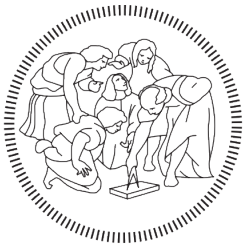


POLITECNICO
MILANO 1863

School of Industrial Engineering
Department of Mechanical Engineering

The Cardinality-Constrained Approach applied to manufacturing problems

Advisor: Andrea Matta
Co-advisor: Ettore Lanzarone

Student:
Giovanni Lugaresi
Student number 856645

Academic Year 2015-2016

不入虎穴, 焉得虎子

How can you catch tiger cubs without entering the tiger's lair?
[Chinese idiom]

Acknowledgments

I am grateful towards all the people who made this work possible.

First of all, I would like to thank professor Andrea Matta, who gave me the opportunity to join his research team in Shanghai, guided me throughout all the duration of this project, and taught me the importance of properly present a work, the attention to detail and team spirit.

I am thankful to Ph.D. Nicla Frigerio, with whom I shared both work and leisure moments, and I am glad to call a great friend of mine.

I am very grateful to professor Ettore Lanzarone, who was always kind and patient to help me from wherever in the world.

This work would not have been possible without professor Tullio Tolio, who gave me not only the possibility to do this experience, but also inspired me through his lessons to love this discipline.

I am obliged to all the "Prof. Matta's world": Lin Ziwei, Mengyi Zhang, Zhou Chunmeng, Su Huiting, Kan Li. They have been of great help in several occasions, and they remain my good friends in Shanghai.

I cannot forget to thank professor Jim McGregor Smith, who encouraged me to be confident in my work and to never stop looking for things that I love to do.

My experience in Shanghai was so enjoyable thanks to all the great friends I met: it would be too long to mention all, but I will never forget anyone of them. Among all, my gratitude goes to Jacopo Bonechi, my partner in many adventures.

I am indebted to my girlfriend Vivian, for being so patient with me being so far, never denying me her loving support.

Last but not least, I am grateful to my Family, for being always besides me in my choices, and supporting me in every moment.

Contents

List of Figures

List of Tables

Abstract

Traditionally, several models in the manufacturing literature assume that the main parameters which describe the systems' behavior are deterministic. However, in many situations the uncertainty affecting those parameters should not be neglected. This thesis provides the robust version of three of the most known problems in manufacturing: the Part Type Selection Problem, the Machine Loading Problem, and Buffer Allocation Problem (BAP). First, a literature review identifies the main methodologies to address parameters' uncertainty, and the core issues regarding this field. The advantages of the Cardinality-Constrained approach motivate its application throughout this work. The robust counterpart of the Part Type Selection Problem has been applied both in test and real case studies, that confirmed its applicability in the practice with very small computational effort. The robust version of The Machine Loading Problem has also been developed: again, results show that the model can be applied with very short computational times to real size cases. The robust version of the Buffer Allocation Problem has not been derived in its analytical form, since by integrating it with the Cardinality-Constrained formulation the linearity of the Optimization Model would have been lost. Nevertheless, the concept of cardinality has been used in the development of a Matheuristic algorithm to calculate the bounds of the BAP. A Matlab® routine generates failure-patterns as they constitute the inputs of the optimization model, respecting the cardinality

of the set. The algorithm is a Tabu Search Matheuristic, and permits to seek the combinations of those failures which impact the most on the objective function. The computational times are promising for future developments aiming at the application to real size problems.

Key Words: Robust Optimization; Machine Loading; Buffer Allocation; Part Type Selection; Cardinality-Constrained approach; Tabu Search; Matheuristics.

Sommario

Tradizionalmente, molti modelli proposti nella letteratura del manufacturing assumono che i principali parametri che descrivono l'andamento di un sistema siano deterministici. Ad ogni modo, in molte situazioni non si dovrebbe trascurare l'incertezza che influenza questi parametri. Questa tesi fornisce la versione robusta di tre dei piu noti problemi nel manufacturing: Part Type Selection Problem, Machine Loading Problem, e Buffer Allocation Problem (BAP). Inizialmente, un'analisi bibliografica identifica le metodologie principali per affrontare l'incertezza dei parametri e le principali questioni di questa disciplina. I vantaggi dell'approccio Cardinality-Constrained ne motivano l'uso in questo lavoro. La versione robusta del problema di Part Type Selection e stata applicata sia in casi fittizi che reali, che hanno cofermato la possibilita di applicarlo nella pratica con un basso sforzo computazionale. E stata anche sviluppata la versione robusta del problema di Machine Loading. Di nuovo, i risultati mostrano che il modello puo essere applicato a casi di dimensioni reali con brevi tempistiche computazionali. La versione robusta del problema di Buffer Allocation non e stata sviluppata in versione analitica perche integrando il problema di BAP con l'approccio Cardinality-Constrained viene persa la linearita del modello di ottimizzazione. Ciononostante, il concetto di cardinalita e stato usato nello sviluppo di un algoritmo "mateuristico" per calcolare i limiti della Buffer Allocation. Un programma scritto in Matlab genera pattern di guasti che costitu-

iscono l'input del modello di ottimizzazione, sempre rispettando la cardinalita del set di parametri incerti. L'algoritmo e una "mateuristica" di tipo Tabu Search, e permette di cercare quelle combinazioni di guasti che piu impattano sulla funzione obiettivo. I tempi computazionali sono promettenti per lo sviluppo futuro che punti all'applicazione su casi reali.

Parole chiave: ottimizzazione robusta; Machine Loading; Buffer Allocation; Part Type Selection; approccio Cardinality-Constrained; Tabu Search; Mateuristiche.

Chapter 1

Introduction

Uncertainty marks several aspects of our lives, and it affects every decision we make, from the simplest ones such as deciding what to wear in the morning to more complex ones like designing a factory. There are no doubts that, with the increasing complexity of nowadays systems, where the events we observe can be the results of the combinations of numerous causes, uncertain events have a much bigger playground than before and the consequences of not considering it are not sustainable. The popularity of the book *"The Black Swan"* by Nassim Taleb remarks that the study of how to properly manage uncertainty has become an issue. People are interested in knowing how to cope with unexpected events, and most of the tools traditionally used are often too complicated and not very effective.

In the manufacturing field, we deal with systems like factories, production lines, machines, as well as with people. Despite society commonly envisions factories as the quintessence of clock-like accuracy, even the most meticulously planned production line can get blind-sided by profit-devouring backups. During their lifetime, these systems encounter different changes in the product specifications or in the production plan, and they face disruptive events, such as failures. Moreover, most of the systems' parameters are subject to natural uncertainty. For ex-

ample, if we imagine a production line during its lifetime, we can expect it will incur in many different scenarios: there will be common situations, like failures occurring regularly and not having a big impact on the system performance, as well as exceptional situations, such as failures requiring longer than usual to be repaired. Indeed, machine repair times can vary a lot and idle time has a major impact on the system performance.

In 1993 Shapiro had already introduced stochastic models for taking account of uncertainty in production planning [1]. Nevertheless, at that time, he also discouraged the introduction of stochastic models pointing out that production managers had just begun to use deterministic models. After 2 decades, uncertainty is often not taken into consideration by production planners, as reminded by Graves in 2011 [2]:

> *"Most systems for production planning do not recognize or account for uncertainty. Yet these systems are implemented in uncertain contexts."*

The reason for the models handling uncertainty have not yet been broadly applied probably stands in the complexity of those models: this may be the biggest obstacle to overcome before seeing their application to solve real issues.

When a system faces an uncertain environment – due to uncertainty coming from the outside or from the inside – there are commonly two main ways to react: flexibility and robustness (figure 1.1).

In the literature there is a small degree of misunderstanding when discussing the differences between robustness and flexibility. Often, this is due to the fact that several definitions of flexibility have been proposed during the last two decades [3]. Moreover, some authors consider robustness as one of the several forms of flexibility. In this thesis we will not consider aspects related to the flexibility of a system: therefore, let us just provide the definition of robustness that fits our scope. In this work, we

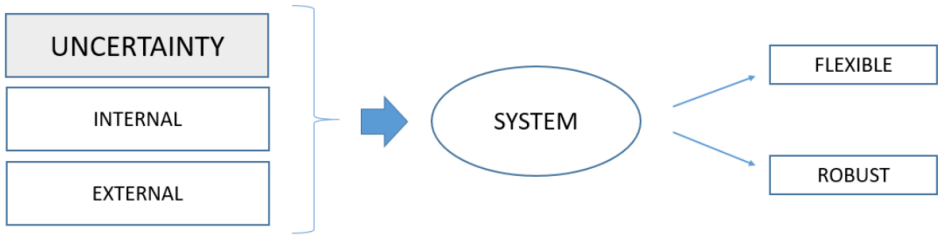


Figure 1.1: Uncertainty and the possible reactions of a system

consider a system *robust* when it possesses the ability to withstand as many types of situations as possible during its lifetime, maintaining a satisfactory output level even if some parameters changed to an off-design condition.

The main contribution of this thesis is to provide robust models for dealing with some of the main problems in the manufacturing field: (1) the Part Type Selection Problem, (2) the Machine Loading Problem, and (3) the Buffer Allocation Problem (BAP). Our goal is to cope with internal uncertainty (e.g. machine failures or uncertain processing times). In particular, we integrate Mathematical Programming models with one of the simplest, yet very effective, Robust Optimization methods that has been proposed in the literature: the Cardinality-Constrained approach. The choice of this method derives from the observation that most of the existing methods with the same scope often lack in applicability. In particular, other methods often require the knowledge of the probability distributions of the system parameters when, unfortunately, they are either unknown or require considerable time to be analysed. In other cases, the methods are too conservative, since they provide robust solutions that cover from the worst case scenarios, which are in most occasions very unlikely to take place.

The Cardinality-Constraint approach, introduced in 2004 by

Bertsimas and Sim [4], differs from these previous contributions. Their approach is very interesting because of its degree of applicability to real-life situations. Further, its basic idea can be easily understood without a deep background in operations research. In fact, it only requires one value as input: with the latter, a decision maker is able to tune the level of robustness he/she desires to achieve for the specific application. The approach takes advantage of the fact that the number of disruptive events that affect a system is going to be available in most of real cases, or at least it is easy to estimate. Moreover, a number is very easy to remember. For example, the approach can be based on how many failures occurred in the last month, or on how many work-pieces were off specifications, or, in general, on all those events that jeopardize the production correct behavior. Once chosen the cardinality (number) of unfortunate events to be covered from, the method finds the optimal solution when any possible combination of that number of events takes place. Therefore, the solutions obtained with the Cardinality-Constrained method are robust according to a certain cardinality. In other words, the method protects from the "*budget of unluckiness*" the decision maker decides to keep within.

The combined models proposed in this thesis are meant to support people working in manufacturing in decision making and system design. In this work, each model is presented together with a discussion on its applicability to real situations, and this is supported by numerical examples. The thesis is organized as follows:

- Chapter 2 outlines the the main optimization models that can be found in the literature to handle parameters uncertainty: Stochastic Programming is briefly introduced, whereas more attention is given to Robust Optimization techniques, since the Cardinality-Constrained approach be-

longs to this field.

- Chapters 3 and 4 consider the Part Type Selection Model. The problem focuses on the selection of which part types to include in a batch, satisfying both capacity and tool availability constraints. In chapter 3 is presented a deterministic version of the problem that has been chosen from the literature. Moreover, its robust counterpart is proposed, as well as results of its application to small test cases. In this analysis, we consider the uncertainty affecting the processing times for machining the workpieces. Chapter 4 presents the application of the robust counterpart of the Part Type Selection Model to a real case, by showing how a robust batch can be computed.

- Chapter 5 considers the Machine Loading Model and proposes a robust formulation for a problem chosen from the literature. Also in this case, both the deterministic and the robust formulations are shown. The robust counterpart is tested on some case studies, and results are available in this chapter

- Chapters 6 and 7 deal with the Buffer Allocation Problem. The BAP is a well-known problem that aims at configuring the buffer space along a production line. In chapter 6 we give a short literature review of the problem. Then, we attempt to keep within the analytical formulations of the Cardinality-Constrained approach in order to build a robust formulation of the BAP. The main limitations for the application of the method are indicated. Chapter 7 proposes a Tabu Search algorithm to evaluate the bounds of the BAP, in particular the upper bound. The proposed matheuristic is applied to a test case, and the results are presented.

- Conclusions are drawn in chapter 8 where we also discuss about the future developments of this work.

Chapter 2

Optimization models under uncertainty of data: literature review

In this chapter the literature concerning optimization models under uncertainty will be reviewed, with particular focus on robust optimization, which is the main topic of this work. Section 2.1 introduces the main methods for optimization under uncertainty. In section 2.2 the model for data uncertainty that will be taken as reference in this work is presented. Sections 2.3 and 2.4 give an overview over two of the main robust optimization methods that have been proposed in the literature before the publication of Bertsimas and Sim, whose method is presented in section 2.5, and its applications are shown in section 2.7.

2.1 Methods

In many real-world optimization problem, uncertainty of data is a delicate issue. Uncertainty causes model parameters to be subject to fluctuations, thus affecting the optimal solution. When an optimization problem is solved under the uncertainty of data, two main goals are typically pursued:

- **Feasibility.** To keep the solution feasible when it is applied

to the actual realizations of parameters. As data change, the feasibility range of the problem may vary so much that even former optimal solutions may be excluded. Thus, the solution will have to change accordingly to the new feasibility set.

