Quality Technology* **23**, 44–47.

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

Box, G. E. P. & N. R. Draper [1987], *Empirical Model-Building and Response Surface*, John Wiley & Sons.

Box, G. E. P., W. G. Hunter & J. S. Hunter [1978], *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*, John Wiley & Sons.

Carlsson, O. [1984], 'Determining the most profitable process level for a production process under different sales conditions', *Journal of Quality Tecnology* **16**, 44–49.

Carlsson, O. [1989], 'Economic selection of a process level under acceptance sampling by variables', *Engineering Costs and Production Economics* **16**, 69–78.

Council Directive [1976], '76/211/eec', OJ L 46, 21.2.1976.

Del Castillo, E. [1997], 'Stopping rules for steepest ascent in experimental optimization', *COMMUN STAT PART B SIMUL COMPUT* **26**(4), 1599–1615.

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

del Castillo, E., A. Beretta & Q. Semeraro [2014], 'Analysis and optimal setup of a multihead weigher machine', *Submitted* .

Dorigo, M., M. Birattari & T. Stutzle [2006], 'Ant colony optimization', *Computational Intelligence Magazine, IEEE* **1**(4), 28–39.

Eberhart, R. C. & J. Kennedy [1995], A new optimizer using particle swarm theory, *in* M.'95, ed., 'Micro Machine and Human Science', Proceedings of the Sixth International Symposium on, Nagoya, Japan, pp. 39–43.

Eberhart, R. & Y. Shi [2000], Comparing inertia weights and constriction factors in particle swarm optimization, Proceedings Congress on Evolutionary Computation, La Jolla, CA, pp. 84–88.

Evers, G. [2009], An automatic regrouping mechanism to deal with stagnation in particle swarm optimization, Master's thesis, University of Texas-Pan American.

Evers, G.I. & M. Ben Ghalia [2009], Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks, *in* 'Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on', pp. 3901–3908.

Ferrario, F. & P. Laghezza [1999], Proposta di una metodologia di configurazione e gestione di una pesatrice multitesta (in italian), Master's thesis, Politecnico di Milano, Milano, Italy.

GE, Goldberg [1989], *Genetic algorithms in search optimization and machine learning*, Addison Wesley.

Gilmour, S. G. & R. Mead [1995], 'Stopping rules for sequences of factorial designs', *Applied statistics* pp. 343–355.

Golhar, D. Y. [1987], 'Determination of the best mean contents for a canning problem', *Journal of Quality Technology* **19**, 82–84.

Golhar, D. Y. & S. M. Pollock [1988], 'Determination of the optimal process mean and the upper limit for a canning problem', *Journal of Quality Technology* **20**, 189–192.

Haze, S. [1987], 'Combinatorial weighing method and apparatus', Google Patents. US Patent 4,642,788.

Hunter, W. G. & C. P. Kartha [1977], 'Determining the most profitable target value for a production process', *Journal of Quality Technology* **9**, 176–181.

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

Imahori, S., Y. Karuno, R. Nishizaki & Y. Yoshimoto [2012], Duplex and quasi-duplex operations in automated food packing systems, *in* '2012 IEEE/SICE International Symposium on System Integration', Kyushu University, Fokuoka, Japan, pp. 810–815.

Imahori, S., Y. Karuno & Y. Yoshimoto [2010], Dynamic programming algorithms for duplex food packing problems, *in* 'Proceeding of the 8th IEEE International Conference on Industrial Informatics', pp. 857–86212–17.

Incletolli, S. [2012], Configurazione di una pesatrice multitesta (in italian), Master's thesis, Politecnico di Milano, Milano, Italy.

Joshi, S., H. D. Sherali & J. D. Tew [1998], 'An enhanced response surface methodology (RSM) algorithm using gradient deflection and second-order search strategies', *Computers & operations research* **25**(7), 531–541.

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

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

Karuno, Y., H. Nagamochi & X. Wang [2009], Combinatorial optimization problems and algorithms in double-layered food packing equipments, *in* 'Proceeding of 4th International Symposium on Scheduling (ISS 2009)', pp. 12–17.