- **Optimality.** To keep a good value of the objective function. A change in the parameters may result to force the objective function to be lower than the maximal case: the result is what is called an excessively *conservative solution*, and happens when the degree of protection against uncertainty is excessive, or when the method is not efficient enough. This reduction in the objective function must therefore be minimized.

The ability to handle parameter uncertainty has been widely studied in the literature. The main approaches that have been proposed to face uncertain data in optimization problems belong to two main groups: Stochastic Programming (SP) and Robust Optimization (RO). In the rest of this section these two methods, with their advantages and drawbacks, will be analysed, with particular focus on robust optimization, which is the main topic of this work.

2.1.1 Stochastic Programming

Stochastic Programming is based on the assumption that the fluctuation of model parameters is governed by their probability distribution. By identifying scenarios that correctly represent the outcome of the system, one can foresee the possible realizations of the parameters. Each scenario is supposed to have a probability of occurrence, assumed to be known, so all together the scenarios would build a probability distribution. We could graphically represent the Stochastic Programming ap-

proach with figure 2.1, where scenarios are combinations of parameters a_1 and a_2, and the circles' area represent the probability of each scenario's realization.

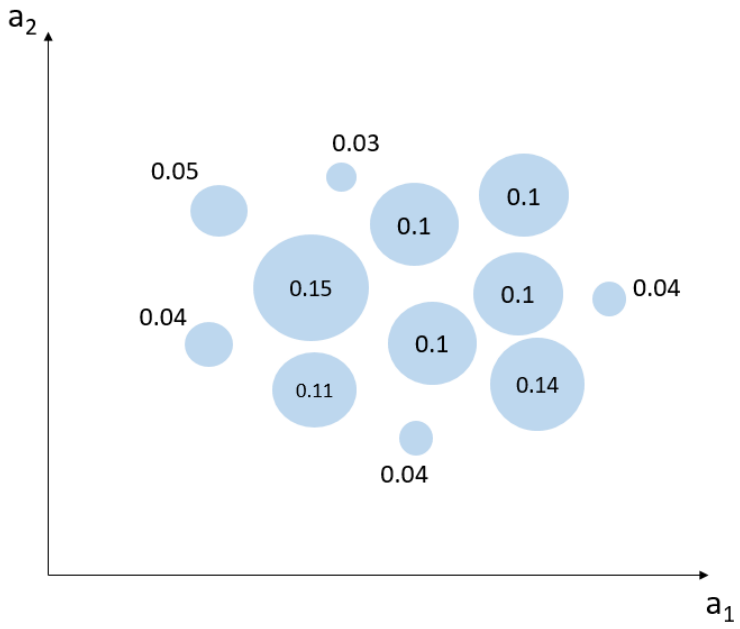


Figure 2.1: Scenarios in Stochastic Programming: here, the scenarios are represented as realizations of the 2 parameters a_1 and a_2, and the circles extension represents their probability of taking place; the numbers next to each circle represents the probability of each scenario's realization.

One of the advantages of this method is that it permits to model statistical dependency of parameters, which could be very useful since dependence occurs very often in reality. In addition, Stochastic Programming is usually generating solutions that are not over-conservative, as they protect against the most likely realizations. However, it has some drawbacks. The main disadvantage is probably that it assumes to have the correct information regarding probability distributions, which in most real applications is not available. Moreover, identifying the scenarios and their probabilities may prove to be a complex and time-consuming task. If the knowledge about scenarios is not a reliable description of the problem, the solutions may not be

robust. Hence, it may be necessary to model the system with a great number of scenarios with the consequent requirement of considerable computational effort. As far as this work is concerned, this is the main reason why we did not decide to adopt this approach. In our setting, in fact, we suppose that a reliable estimate of the probability distribution is not available. Since we want to use an approach that could be extended as much as possible to real situations, we think that requiring the assumptions we have just mentioned could be a considerable obstacle to our purpose. Since this work does not cope with Stochastic Programming, we refer the reader to [5].

2.1.2 Robust Optimization

Robust optimization is a relatively recent technique with which the decision-maker builds a solution that is feasible for any realization of the uncertainty in a given set, thus thinking of model parameters not as stochastic, but deterministic. In fact, it assumes the uncertain parameters to belong to a given set of values, and all the region of their possible realizations is supposed to be known. The goal is to construct a solution that is feasible for any realization inside the considered set.

In contrast with figure 2.1, the graphical representation of Robust Optimization is shown in figure 2.2: the scenarios are now replaced by a search area, that represents the set of values the parameters are assumed to belong. One of the points belonging to this area will consequently be the combination of the parameters that causes the worst impact on the system.

Bertsimas et al. in [6] discuss about three main issues regarding RO:

- **Tractability.** Many well known classes of optimization problems may have a RO formulation that is tractable[1];

[1]The authors define *tractable* those problems that can be reformulated into equivalent

Figure 2.2: Robust Optimization: the area in this graph represents the set of value the parameters a_1 and a_2 are assumed to belong to. The star identifies the combination of those parameters that causes the worst impact on the system performance.

nevertheless, in general the robust version of a tractable optimization problem may not itself be tractable. The authors warn that, in order to preserve tractability, care must be taken in the choice of the uncertainty sets.

- **Conservativeness.** Robust Optimization produces solutions that are deterministically immune to realizations of the uncertain parameters in certain sets: the approach proves to be reasonable when parameter uncertainty is not stochastic, or when the distributional information is not available. Nevertheless, with a proper parameterization of different classes of uncertainty sets, the designer is able to choose the tradeoff between robustness and performance, and the level of probabilistic protection. What follows is that even though Robust Optimization is inherently protecting against worst case scenarios, the solution it pro-

problems for which there are known solution algorithms with worst-case running time polynomial in a properly defined input size

duces is not over conservative and in many cases is very similar to those produced by stochastic methods.

- **Flexibility.** In [6] the authors show that robust optimization can and has been applied to a wide variety of applications: a short recap of the applications fields is presented in the next section.

2.2 A model for data uncertainty

Let us then start by defining the nominal formulation of a linear problem.

$$\begin{aligned} \max \quad & \mathbf{c'\,x} \\ s.t. \quad & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{2.1}$$

where **c'** is the vector containing the profits, **A** is the matrix of the cost coefficients, **b** is the known terms vector. **l** and **u** are vectors bounding the decision variables **x**. In this problem, it is assumed that data uncertainty will only affect the elements of matrix A. This does not influence the generality of the model, since it is always possible to model the uncertainty of the objective function coefficients in the vector **c** by writing the same problem in the equivalent form:

$$\max \quad \mathbf{z} \tag{2.2}$$

$$s.t \quad \mathbf{z} - \mathbf{c}'\mathbf{x} \leq \mathbf{0} \tag{2.3}$$

and including the constraint (2.3) in $A\mathbf{x} \leq \mathbf{b}$. For each row i of

the matrix A we define J_i as the subset of coefficients $\{a_{ij}\}_{j \in J_i}$ along the row that are subject to uncertainty. In general, robust optimization models assume that systems' parameters belong to a generic uncertainty set. In most cases, this set is a range of values in the form $[a_{lower}, a_{upper}]$. Since all the models we present in this work consider uncertainty sets as intervals of values, from now on we will always refer to this case. Each coefficient a_{ij} is modeled as a continuous, symmetric, random variable $\tilde{a}_{ij}$ that takes values in the interval $[\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij} + \hat{a}_{ij}]$ where $\bar{a}_{ij}$ is the nominal value and $\hat{a}_{ij}$ is the maximum variation from it. Each realization of the parameter $a_{i,j}$ can be derived from:

$$\tilde{a}_{ij} = \bar{a}_{ij} + \hat{a}_{ij}\xi_{ij} \tag{2.4}$$

where the variables ξ_{ij} are assumed to be simmetrically distributed in the interval $[-1, 1]$.

2.3 Soyster's approach

The first main contribution to Robust Optimization has been given by Soyster [7]. The author proposed a linear optimization model, the solution of which is feasible for all data that belong to a convex set of values. According to Soyster's formulation, the robust counterpart of the nominal problem (2.1) is as follows:

$$\max \quad \mathbf{c'x} \tag{2.5}$$

$$s.t. \quad \sum_{j \in J_i} a_{ij}x_j + \sum_{j \in J_i} \hat{a}_{ij}y_j \leq b_i \quad \forall i \tag{2.6}$$

$$-y_j \leq x_j \leq y_j \quad \forall j \in J_i \tag{2.7}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{2.8}$$

$$\mathbf{y} \geq \mathbf{0} \tag{2.9}$$

where the variables y_j have been introduced since also a negative

variation of x_j could have an impact on the system. Thus the biggest variation due to the uncertain parameter a_{ij} is given by $\hat{a}_{ij}\,|x_j|$. Even though this method successfully addresses the uncertainty of the parameters by guaranteeing the feasibility of the solution for any of their realizations (constraint (2.6)), it excessively reduces the feasibility set of the nominal problem: as a result, this approach is providing too conservative solutions. Indeed, the produced solutions show a significant reduction in the objective function.

2.4 Ben-Tal and Nemirovsky approach

Ben-Tal and Nemirovsky [8] have proposed a method that aims to solve the conservatism found in Soyster's solution. Their method guarantees the feasability of the solution only if, for each row i, the data $\{\tilde{a}_{ij}\}_{j\in J_i}$ lie within an ellipsoidal set E_i. The authors proposed the following robust linear optimization problem:

$$\max \quad \mathbf{c'\,x} \tag{2.10}$$

$$s.t. \quad \sum_j a_{ij}x_j + \sum_{j\in J_i} \hat{a}_{ij}y_{ij} + \Omega_i\sqrt{\sum_{j\in J_i} \hat{a}_{ij}^2 z_{ij}^2} \leq b_i \quad \forall i \tag{2.11}$$

$$-y_{ij} \leq x_j - z_{ij} \leq y_{ij} \quad \forall i, j \in J_i \tag{2.12}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{2.13}$$

$$\mathbf{y} \geq \mathbf{0}$$

For each row i, the width of each ellipsoid E_i is defined by the parameter Ω_i which controls the feasibility region of the nominal problem for the i-th constraint. These parameters can – and must – be properly tuned in order to obtain the desired level of robustness of the solution. The robust model of Ben-Tal

and Nemirovsky is less conservative than the Soyster approach, because every feasible solution of the latter problem is a feasible solution to the former. The authors prove that, under the model of uncertainty they propose, the probability of violation of the i-th constraint is at most $\left(\exp -\Omega_i^2/2\right)$. Moreover, they show possible applications of their approach to well-known problems. However, this approach has the main disadvantage of directing to conic quadratic problems, which even though convex are non-linear [8]. Therefore, from a computational point of view, the robust counterpart turns out to be much more demanding than the nominal problem, and this makes the approach not very suitable to large-size discrete optimization problems.

2.5 The Cardinality-Constrained approach

Bertsimas and Sim have developed an approach that is an essential contribution in the field of Robust Optimization. In [4] they introduce a new model that is able to solve the over-conservatism of Soyster's formulation. Moreover, differently from the previous robust approaches, they consider unlikely the scenario in which all the parameters $\{\tilde{a}_{ij}\}_{j\in J_i}$ on a generic row i take their worst values at the same time as very unlikely to happen. Therefore, they introduce the parameter Γ_i that is the number of entries a_{ij} from the i-th row that are expected to go to their maximum values; Γ_i is set in the continuous range $[J_i]$. The goal of their method is to find a solution that is feasible and covered against all the possible outcomes in which up to $\lfloor \Gamma_i \rfloor$ parameters change from their nominal values (where $\lfloor\ \rfloor$ indicates the lower integer of a real number), and one coefficient $\tilde{a}_{it}$ changes by $(\Gamma_i - \lfloor \Gamma_i \rfloor)\,\hat{a}_{it}$.

Refering to the model presented in section 2.3, the robust formulation is as follows:

$$\max \quad \mathbf{c}'\mathbf{x} \tag{2.14}$$

$$s.t. \quad \sum_j a_{ij}x_j + \max_{\Omega} \left\{ \sum_{j \in S_i} \hat{a}_{ij}y_j + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it}y_{t_i} \right\} \leq b_i \, \forall i \tag{2.15}$$

$$-y_j \leq x_j \leq y_j \quad \forall j \in J_i \tag{2.16}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{2.17}$$

$$\mathbf{y} \geq \mathbf{0} \tag{2.18}$$

where the subset Ω is defined as:

$$\Omega = \{S_i \cup \{t_i\};\, S_i \subseteq J_i;\, |S_i| = \lfloor \Gamma_i \rfloor;\, t_i \in J_i \setminus S_i\} \tag{2.19}$$

Let us define the second term on the left side of equation (2.15):

$$\beta_i(\mathbf{x}, \Gamma_i) = \max_{\Omega} \left\{ \sum_{j \in S_i} \hat{a}_{ij}y_j + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it}y_t \right\} \tag{2.20}$$

as the protection function on the i-th constraint. It is straightforward that when $\Gamma_i = 0$, $\beta_i = 0$, since in this case the set S_i is empty and the second term of the sum in equation (2.20) is equal to 0: with no protection the problem coincides with the nominal one (2.1). On the other hand, when $\Gamma_i = |J_i|$, we have the maximum level of protection on the i-th row, so we are in the Soyster's framework. It is already possible to notice that with the Cardinality-Constrained formulation the decision maker is able to "tune" the level of robustness by setting the parameter Γ_i, depending on the number of parameters per row which he/she expect to fluctuate.

It is now possible to show that problem (2.14)-(2.18) admits an equivalent linear formulation. Given a vector $|x^*|$ the protection function $\beta_i(x, \Gamma_i)$ of the i-th constraint is equal to the objective function of the following problem:

$$\beta_i(|x^*|, \Gamma_i) = \max \quad \sum_{j \in J_i} \hat{a}_{ij}|x^*|z_{ij} \tag{2.21}$$

$$s.t. \quad \sum_{j \in J_i} z_{ij} \leq \Gamma_i \tag{2.22}$$

$$0 \leq z_{ij} \leq 1 \quad \forall j \in J_i \tag{2.23}$$

In fact, the solution of the problem above is made by $\lfloor \Gamma_i \rfloor$ variables equal to 1 and one variable equal to $\Gamma_i - \lfloor \Gamma_i \rfloor$. This corresponds to selecting the subsets S_i and $\{t_i\}$. Consider now the dual of the previous problem:

$$\min \quad \sum_{j \in J_i} p_{ij} + \Gamma_i z_i \tag{2.24}$$

$$s.t. \quad z_i + p_{ij} \geq \hat{a}_{ij}|x_j^*| \quad \forall j \in J_i \tag{2.25}$$

$$z_i \geq 0 \tag{2.26}$$

$$p_{ij} \geq 0 \quad \forall j \in J_i \tag{2.27}$$

The robust approach of Bertsimas and Sim starts from noticing that since the dual is a feasible and bounded linear problem, the Strong Duality Theorem holds; so, at optimum, the objective function of the primal and the dual problem will coincide. Let us notice again the equation (2.15), and write it in the form:

$$\sum_j a_{ij}x_j + \beta_i \leq b_i \quad \forall i \tag{2.28}$$

Since we are considering a maximization problem, the term β_i will be implicitly minimized by satisfying the constraint (2.15).

Therefore, it is possible to substitute the dual problem in the Master Problem formulation (2.14)-(2.18), thus obtaining:

$$\max \quad \mathbf{c'x} \tag{2.29}$$

$$s.t. \quad \sum_j a_{ij}x_j + \sum_{j\in J_i} p_{ij} + \Gamma_i z_i \leq b_i \quad \forall i \tag{2.30}$$

$$z_i + p_{ij} \geq \hat{a}_{ij} y_j \quad \forall i, j \in J_i \tag{2.31}$$

$$z_i \geq 0 \quad \forall i \tag{2.32}$$

$$p_{ij} \geq 0 \quad \forall i, j \in J_i \tag{2.33}$$

$$-y_j \leq x_j \leq y_j \quad \forall j \in J_i \tag{2.34}$$

$$y_j \geq 0 \quad \forall j \in J \tag{2.35}$$

$$\mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \tag{2.36}$$

In this way, the dual problem will be implicitly solved in the Master Problem, thus finding the maximum value of the protection function β_i. Hence, the optimization is solved in the worst case scenario constrained to the cardinality Γ_i: indeed, it will be selected the combination of parameters going to their maximum value that – respecting the cardinality – would cause the worst impact on the system.

Bertsimas and Sim have developed an approach that stands out among other robust models. We can identify the main reasons for the success of the methods:

- The approach is able to preserve the linear form of the problem.
- It models the uncertainty of the parameters in a very simple way, in fact only 2 coefficients are required: the cardinality, and the maximum variation of the parameters.
- It does not require a knowledge in the probability distributions of the parameters.

- It offers full control on the conservatism of the solution.
- It is intuitive: the designer is asked to estimate a very simple parameter. This is very important when the applicability of this approach is concerned.
- It guarantees deterministic feasibility for any cardinality value smaller than the considered one.

It is worth to notice that the assumption of symmetry on the probability distribution of the uncertain parameters plays an important role. Relying on this hypothesis, in [4] upper bounds on the probability of violation of each i-th constraint are derived; it is showed that a valid upper bound is:

$$\exp\left(-\frac{\Gamma_i^2}{2|J_i|}\right) \tag{2.37}$$

and the authors also show that tighter bounds can be found: intuitively, since the probability distribution is assumed to be symmetric, each parameter has the same probability of assuming the value $\bar{a}_{ij} - \hat{a}_{ij}\xi_{ij}$ or $\bar{a}_{ij} + \hat{a}_{ij}\xi_{ij}$, where $\xi_{ij} \in [-1, 1]$. Therefore, the random deviations in the constraint matrix tend to compensate each other when they are summed, especially when the number of uncertain parameters grows.

2.6 Applications of Robust Optimization

In [6] the authors present a review of the applications approached by Robust Optimization techniques:
Portfolio Optimization. It is well-known that a main issue in finance is how to allocate monetary resources across risky assets. Robust Optimization has been widely applied in this field, and in the literature it is possible to find a considerable number of

models: among all, the authors pose particular attention to [9] and [10].

Statistics, Learning and Estimation. Since the process of analyzing data and describing the parameters of a system is naturally uncertain, it is not surprising that those problems have been approached from a Robust Optimization perspective. That has been done by [11] and [12].

Supply chain management. Bertsimas and Thiele [13] consider a robust model for inventory control, using a cardinality-constrained uncertainty set, as shown in section 2.5. The authors claim as advantage that the robust approach provides inventory control policies that are structurally identical to the stochastic case, with the added advantage that probability distributions need not to be assumed in the robust case.

Engineering. Several papers on robust engineering design problems include applications in:

- Structural design [14].
- Circuit design: see [15] and [16].
- Power control in wireless channels [17].
- Antenna design: [18]; [19].
- Control [20].
- Simulation-based optimization in engineering [21].

2.7 Applications of the Cardinality-Constrained approach

Several applications of the Cardinality-Constrained approach can be found in the literature: a few of them which regard problems in the manufacturing field will be shortly presented. The

main publications of robust approaches as applications of the Cardinality-Constrained method can be found in table 2.1.

In [22], Moreira developed a robust Assembly Line Balancing on the basis of the work of Borba and Ritt [23], who introduced a formulation in which the number of stations is minimized by minimizing the number of workers assigned to the assembly line. The authors show that the proposed model, together with a heuristic, can provide solutions considerably more robust to task time variations with a relatively low increase in the number of stations and workers needed: this can be an effective tool for assembly line managers.

Alem and Morabito [24] developed Robust Optimization tools for robust combined lot-sizing and cutting-stock models. The authors aim at protecting against uncertainty in production costs and product demand.

Hazir et al. [25] proposed a robust version of the discrete time/cost trade-off problem (DTCTP),a well-known project scheduling problem with practical relevance in which the total cost of a project is minimized, constrained to the project deadline, or viceversa. The authors address the deadline version of the problem and consider the uncertainty of the costs of the activities.

Lu et al. [26] introduced a robust single machine job scheduling problem, in which they cover from uncertainty of processing times.

Solyali et al. [27] developed a robust approach of the inventory routing problem. The issue for the supplier is to determine the delivery quantities of a single product as well as the times and routes to the multiple customers facing uncertain demands over a finite time horizon.

Table 2.1: Main Publications regarding applications of the cardinality-constrained approach

Authors	Topic	Uncertainty sets	Reference
Moreira et al.	Assembly line balancing	Workers' task execution times	[22]
Alem & Morabito	Production planning	Production costs, Demand	[24]
Bertsimas & Thiele	Supply chain management	Product demand	[13]
Hazir & Dolgui	Assembly line balancing	Operation times	[28]
Hazir et al.	Project scheduling	Cost uncertainty	[25]
Moon and Yao	Portfolio optimization	Return rate of assets	[29]
Lu et al.	Machine scheduling	Processing times	[26]
Solyali et al.	Inventory routing	Product demand	[27]

Chapter 3

Part Type Selection Problem

This chapter provides a robust formulation of a well-known problem in the manufacturing field. The intention is to show the applicability of the robust counterpart of a Part Type Selection model to the Production Planning, as well as its simplicity, since a small number of input parameters are required. The robust model has been tested on small case studies, which motivated its successive application to a real case (chapter 4). Section 3.1 will introduce the problem; in section 3.2 the deterministic model proposed by Hwang and Shogan will be presented, and its robust version will be developed in section 3.3. The test studies and the numerical results of the application of the robust model will be shown in section 3.4.

3.1 Problem Description

One of the main issues in a Flexible Manufacturing Systems environment is Production Planning: the selection of which tools to load on the machines and which workpieces to produce. This problem has been analysed by many researchers, and during the last 30 years several papers regarding solution approaches have been published: among them are simulation, mathematical programming, queueing, as well as heuristics.

Flexible Manufacturing Systems

Flexible Manufacturing Systems are configurations of machines, linked by automatic transportation systems, capable of unmanned production shifts and automatic tool changing. FMSs are capital intensive systems, meaning they require a consistent investment (training, hardware, software, floor space), with the capability of producing a variety of high quality workpieces in short lead times. The flexibility of the system permits to change tools and route workpieces in such a way so to maximize the utilization of machines. FMSs played a central role in the development of the 1980's production systems and were indicated by many experts as one of the reasons for Japan's manufacturing leadership during that period [36].

Production Planning

A general Production Planning problem is divided into two subproblems: Part Type Selection and Machine Loading. Some constraints are present and must be satisfied in both of these problems, since they are closely interrelated; those are:

- Given a selected set of parts, the set of tools required to process them is determined through the part-tool incidence matrix. Each tool requires a certain number of tool slots. The total slots needed shall not exceed the tool magazine capacity.
- Parts have their corresponding due dates: the production must be completed in a certain amount of time.
- Sometimes, the workload balance of the machines has to be taken into account too. In fact, an unbalanced Production Plan would result in some machines being over-saturated, while others under-utilized. The introduction of this constraint is not mandatory for guaranteeing the feasibility of

the solution, even though a smoother Production Plan is desirable.

In this chapter we focus on the Part Type Selection Problem, whereas we deal with the Machine Loading Problem in chapter 5.

Part Type Selection

The Part Type Selection Problem is the optimization problem selecting the composition of the batch that maximizes an objective function (typically expressed in profit terms) over a considered amount of time. In particular, the problem deals with the minimization of the setup time, that is the time required to switch from a product type to another. Since setup time is usually constant, minimizing it corresponds to minimize the number of setups, or, equivalently, minimizing the number of batches.

Hwang and Shogan [37] simplified the objective of the Part Type Selection Problem by translating it into the maximization of the number of part types included in successive batches: indeed, the more part types can fit in each batch, the less batches will be required. Starting from a general order (a pool of items demanded by the customer), the authors propose to select only a first batch, the size of which will be maximized by the optimization model. Consequently, the problem is solved again to select a second batch from the remaining part of the order, and so on until the order is completed. In this way, the model keeps flexible enough to cope with incoming orders, with machine breakdowns and with other dynamic conditions. Moreover, maximizing the number of selected part types permits to achieve high utilization of the tool storage capacity.

3.2 The model of Hwang and Shogan

The model of Hwang and Shogan [37] has been chosen for this work because it is simple enough to take account of the main properties of the results, and at the same time it does not lack of significance and applicability to real problems. In their work, the authors consider parts as tuples characterized by three attributes:(1) part type, (2) order quantity, (3) due date. Tools may perform one or more operations on each parts, and each of them could occupy a different number of slots on the machines.

3.2.1 Assumptions

The set of the parts that can be included in the batch is the production order: therefore, a batch is a subset of the order that can be processed simultaneously by the system. The following assumptions are considered:

1. Setups are done only once the batch has been completed.
2. The tool storage capacity is limited.
3. All the necessary tools for processing each part order are assumed to be in the system before it starts operations.
4. Each part has one and only one corresponding set of tools.
5. The machines are general purpose, meaning they all can be configured with the tools necessary to produce the parts composing the batch. Hence, it is possible to treat the whole system as a single machine. Notice that this assumption does not properly represent the practice: indeed, there will be a successive assignment of tools to the machines, which is the solution to the Machine Loading models (see chapter 5).

3.2.2 Part Type Selection Model

The following is the formulation of the Part Type Selection model as in [37]:

$$\max \quad \sum_i w_i y_i \tag{3.1}$$

$$s.t. \quad \sum_j c_j x_j \leq S \tag{3.2}$$

$$a_{i,j}\, y_i \leq x_j \quad \forall i, j \tag{3.3}$$

$$\sum_i \left(\sum_j t_{i,j} q_i \right) y_i \leq M \tag{3.4}$$

$$y_i \in \{0, 1\} \tag{3.5}$$

$$x_j \in \{0, 1\} \tag{3.6}$$

Decision variables. y_i is 1 when part type i is inserted in the batch, 0 otherwise; x_j is 1 if tool type j is required by the system, 0 otherwise.

Parameters. a_{ij} is 1 if part type i requires tool j; c_j is the number of tool magazine slots occupied by tool j; S is the total number of tool slots in the system; w_i is the weight of the i-th part order (the authors propose a direct relationship with the due date of the part order); t_{ij} is the processing time of tool j required by the product type i in the order; q_i is the order quantity; M is the upper bound on the total machining time for the batch, that reflects the time available for production (for example, a shift).