Karuno, Y., H. Nagamochi & X. Wang [2010], 'Optimization problems and algorithms in double-layered food packing systems', *Journal of Advanced Mechanical Design, Systems, and Manufacturing* **4**(3), 605–615.

Kennedy, J. & R. C. Eberhart [1995], Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948.

Khuri, A. I. & J. A. Cornell [1996], *Response Surface: design and analyses*, Marcel Dekker, Inc.

Kirkpatrick, S., C. D. Gelatt & M. P. Vecchi [1983], 'Optimization by simulated annealing', *Science* **220**(4598), 671–680.
URL: *http://home.gwu.edu/~stroud/classics/KirkpatrickGelattVecchi83.pdf*

Kleijnen, J. P. C., D. Den Hertog & E. Angün [2004], 'Response surface methodology's steepest ascent and step size revisited', *European Journal of Operational Research* **159**(1), 121–131.

Kolesar, P. J: [1967], 'A branch and bound algorithm for the knapsack problem', *Management Science* **13**, 723–735.

Konishi, S. [1983], 'Combinatorial weighing system', Google Patents. US Patent 4,399,880.

Land, A. H. & A. G. Doig [1960], 'An Automatic Method for Solving Discrete Programming Problems', *Econometrica* **28**, 497–520.

Martello, Silvano & Paolo Toth [1990], *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Inc., New York, NY, USA.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller & Edward Teller [1953], 'Equation of state calculations by fast computing machines', *The Journal of Chemical Physics* **21**(6), 1087–1092.
URL: *http://dx.doi.org/10.1063/1.1699114*

Miró-Quesada, G. & E. Del Castillo [2004], 'An enhanced recursive stopping rule for steepest ascent searches in response surface methodology', *Communications in Statistics-Simulation and Computation* **33**(1), 201–228.

Myers, R. H. & A. I. Khuri [1979], 'A new procedure for steepest ascent', *Communications in Statistics-Theory and Methods* **8**(14), 1359–1376.

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

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

Nelson, L. S. [1979], 'Nomograph for setting process to minimize scrap cost', *Journal of Quality Technology* **11**, 48–49.

Nicolai, R. P., R. Dekker, N. Piersma & G. J. van Oortmarssen [2004], Automated response surface methodology for stochastic optimization models with unknown variance, *in* 'Proceedings of the 36th conference on Winter simulation', pp. 491–499.

Nicolai, R. & R. Dekker [2009], 'Automated response surface methodology for simulation optimization models with unknown variance', *Quality Technology & Quantitative Management* **6**(3), 1–28.

Pulak, M. F. S. & K. S. Al Sultan [1989], 'The optimum targeting for a single filling operation with rectifying inspection', *Omega* **24**(6), 727–733.

Sakaeda, K. & T. Sashiki [1986], 'Combinatorial weighing method and apparatus therefor', Google Patents. US Patent 4,609,058.

Schimdt, R. L. & P. E. Pfeifer [1989], 'An economic evaluation of improvements in process capability for a single level canning problem', *Journal of Quality Technology* **21**, 16–19.

Shi, Y. & R. Eberhart [1998], A modified particle swarm optimizer, *in* 'Evolutionary Computation Proceedings', IEEE World Congress on Computational Intelligence, Anchorage, AK, pp. 69–73.

Smith, D. E. [1976], 'Automatic optimum-seeking program for digital simulation', *Simulation* **27**(1), 27–31.

Springer, C.H. [1951], 'A method of determining the most economic position of a process mean', *Industrial Quality Control* **8**, 36–39.

The Weighing & Force Measurement Panel [2010], *A Guide to Dynamic Weighing for Industry*, Learned Society Board of the Institute of Measurement and Control, 87 Gower Street, London, WC1E 6AF.

Van Den Bergh, Frans [2002], An Analysis of Particle Swarm Optimizers, PhD thesis, Pretoria, South Africa, South Africa. AAI0804353.

Yamato Corporation [2012], 'Optimiziong product flow on the packaging line', Key Technology, Inc.