Objective function. The objective function (3.1) aims at maximizing the number of items inserted in a batch. The authors of [37] propose a relationship between the weights and the due date of a part order such as $w_i = \frac{1}{d_i}$, where d_i represents the time until part order i is due to be delivered. Nevertheless, other factors

may be taken into consideration (profit, late delivery penalties, importance of the customer, etc.).
Constraints. Constraint (3.2) ensures that the tool magazine capacity is satisfied; constraint (3.3) guarantees that for each part type selected, all the necessary tools to process it are selected; constraint (3.4) ensures that the total processing time needed does not exceed the total machining time available, and consequently avoids the possibility of some orders not respecting their due date. Constraints (3.5) and (3.6) are assignment constraints that state the boolean nature of the decision variables.

3.3 Robust formulation

In a production environment, any disruption causing an elongation of the operations to process a part could compromise the possibility of meeting customers expectations in terms of lead time. In fact, the model proposed by Hwang and Shogan explicitly considers the trade-off between large batches and order urgency. For this reason, it is desirable to build a robust model that covers from unexpected events that could happen during the production: these events can be, for example, sudden tool breaks, unplanned extra-ordinary maintenance, as well as incorrect positioning of work-pieces that could block the machines for a considerable amount of time, and so on.

Building a robust Part Type Selection Model by means of the Cardinality-Constrained approach has the scope to find how to include part types in the batches of a production plan, the execution of which is supposed to be affected by a certain number of disruptions. Among all the combinations of those unfortunate events, the ones causing the worst impact will be selected, so that any other combination of that number of events will guarantee the optimal solution (for example, over the production of 100 part types, we expect 2 of them will be affected by failures

as their process will require more time: among all the possible combinations, the 2 part types causing the worst impact on the system in case of failures will be selected for the solution of the model).

3.3.1 Model of data uncertainty

Let us then consider the processing times t_{ij}. Following the same considerations done in section 2.2, we can model the processing times as random variables $\tilde{t}_{ij}$, symmetrically distributed in the interval $\left[\bar{t}_{ij} - \hat{t}_{ij}, \bar{t}_{ij} + \hat{t}_{ij}\right]$, where $\bar{t}_{ij}$ is the expected value of the operation time on order type i using tool j, and $\hat{t}_{ij}$ is the maximum deviation from it. This model of data uncertainty – that has been presented in chapter 2 – permits to cover from any type of disruptive events that forces the operations to require more time.

It is worth to notice that despite in this formulation all the realizations of $\tilde{t}_{ij}$ are referring to the time to process an item belonging to the order type i, since constraint (3.10) includes the total order quantity q_i, by assigning a cardinality to the uncertain event we are assuming they would impact the whole batch. On the basis of the latter consideration, notice it is still easy to reasonably estimate the time deviations: for example, if we expect the batch of the i-th part type to require an additional 20% of time, then the corresponding time deviations will simply be: $\hat{t}_{ij} = 0.2\,\bar{t}_{ij}$.

3.3.2 Protection function

Let us then build a constraint on the production time budget, in order to fit in the Cardinality-Constrained framework. Then, let us split constraint (3.4) into constraints (3.9) and (3.10), thus reformulating the model as follows:

$$\max \quad \sum_i w_i y_i \tag{3.7}$$

$$s.t. \quad \sum_j c_j x_j \leq S \tag{3.8}$$

$$\sum_j T_j \leq M \tag{3.9}$$

$$\sum_i t_{i,j} q_i y_i \leq T_j \quad \forall j \tag{3.10}$$

$$a_{i,j}\, y_i \leq x_j \quad \forall i, j \tag{3.11}$$

$$y_i \in \{0, 1\} \tag{3.12}$$

$$x_j \in \{0, 1\} \tag{3.13}$$

$$T_j \in \mathbb{R} \tag{3.14}$$

where T_j is a continuous decision variable that represents a time budget on the machining time spent on the j-th tool.

It is now possible to consider the constraint (3.10) and write its robust formulation by inserting the protection function.

$$\sum_i \bar{t}_{i,j} q_i y_i + \max_{\Omega_j} \left\{ \sum_i \hat{t}_{i,j}\, q_i\, y_i + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{t}_{u_j,j}\, q_i\, y_{t_j} \right\} \leq T_j \quad \forall j \tag{3.15}$$

where

$$\Omega_j = \{S_i \cup \{u_j\};\, S_i \subseteq J_i;\, |S_i| = \lfloor \Gamma_i \rfloor;\, u_j \in J_i \setminus S_i\} \tag{3.16}$$

is the subset of values subject to uncertainty that will go to their maximum value, constrained to have cardinality Γ_j. This way, the second term of the right side of equation (3.15) represents the protection function in the event of disruptions that would cause the processing times for workpiece i on tool j to rise up to $t_{ij} + \hat{t}_{ij}$.

At this point, we can explicit the protection function and the selection of the subset S_j in the following maximization problem. Let y_i^* be a particular solution of the batching problem, then:

$$\beta_j = \max \quad \sum_i \hat{t}_{i,j}\, q_i\, y_i^*\, z_{i,j} \tag{3.17}$$

$$s.t. \quad \sum_i z_{i,j} \leq \Gamma_j \tag{3.18}$$

$$0 \leq z_{i,j} \leq 1 \quad \forall i \tag{3.19}$$

$$z_{i,j} \in \mathbb{R} \tag{3.20}$$

where z_{ij} are continuous variables in the interval $[0, 1]$, Γ_j is the cardinality parameter, representing the number of part orders i expected to require more time to be processed on the j-th tool. The dual form of this maximization problem is:

$$\beta_j = \min \quad \Gamma_j\, z_j + \sum_i p_{i,j} \tag{3.21}$$

$$s.t. \quad z_j + p_{i,j} \geq \hat{t}_{i,j}\, q_i\, y_i^* \quad \forall i \tag{3.22}$$

$$z_j \in \mathbb{R}^+ \tag{3.23}$$

$$p_{i,j} \in \mathbb{R}^+ \tag{3.24}$$

At this point, the dual problem can be inserted in the master one:

$$\max \quad \sum_i w_i y_i \tag{3.25}$$

$$s.t. \quad \sum_j c_j x_j \leq S \tag{3.26}$$

$$\sum_j T_j \leq M \tag{3.27}$$

$$\sum_i t_{i,j} q_i y_i + \Gamma_j \, z_j + \sum_i p_{i,j} \leq T_j \quad \forall j \tag{3.28}$$

$$z_j + p_{i,j} \geq \hat{t}_{i,j} \, q_i \, y_i \quad \forall i, j \tag{3.29}$$

$$a_{i,j} \, y_i \leq x_j \quad \forall i, j \tag{3.30}$$

$$y_i \in \{0, 1\} \tag{3.31}$$

$$x_j \in \{0, 1\} \tag{3.32}$$

$$z_j \in \mathbb{R}^+ \tag{3.33}$$

$$p_{i,j} \in \mathbb{R}^+ \tag{3.34}$$

With this formulation, the dual problem will be implicitly solved in the maximization problem, and the solution will be the batch covering from the expected unfortunate events.

3.4 Test cases

In this section the application of the robust version of the Part Type Selection Model to some test cases will be presented. The aim is to show the general behavior of the model, in terms of objective function and batch patterns, when the main parameters vary. This preliminary comprehension is useful for the consequent application of the model to real case studies (Chapter 5).

3.4.1 Data and assumptions

- Let us consider a problem with 10 part types and 10 tool types. This is a smaller problem than a real-life situation, but not trivial.

- Each workpiece requires 2 to 5 tools to complete its production. This corresponds to a common FMS production environment. So, let the matrix of coefficients a_{ij} be:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.35}$$

- The processing time for each workpiece has been set to the arbitrary value of $1\,min$. This is to keep the formulation simple, as well as the interpretation of the results. So, the matrix of coefficients $\bar{t}_{ij}$ will be:

$$\bar{\mathbf{T}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.36}$$

Notice that, numerically, $t_{ij} = a_{ij}$.

- Production time deviations $\hat{t}_{ij}$ are chosen equal to the nominal production time $\bar{t}_{ij}$. Thus, a disruptive event causes the operation time for a workpiece to double its value, since $\bar{t}_{ij} + \hat{t}_{ij} = 2\,\bar{t}_{ij}$. This choice is still arbitrary in a test case, but in a real case this has to be carefully chosen by the decision maker, in order to correctly represent the real situation. The time deviations have to be selected from reasonably expected realizations of the processing times.

- One production week has been chosen as planning horizon. This is mainly due to the small size of the problem, that makes the choice of longer time horizon not very meaningful in this case. Considering a 5-day week with one 8-hour shift per day, then:

$$M = 5\,days{\cdot}8\,hours/day{\cdot}60\,min/hour = 2400\,min \tag{3.37}$$

- The number of slots required by each tool are chosen so to resemble a real situation. Normally, tools require between 1 to 10 slots, so let the values c_j be:

$$\mathbf{c} = \begin{bmatrix} 1 & 1 & 2 & 2 & 2 & 2 & 5 & 5 & 7 & 10 \end{bmatrix} \tag{3.38}$$

- The value of the tool slot capacity S has initially been chosen in order to not be the critical resource ($S = 37$).

  This choice is mainly due to the fact that we are interested in capturing the behavior of the model when affected by uncertainty in the processing times: the lack of tool slots is an issue that involves the deterministic version of the problem, the solution of which is often trivial.

- Two different order quantity vectors have been considered in the analysis. (1) diverse values for each part type demand, $q_i^{(1)}$, and (2) constant order quantities $q_i^{(2)}$. So:

$$\mathbf{q}^{(1)} = \begin{bmatrix} 200 & 200 & 100 & 100 & 100 & 100 & 100 & 50 & 50 & 10 \end{bmatrix} \tag{3.39}$$

$$\mathbf{q}^{(2)} = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \end{bmatrix} \tag{3.40}$$

- The chosen cardinality values are meant to allow the identification of the behavior of the system. Given the small size of this problem, reasonable values shall not exceed 3. The cardinality values used in the experimentation are shown in table 3.1.

- Weights are chosen so to represent different types of importance of the part types: $w_i^{(1)}$ constant, $w_i^{(2)}$ linear, $w_i^{(3)}$ quadratic, $w_i^{(4)}$ random, $w_i^{(5)}$ random with 2 important workpieces. They are listed in table 3.2.

3.4.2 Results

The results obtained by applying the new robust Part Type Selection Model to the test case proposed in section 3.3 will

Table 3.1: Cardinality Values Γ_j that have been considered for conducting the test case analysis

$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
0	1	2	3	2
0	1	2	3	2
0	1	2	3	1
0	1	2	3	1
0	1	2	3	1
0	1	2	3	1
0	1	2	3	1
0	1	2	3	1
0	1	2	3	1
0	1	2	3	1

Table 3.2: Weights w_i that have been used in the test case

$w_i^{(1)}$	$w_i^{(2)}$	$w_i^{(3)}$	$w_i^{(4)}$	$w_i^{(5)}$
Constant	Linear	Quadratic	Random	2 important types
0.1	0.02	0.00	0.11	0.27
0.1	0.04	0.00	0.01	0.02
0.1	0.05	0.00	0.16	0.04
0.1	0.07	0.01	0.16	0.10
0.1	0.09	0.02	0.05	0.14
0.1	0.11	0.03	0.05	0.29
0.1	0.13	0.06	0.03	0.01
0.1	0.15	0.13	0.17	0.01
0.1	0.16	0.25	0.14	0.03
0.1	0.18	0.50	0.12	0.08

be herewith listed and interpreted. All the instances described below have been solved using IBM ILOG CPLEX ® v12.5 on a notebook Dell XPS13 with a i7 Intel Core @2.5GHz and 8GB RAM (WIN10). Table 3.3 shows the computational times for the case studies that have been completed. In all cases, the time to solve is between 5 and 6 seconds, so there is no big difference among the cases.

Table 3.3: Computational times for the application of the robust counterpart of the Part Type Selection model to test cases

	CPU time [s]				
Case	$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
CASE 1	5.91	5.22	5.45	5.34	5.20
CASE 2	5.43	5.38	5.02	5.16	5.43
CASE 3	5.47	5.53	5.31	5.15	5.54
CASE 4	5.64	5.46	5.33	5.42	5.16
CASE 5	5.14	5.47	5.28	5.26	5.13

Case 1: constant weights, diverse order quantities

In table 3.4 the batches resulting from the application of the robust Part Type Selection model are shown, in the case of part types with the same importance and with diverse order quantities. It can be easily noticed that the number of items included in the batch decreases as the cardinality Γ_j increases. Consequently, the objective function value decreases following a non-increasing pattern with Γ_j. The behavior of the objective function can be contemplated in figure 3.1.

Table 3.4: Case 1: constant weights, diverse order quantities

Part Type	w_i	$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
1	0.10	0	0	0	0	0
2	0.10	1	0	0	0	0
3	0.10	1	1	0	1	1
4	0.10	1	1	1	0	1
5	0.10	1	0	0	0	0
6	0.10	1	1	1	1	1
7	0.10	1	1	1	1	1
8	0.10	1	1	1	1	1
9	0.10	1	1	1	1	1
10	0.10	1	1	1	1	1
Objective function:		0.9	0.7	0.6	0.6	0.7

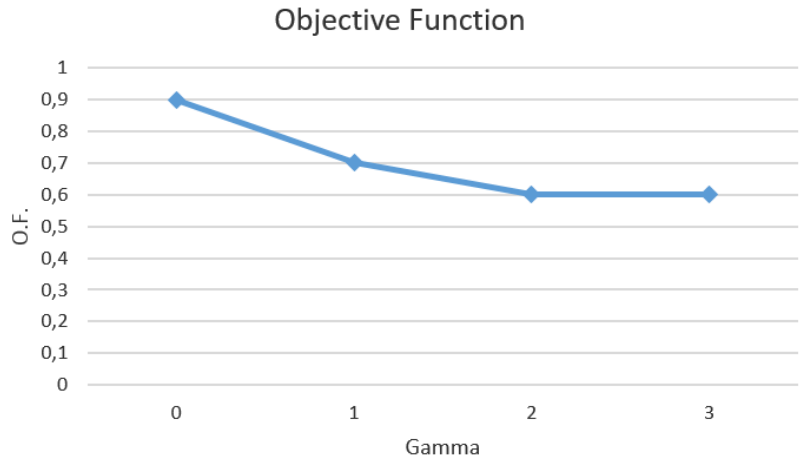


Figure 3.1: Case 1: Objective function value over Γ_j, in the case of constant weights, diverse order quantities

Case 2: linear weights, diverse order quantities

Table 3.5 shows the results obtained in the case where to part types have been assigned linear weights. The similar behaviour for the batching and for the objective function can be noticed.

Table 3.5: Case 2: linear weights, diverse order quantities

Part Type	w_i	$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
1	0.02	0	0	0	0	0
2	0.04	1	0	0	0	0
3	0.05	1	1	0	0	1
4	0.07	1	1	0	0	1
5	0.09	1	0	1	1	0
6	0.11	1	1	1	1	1
7	0.13	1	1	1	1	1
8	0.15	1	1	1	1	1
9	0.16	1	1	1	1	1
10	0.18	1	1	1	1	1
Objective function:		0.98	0.85	0.81	0.81	0.85

From the results, since part type 5 is not included in the cases with $\Gamma^{(1)}$ and $\Gamma^{(4)}$, one could argument that the batching does

not respect the importance of the part types. Nevertheless, a closer look at the data can show the reason for it. Table 3.6 reports, for each part type, its tool time τ_i: this quantity represents the total time a part type requires for the completion of the production of its order quantity; it is calculated in equation 3.41.

$$\tau_i = \sum_i q_i \, t_{ij} \quad \forall i \tag{3.41}$$

Table 3.6: Tool time τ_i for each part type

Part Type	τ_i
1	600
2	600
3	300
4	300
5	500
6	200
7	100
8	200
9	150
10	20

From this it is possible to notice that the items that have not been included in the batch are the ones requiring more time for production. The occurrence of a disruptive event has lowered the total available time, so the items that were more time consuming have been removed from the batch.

Case 3: quadratic weights, diverse order quantities

As shown in table 3.7, the case of quadratic weights presents a very similar behavior as the case with linear ones. Nevertheless, from the batching results can be noticed that the least important items have been already taken out from the batch with $\Gamma_j^{(1)}$

values; consequently, the time saved was such that it was not necessary to eliminate any other items.

Table 3.7: Case 3: quadratic weights, different order quantities

Part Type	w_i	$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
1	0.001	0	0	0	0	0
2	0.002	1	0	0	0	0
3	0.004	1	0	0	0	0
4	0.008	1	0	0	0	0
5	0.016	1	1	1	1	1
6	0.031	1	1	1	1	1
7	0.063	1	1	1	1	1
8	0.125	1	1	1	1	1
9	0.250	1	1	1	1	1
10	0.500	1	1	1	1	1
Objective function:		0.99	0.98	0.98	0.98	0.98

Case 4: random weights with two important orders, different order quantities

This case has been carried out with the intent to show that it is possible to use weights to keep some important items in the batch. Table 3.8 shows the results obtained in this case. It is possible to notice that the 2 most important part types (e.g. parts 1 and 2) are kept in the batch no matter what the cardinality values are. This is important because it proves the decision maker is still able to control the batching results depending on the requirements.

Case 5: linear weights, constant order quantities

The case of constant order quantities $q_i^{(2)}$ shows the dependence of the results from the tool time. In fact, we can notice in table 3.9 that the part types requiring the most time are not included in the batch, and the results are not influenced by the cardinality

Table 3.8: Case 4: random weights with two important orders, different order quantities

Part Type	w_i	$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
1	0.27	1	1	1	1	1
2	0.02	0	0	0	0	0
3	0.04	1	0	0	0	0
4	0.10	1	0	1	1	1
5	0.14	1	1	0	0	0
6	0.29	1	1	1	1	1
7	0.01	1	0	0	0	1
8	0.01	1	0	0	0	0
9	0.03	1	0	0	0	0
10	0.08	1	1	1	1	1
Objective function:		0.97	0.78	0.73	0.73	0.75

after $\Gamma_j = 1$, since the time saved is enough to cover from the additional disruptive events.

Table 3.9: Case 5: linear weights, constant order quantities

Part Type	w_i	$\Gamma_j^{(0)}$	$\Gamma_j^{(1)}$	$\Gamma_j^{(2)}$	$\Gamma_j^{(3)}$	$\Gamma_j^{(4)}$
1	0.02	0	0	0	0	0
2	0.04	0	0	0	0	0
3	0.05	1	0	0	0	0
4	0.07	1	0	0	0	0
5	0.09	1	0	0	0	0
6	0.11	1	1	1	1	1
7	0.13	1	1	1	1	1
8	0.15	1	1	1	1	1
9	0.16	1	1	1	1	1
10	0.18	1	1	1	1	1
Objective function:		0.94	0.72	0.72	0.72	0.72

3.5 Conclusion

A robust model for the assessment of a Part Type Selection has been developed and tested on some small cases. The results have proved to be aligned with the expected behavior of a Cardinality-Constrained robust model, such as the decrease in the objective function with an increase in the cardinality. Therefore, this model can be applied to real cases (Chapter 4) and is fit to become part of the Production Planning process. With this tool, a decision maker can evaluate the cost of disruptive events on the batch production campaign, and consequently quantify how much he/she is willing to pay in order to be covered: that is the trade-off between robustness and the price of it.

Chapter 4

Case study: robust batching in a screws production facility

In this chapter the robust counterpart of the Part Type Selection Model (Chapter 3) has been applied to a real situation. The case study that has been considered is the Production Planning for a small business of screws production located in the northern part of Italy. The company has a year revenue of 10 million € and it employs 50 people. The production facility is constantly facing delays and it struggles to meet all the customers' orders. The aim is to provide a tool that the management can use in the prioritization of the production in order to cover from the disruptive events that cause delays.

4.1 Problem Description

The production of screws is a quite simple process: iron wire is inserted in a cold heading machine that aligns the steel wire and cuts it so to shape the head of the screw. The headed parts are then cleaned in a washing tank; a threading process follows, where the desired thread is shaped on the products. The last part of the process depends on the product requirements, and it can include either a heat treatment or a carburizing process,

as well as no further processes in case of basic product types.

Figure 4.1: The heading machines floor of 2 screws production environments

It is worth to notice in advance that this production environment does not involve FMS machines. The process in fact is made by simple steps one after the other, but those steps do not take place in the same workstation as in FMS systems. Nevertheless, this does not compromise the applicability of the model. From the description of the facility layout as in figure 4.2, it can be seen that the heading machines are disposed one next to the

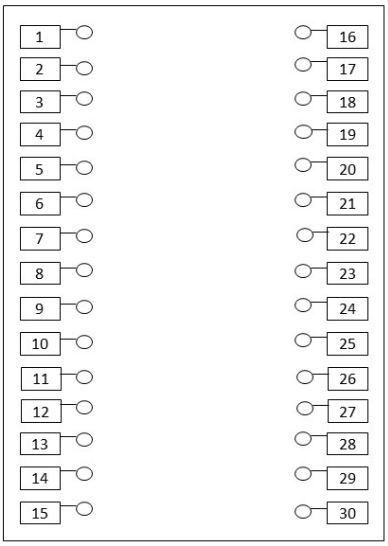


Figure 4.2: Layout of the heading machines floor

other. The heading machine is the first part of the process for any of the product type that will be done. Moreover, all the machines are able to produce all the product types, so – in the first planning phase of the production – the main decision that the management will take is the prioritization of the product types to be produced in a certain time horizon. In doing this, the whole system can be considered as a single machine, since all the production lines in the facility have the capability of handling all the part orders. Thus, with the appropriate hypothesis (see assumption number 5 in chapter 3), the model can be fit to this case.

Our scope is to provide the company a simulated batch that involves all the part type orders planned for March, 2016. In particular, in that month the production facility was asked to make 182 different product types. We have analysed the orders and provided batches by using the robust version of the Part

Type Selection Model (chapter 3).

4.2 Assumptions

In the development of this real case test, the following assumptions have been made:

1. The whole system is considered as a single machine (assumption number 5 in chapter 3).

2. Aggregate production will be considered: orders that include the same product code will be merged as if they were the same order. In fact, once a machine is set to produce one part type, the highest number of parts possible will be produced, since setups prove to be quite costly to the facility (each demands around 45 minutes). Therefore, all the orders including the same product specifications tend to be produced one after the other on a specific machine.

3. Only one tool is considered to be mounted on each machine. This represents the fact that each part order requires a certain setup of the machine, but it never happens that a part order requires tool changes during its production. Therefore, in this case, no tool magazine capacity problems will affect the production planning process.

4. Demand quantities are considered as the actual order that the company management delivers to the production floor: to the market demand is added between 5 and 10% of increase. This is due to the inclusion of the expected production scrap in the calculation of the quantities to be produced, which is normally done on the basis of historical data.

5. Priorities are assigned to the workpieces based on their due date, as proposed by Hwang [37]: products with the earlier due date will be given a higher weight; when products have the same due date, they will be given the same weight. For example, the products to be processed on the first heading machine with their relative weights are shown in table 4.1.

Table 4.1: Parts to be produced on the heading machine M1 and their corresponding weights

Product Code	**ID**	**Demand**	**Weight** w_i	**Due date**
92658	117	667675	1	01-apr
92658	117	5551	2	24-mar
93270	157	131770	2	18-mar
92699	119	174984	3	16-mar
92699	119	79454	4	15-mar
92985	140	42341	4	10-mar
92985	140	122539	4	14-mar
92985	140	64523	5	10-mar
93301	160	66679	5	09-mar
93331	161	205829	6	07-mar
93331	161	112500	7	04-mar
93290	159	63946	7	03-mar

6. Some products, even if they have the same product code, due to their different dimensional specifications are set to be produced on different machines operating at different speeds: this is also due to the minimization of the setups. In this case, they are considered as if they were two different products.

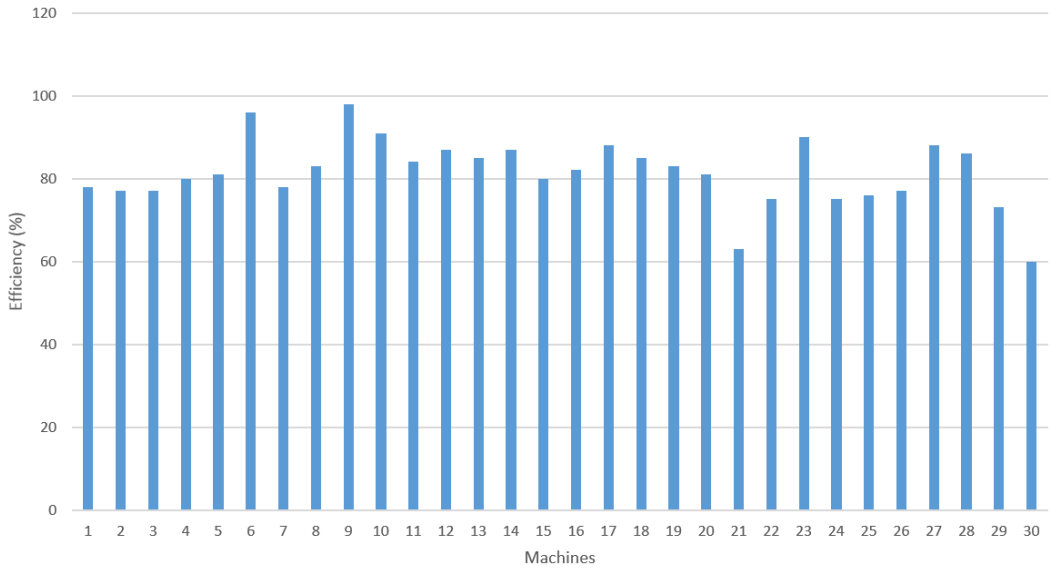
4.3 Model of data uncertainty

The model for data uncertainty as presented in section 2.2 will be used in this case. The robust model covers from a certain

number of disruptive events that are expected to force the processing of the batch to be more time-demanding. In this case, we are considering data uncertainty on the basis of observations from the maintenance records and from the statistics regarding disruptive events and extra-ordinary maintenance interventions.

Let us observe, in figure 4.3, the average efficiency of the production floor. The graph shows the average efficiency of the

Figure 4.3: Average efficiency of the heading machines in March, 2016



heading machines, calculated over the whole month of March 2016. The efficiency is hereby defined as the effective working time over the total shift time. The downtime details are shown in graph 4.4.

It can be seen that the average working time is $81,45\%$ of the total time available. In fact, the management uses the value of 80% as reference to consider the effective time available for production. Although on average it is possible to rely on these aggregate data, it is helpful to look at more detailed statistics of downtimes. Let us then examine the downtime statistics of one

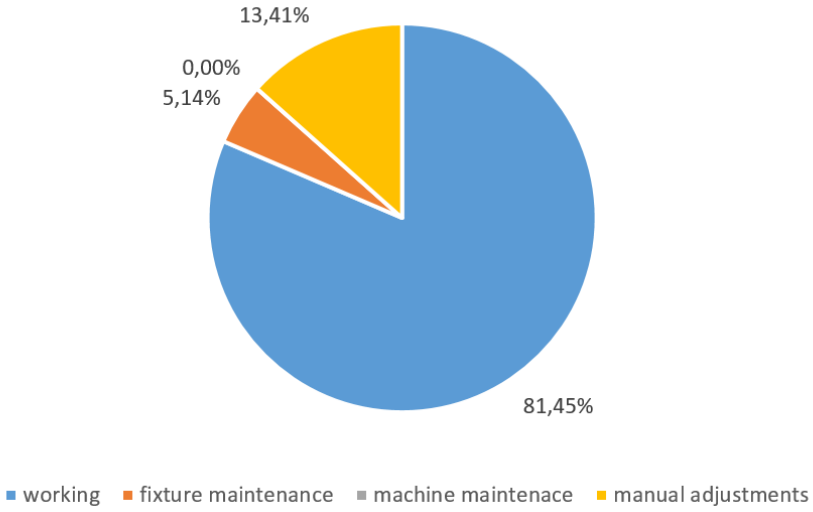


Figure 4.4: Machines downtime for March, 2016

day of production. The day has been randomly selected from the available database, and it is May 4th, 2016.

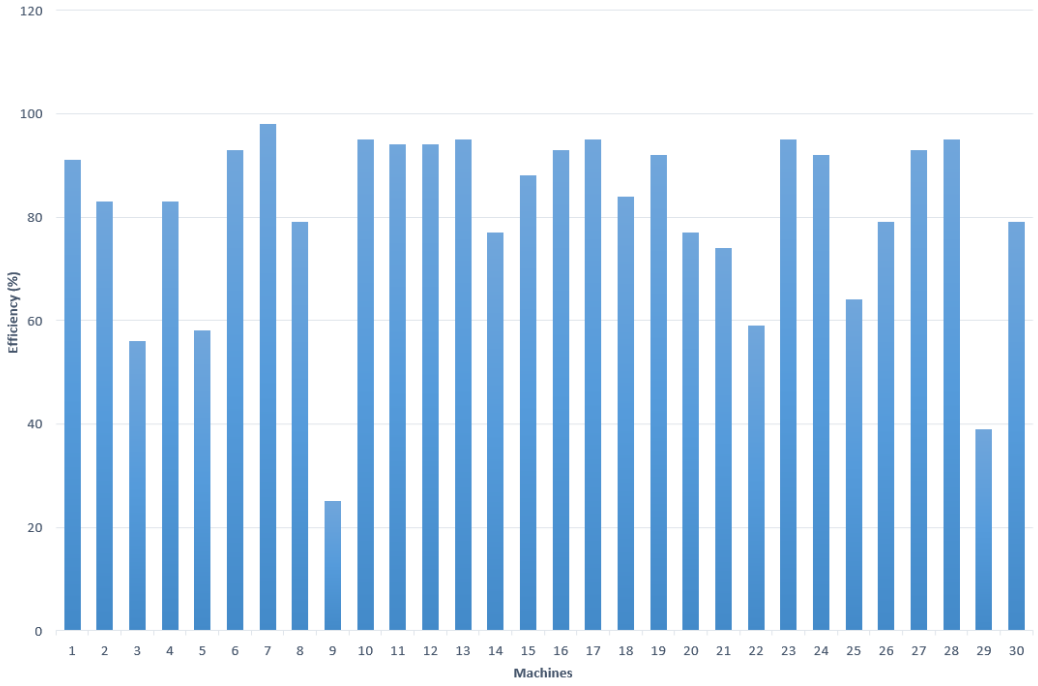


Figure 4.5: Machines downtimes on May 4th

Figure 4.5 shows the same data of figure 4.3, but referred to one day of production: the variability of these data is visibly higher than the one observed over the whole month. In fact, for this day, the average working time happened to be $84,18\%$.

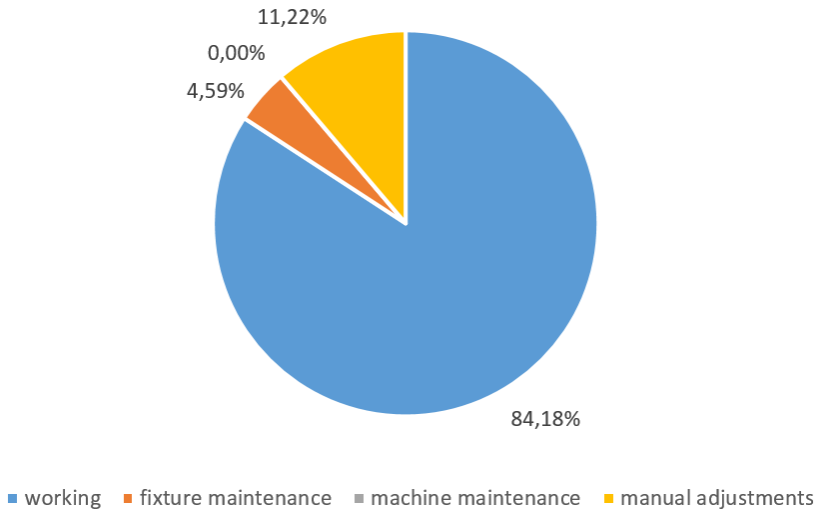


Figure 4.6: Machines downtimes on May 4th, 2016

Maintenance is usually done to adjust the fixtures for the production of a particular batch, to substitute the worn tools at the end of their life, as well as to fill the lubricant levels of the machines. Moreover, the management regularly includes the setup time in the maintenance records. Each setup requires around 45 minutes and is carried out in order to configure a machine to produce a particular part type: both the machine tools and the cycle time are set during this operation. In addition to this, extra-ordinary maintenance is seldom carried out, although there are no obtainable data concerning it.

This statistics motivate the application of the robust formulation of the Part Type Selection Model. A robust solution for this case would provide a list of product types to be produced that not only considers the priority of the orders, but is ready to face a certain number of disruptive events that might happen in

the considered time horizon. The company asserts that this kind of events are normally expected to happen once a year, and they usually involve the loss of one or two orders. The Robust Part Type Selection Model is supposed to be used the same way as the original model of Hwang and Shogan. That is, batches will be decided with a rolling horizon: each time a batch is gathered, the left out items will be considered for the next one, together with the new orders eventually received. Notice that all the data, including the weights, can be changed at each calculation, and this preserves the flexibility of the model.

4.4 Data

The robust Part Type Selection model has been applied for the part orders that have been produced in the month of March 2016. The data used in this case is listed hereafter:

- The weights w_i that have been inserted are: (1) linear, proportional to the due date, and (2) constant.
- The demand and machine speed quantities have been always converted in thousands of workpieces. This simplifies the formulation and makes the data easier to read.
- The demand over the month of March 2016 involves 182 different part types. Each product code has been given a progressive number to be traced.
- The machine tool capacity has been considered so to be not critical, as described in section 4.2. Moreover, one single tool has been included, occupying $k_j = 1$ slots, and the machine tool capacity has been considered to provide $S = 2$ slots. Each workpiece would then require this tool, so $a_{ij} = 1 \, \forall i, j$. Thus, this constraint of the model will never be critical.

- Two cases for the deviation time $\hat{t}_{ij}$ have been analyzed: (1) the same amount of time as the operation time (that corresponds to the double time needed for production) DEV1, and (2) a 10 times larger than the operation time, DEV2, time.

- The total available processing time M has been calculated referring to the working calendar of the facility. The facility works on a 5-day week basis, 1 shift of 9 hours per day. In March 2016 there have been 24 working days. In order to consider overtimes and the fact that many production orders have been completed in the first week of April, an extra week has been added to the working time, for a total of 30 working days. In the calculation of the total available time, the 80% efficiency has been included as well. Thus, we obtain:

$$M = 30\,days \cdot 9\,hours/day \cdot 60\,min/hour \cdot 0.8 = 388800\,min \tag{4.1}$$

- Different cardinality cases have been considered. First of all let us notice that since the model is a one-tool scenario one, only one cardinality value Γ_j is required, as stated in assumption 3. As a consequence, all the disruptive events will affect that specific tool. The tested scenarios are, in order, the nominal case (1) $\Gamma_j = 0$, and the following: (2) $\Gamma_j = 1$, (3) $\Gamma_j = 2$, (4) $\Gamma_j = 3$, (5) $\Gamma_j = 4$, (6) $\Gamma_j = 5$. It is reasonable to think that, even though there are no specific restrictions in the model, a higher number of unfortunate events would not be representative of any particular situation. Moreover, it is not meaningful to plan a production with such a high number of unfortunate events: in that case, clearly, other types of corrections are of priority.

4.5 Results

Let us now analyse the results obtained applying the model to the real case. The objective function obtained in each case will be displayed, as well as the number of part types that, for each run, have been left out from the batch (corresponding to the number of times that y_i equals 0).

4.5.1 Case 1: time deviations DEV1, linear weights

As it can be noticed from figure 4.7, the objective function is a non-increasing function of Γ_j: this means that with an increasing number of disruptive events, the total production time will increase, thus forcing the exclusion of some part orders from the batch. Similarly, the total number of part types included in the batch is a non-increasing function of Γ_j, such as depicted in figure 4.8. This behavior causes the objective function to become lower over Γ, as in figure 4.7.

Γ_j	Objective function	Non-included part types
0	941	3
1	940	3
2	939	4
3	938	5
4	937	6
5	937	6

Table 4.2: Case 1: Objective function and number of non-included part types

4.5.2 Case 2: time deviations DEV1, constant weights

As it can be seen in figure 4.10, the number of part types not included in the batch is the same as in the case with deadline-dependent weights.

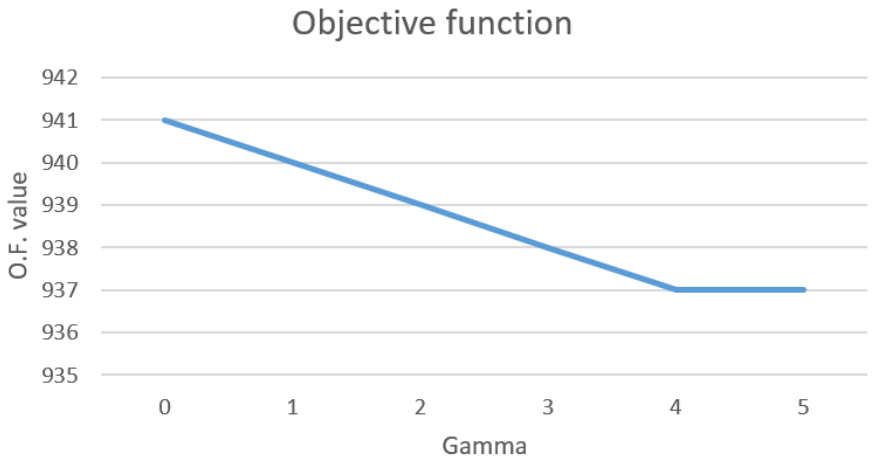


Figure 4.7: Objective function with DEV1, deadline weights w_i

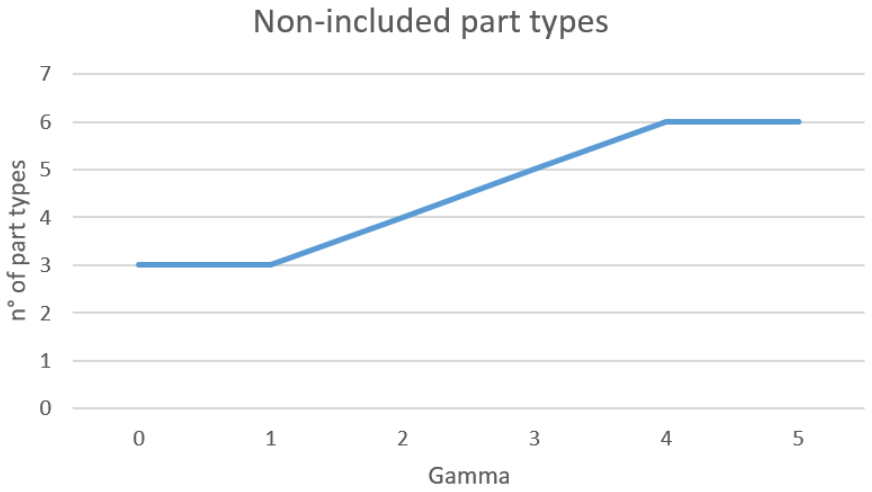


Figure 4.8: Number of part types not included in the batch, with DEV1 and deadline weights w_i

Although the results might seem the same, a closer look to the batch details shows that in the case with variable weights, it is more frequent to see cases in which after the exclusion of a part type in the batch, with an additional disruptive event the same part type is included again in the batch. This phenomenon is illustrated in table 4.3, were the batch result vector $\mathbf{y}^*$ for one of

the product types is presented (1 means the product is included in the batch, 0 otherwise).

Table 4.3: Comparison of the results obtained with dependent and independed weight referred to the the batch for the product type coded 139

Cardinality Γ	0	1	2	3	4	5
$\mathbf{y}^*$, with dependent weights	1	1	1	1	0	0
$\mathbf{y}^*$, without dependent weights	1	1	1	1	0	1

This behaviour is due to the fact that each time a disruptive event is considered, the remaining production time available changes, and the maximization of the part types included in the batch makes it feasible for some parts to be inserted again.

Γ_j	Objective function	Non-included part types
0	179	3
1	179	3
2	178	4
3	177	5
4	176	6
5	176	6

Table 4.4: Case 2: Objective function and number of non-included parts

Since the objective of the model is the prioritization of the parts included in the batches, the latter phenomenon is not very meaningful and could bring confusion in the prioritization process. In fact, the model is meant to be used with a rolling horizon. With an increase in the protection level (cardinality), a decision maker would expect that a part that has been excluded would keep being so for all the time periods the model will be used over. However, this is valid if the set of parts we consider does not change. We refer to section 4.5.5, where it is proposed a way to prevent this problem by changing the objective function of the model.

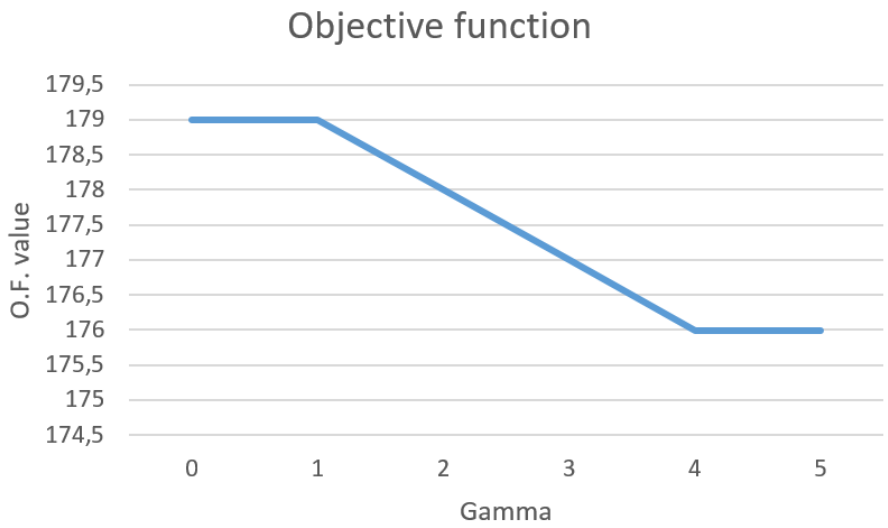


Figure 4.9: Objective function with DEV1, weights $w_i = 1$

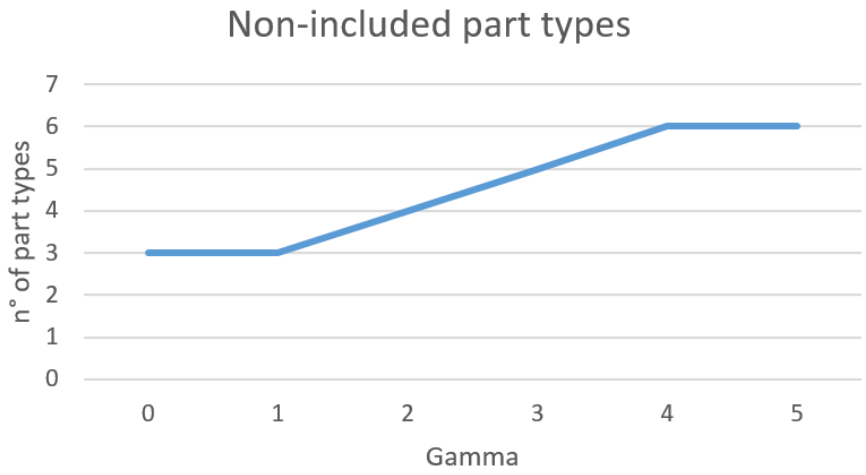


Figure 4.10: Number of part types not included in the batch, with DEV1 and weights $w_i = 1$

4.5.3 Case 3: time deviations DEV2, linear weights

Results of the case with time deviations DEV2 and linear weights are shown in table 4.5, figure 4.11 and figure 4.12. The same pattern obtained in previous cases can be appreciated in this case: the only difference lays in the number of non-included

part types, that is visibly higher than the other cases. This is clearly due to the fact that with longer disruptions, less productive time remains available, so less part types will be included.

Γ_j	Objective function	Non-included part types
0	941	3
1	928	10
2	916	15
3	907	23
4	895	22
5	882	25

Table 4.5: Case 3: Objective function and number of non-included parts

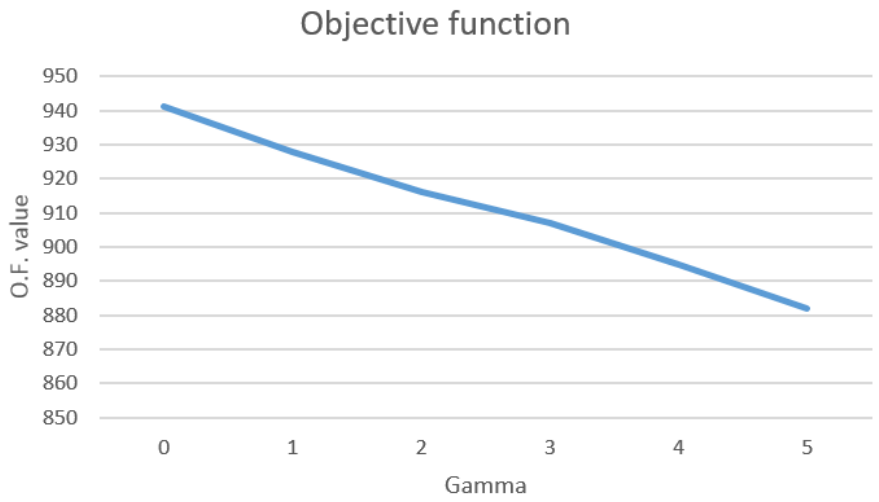


Figure 4.11: Case 3: Objective function value over the cardinality Γ_j

4.5.4 Case 4: time deviations DEV2, constant weights

The forth case has been done with deviations DEV2 and constant weights: results are shown in table 4.6 and figures 4.11 and 4.14. Again we can acknowledge the pattern that has been

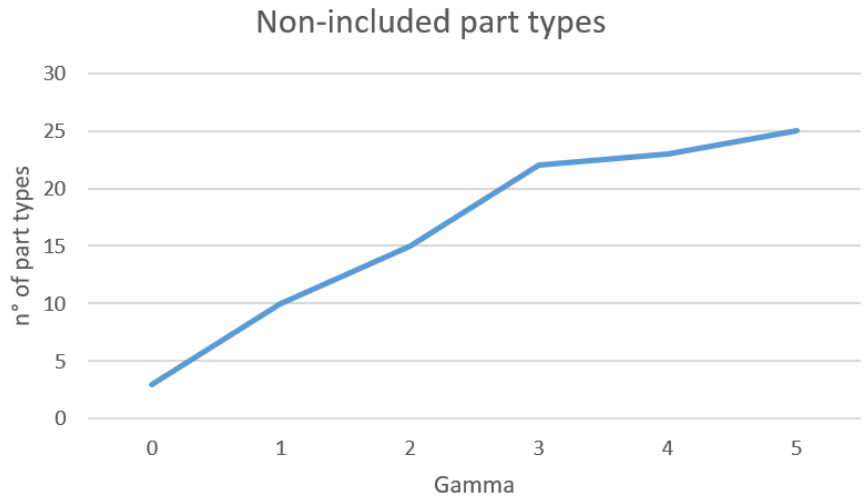


Figure 4.12: Case3: Number of part types not included in the batch over the cardinality Γ_j

observed in previous cases, with the exception of the number of non-included part types.

Γ_j	Objective function	Non-included part types
0	179	3
1	172	10
2	167	15
3	163	19
4	161	21
5	158	24

Table 4.6: Case 4: Objective function and non-included part types

4.5.5 Case 5: dependent weights case

Let us consider equation (3.25): the objective function of the robust formulation. In order to prevent the problem described in section 4.5.2, that is the insertion of part types previously excluded, it is possible to modify the objective function in the following way:

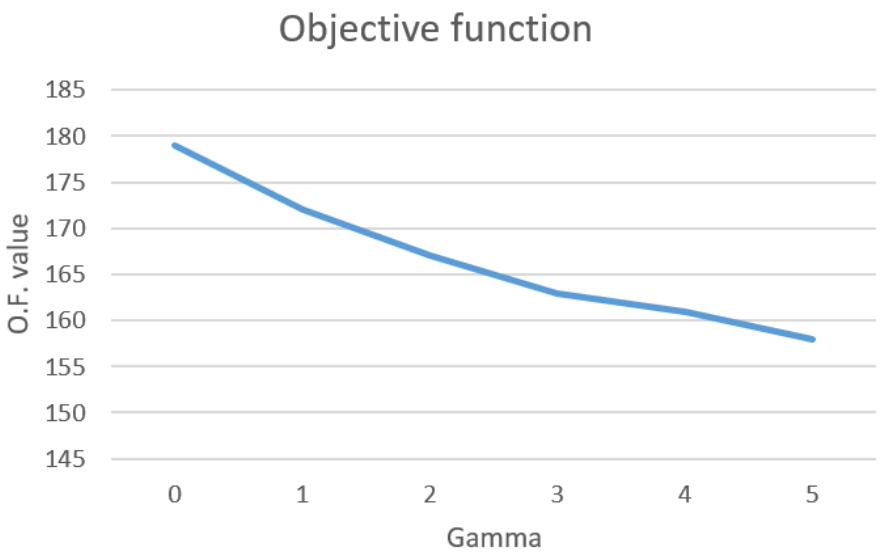


Figure 4.13: Case 4: Objective function value over the cardinality Γ_j

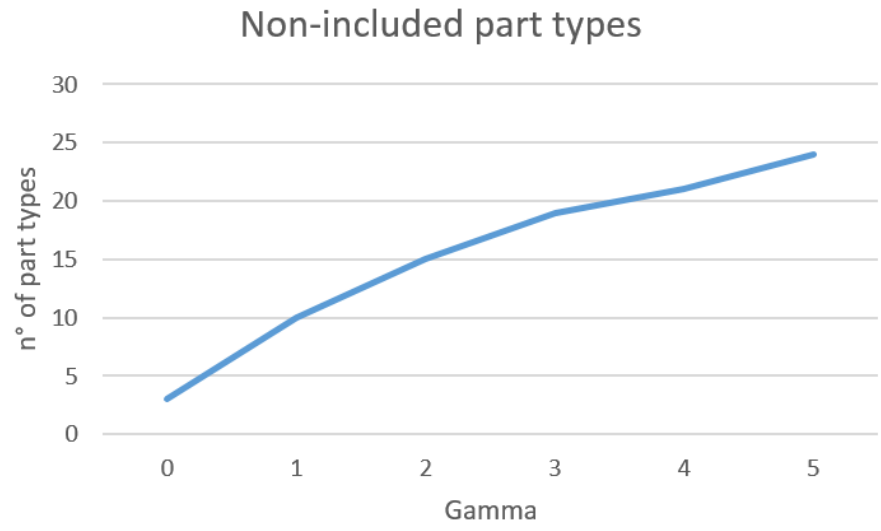


Figure 4.14: Case 4: Number of part types not included in the batch over the cardinality Γ_j

$$\max \sum_i \left[w_i - v_i (1 - y_i^{(\Gamma_j - 1)}) \right] y_i \tag{4.2}$$

where v_i are weights that – in this case – have been set $v_i = w_i$, and represent a penalty to the inclusion of a part types that has been previously excluded in cases with lower cardinality values: that are represented by $y_i^{(\Gamma_j - 1)}$. Notice that since in this new formulation the solution depends on the batches obtained with the previous cardinality value, an initial vector $y_i^{(-1)}$ is required for the calculation in the nominal case $\Gamma_j = 0$; it is easy to see that a initial vector $y_i^{(-1)} = 1 \, \forall i$ will be fit for this scope, since it would reduce the objective function to the previous one, $\sum_i w_i \, y_i$.

Let us then compute and compare the results with the application of independent and dependent weights, respectively. Table 4.7 shows the objective functions obtained in both cases and the reduction in the objective function Δ_{OF} calculated as:

$$\Delta_{OF} = O.F.^{(dependent)} - O.F.^{(independent)} \tag{4.3}$$

The same results are shown in figure 4.15. It can be seen that the reduction in the objective function is not too dramatic, so the model is still applicable with good results.

Γ_j	O.F. (independent)	O.F. (dependent)	Δ_{OF}
0	941	941	0
1	940	939	-1
2	939	938	-1
3	938	937	-1
4	937	937	0
5	937	936	-1

Table 4.7: Case 5: Objective function in case of dependent Vs. independent weights

Similarly, the number of items left out from the batch in both cases has been computed, and shown in table 4.8 and figure 4.16. The quantity Δ_{parts} is the difference in the number of parts that

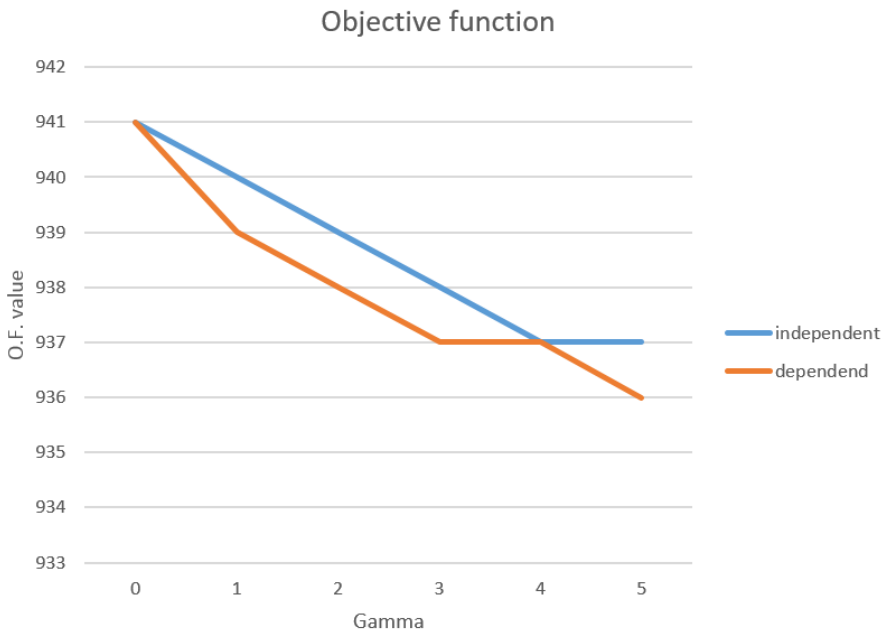


Figure 4.15: Case 5: Comparison between objective functions in the cases of independent Vs. dependent weights

have been excluded. We can assert that the difference is not too big to constitute a burden in the application of the modified model.

Γ_j	Excluded parts (independent)	Excluded parts (dependent)	Δ_{parts}
0	3	3	0
1	3	4	1
2	4	5	1
3	5	6	1
4	6	6	0
5	6	7	1

Table 4.8: Case 5: number of excluded parts in case of dependent Vs. independent weights

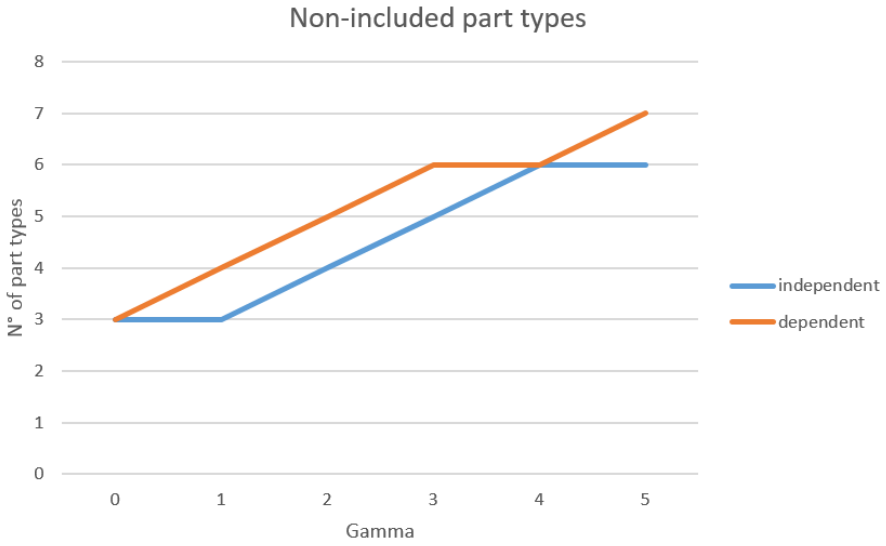


Figure 4.16: Case 5: Comparison between number of excluded items in the cases of independent Vs. dependent weights

4.6 Conclusions

In this chapter, the application of the robust counterpart of the Part Type Selection Model introduced in chapter 3 has been performed and the numerical results have been analyzed. The case of dependent weights has been proposed in order to obtain batching results more applicable in a real situation and the comparison between the two cases has been done. In section 4.5.5, with a slight modification to the objective function of the model, we have shown how the robust model can fit better to real situations; therefore, we believe it can be used as reference for future applications.

Table 4.9: Results of the application of the robust version of the Part Type Selection model to a real case: comparison between the cases with dependent and independent weights (Linear weights case)

	DEV1				DEV2			
	Independend weights		Dependend weights		Independend weights		Dependend weights	
Cardinality Γ	O.F.	non incl. items	O.F.	non incl. items	O.F.	non incl. items	O.F.	non incl. items
0	941	3	941	3	941	3	941	3
1	940	3	939	4	928	10	928	11
2	939	4	938	5	916	15	916	16
3	938	5	937	6	907	23	907	19
4	937	6	937	6	895	22	895	22
5	937	6	936	7	882	25	882	25

Table 4.10: Results of the application of the robust version of the Part Type Selection model to a real case: comparison between the cases with dependent and independent weights (Constant weights case)

	DEV1				DEV2			
	Independend weights		Dependend weights		Independend weights		Dependend weights	
Cardinality Γ	O.F.	non incl. items	O.F.	non incl. items	O.F.	non incl. items	O.F.	non incl. items
0	179	3	179	3	179	3	179	3
1	179	3	179	3	172	10	172	10
2	178	4	178	4	167	15	167	15
3	177	5	177	5	163	19	163	19
4	176	6	176	6	161	21	161	21
5	176	6	176	6	158	24	158	24

Chapter 5

Machine Loading Problem

This chapter deals with the Machine Loading Problem. The aim is to build a robust counterpart of a well-known problem that is applied in the Production Planning phase. In the first part of the chapter we will describe the problem and a widely used model for tooling the machines of a FMS. Consequently, we will show how to develop the robust formulation of the problem, and the results of its application to test cases. In particular, section 5.2 presents the model of Sodhi, Askin, and Sen; in section 5.3 the robust version of the model is developed, and the numerical results of its test on two small cases is presented in section 5.4. In section 5.5 a potential extension to the model is proposed.

5.1 Problem Description

The Machine Loading problem has been defined by Stecke (1983) [38] as follows:

> *"Allocate the operations and the required tools of the selected part types among the machine groups subject to technological and capacity constraints of the FMS"*

A detailed survey about loading models is available in [39], where the authors group the elements that influence the definition of a loading problem in three main categories: (1) the

characteristics of the FMS, (2) the characteristics of the plant where the FMS operates, and (3) the interface of the loading module with the other levels of management.

Sodhi et al. [40] developed a Machine Loading problem that aims to find the allocation of tools and production schedule throughout a short time horizon, in order to satisfy the production plan (see figure 5.1 for a graphical representation of the problem). The latter is assumed to be done in a previous calculation, i.e. by a medium term Part Type Selection Model – see chapter 3 – or by an MRP or any other aggregate Production Planning model.

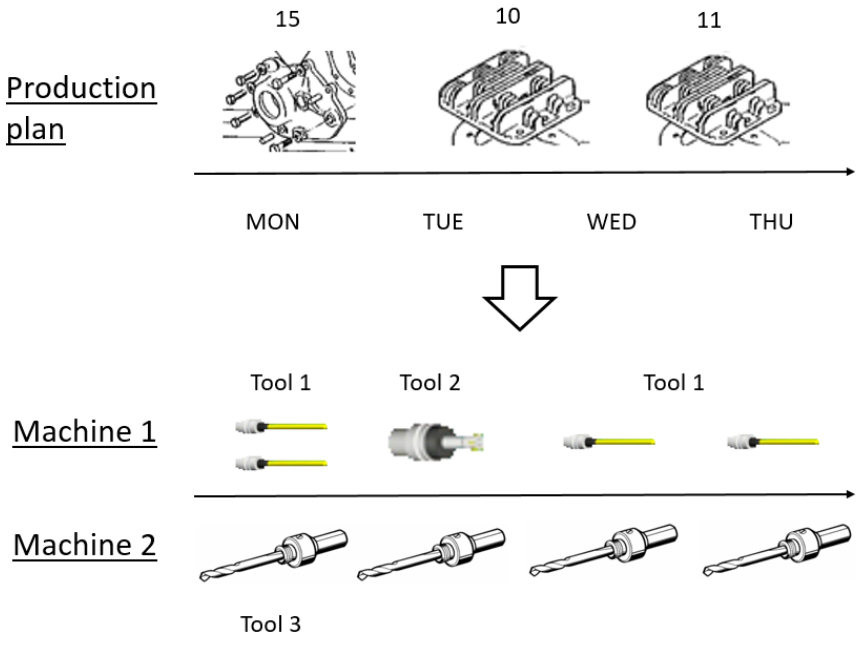


Figure 5.1: The Machine Loading Problem

5.2 The Machine Loading Model of Sodhi, Askin and Sen

We have chosen to base our analysis on the Machine Loading model proposed by Sodhi, Askin and Sen in 1994. The choice of this model is mainly due to the fact that it is simple and it includes constraints easy to understand, but at the same time it is complete since it includes all the main parameters of the problem and the core issues identified in [39].

5.2.1 Assumptions

Assume that a set of parts has already been selected for production over the short term (notice this is the solution to the Part Type Selection Problem). Moreover, it is assumed that no tool transportation system is present, so static tool changes are considered: it is not possible to change the tools without having to stop the system from its operations. This represents a situation in which the system operates unattended for a certain length of time and it is then retooled in order to run again for the next time interval. Thus, a time period corresponds to the length of time between planned tool-changeovers, and the number of periods is defined a priori. Production quantities are assumed to be continuous. This is an approximation of the problem: however, it allows to plan a partial production for certain part types. Tool storage capacity is assumed to be limited. The time horizon of the model is the production planning period in which requirements have been planned.

5.2.2 Model

Let us then display the Machine Loading model:

$$\min \quad \sum_i S_i\, C_i + \sum_i \sum_t x_{it}\, R_{it} \tag{5.1}$$

$$s.t. \quad \sum_j T_{jmt}\, k_j \leq K_m \quad \forall m, t \tag{5.2}$$

$$\sum_t x_{it} = D_i - S_i \quad \forall i \tag{5.3}$$

$$\sum_i O_{ij}\, x_{it} \leq \sum_m p_{jmt} \quad \forall j, t \tag{5.4}$$

$$p_{jmt} \leq T_{jmt}\, A_{mt} \quad \forall j, m, t \tag{5.5}$$

$$\sum_j p_{jmt} \leq A_{mt} \quad \forall m, t \tag{5.6}$$

$$p_{jmt} \geq 0 \quad \forall j, m, t \tag{5.7}$$

$$x_{it} \geq 0 \quad \forall i, t \tag{5.8}$$

$$S_i \geq 0 \quad \forall i \tag{5.9}$$

$$T_{jmt} \in \{0, 1\} \tag{5.10}$$

Decision variables. S_i is the shortage of production of part type i with respect to its demand. x_{it} is the quantity of part order i produced in time period t. T_{jmt} is a boolean variable that assumes value 1 when tool j is loaded on machine m in the time period t. The variable p_{jmt} represents the time spent for production on the j-th tool of machine m in the time period t.
Parameters. C_i is the penalty cost per unit of production for not satisfying the demand: it could represent the sub-furniture cost if outsourcing is allowed. R_{it} is the cost of holding one product unit produced in period t until the end of the time horizon. k_j is the number of slots required by tool j. K_m is the total slots available in the slot magazine of machine m. D_i is the demand of product i in the time horizon that has been considered. A_{mt} is the available time for production on machine m in time period t. O_{ij} is the processing time of one unit of

product i on the j-th tool. It is also possible to consider the maximum number of tool copies available with the addition of the constraint:

$$\sum_m T_{jmt} \leq \alpha_j \quad \forall j, t \tag{5.11}$$

where α_j is the number of copies of tool j available. This is a common situation in a production environment where some operations require expensive tools, and only a limited number of copies will be kept on site.

Objective function. The objective function (5.1) aims at minimizing the total cost composed by stock and shortage costs.

Constraints. Constraint (5.2) limits the number of tools that can be loaded on machines since the total number of slots available. Constraint (5.3) aims at satisfying the demand of product i. Constraint (5.5) guarantees that production can be done only if the necessary tool has been loaded. Constraint (5.4) limits the total production time on the j-th tool in time period t. Constraint (5.6) guarantees that production time does not exceed the availability of each machine m.

5.3 Robust formulation

Let us now build the robust counterpart of the Machine Loading Problem. Firstly, it is desirable to keep a simple formulation and to deal with a maximization problem. Therefore, let us modify the objective function of the problem into:

$$\max \quad \sum_i \sum_t w_i \, x_{it} - \sum_i S_i \, C_i \tag{5.12}$$

where we have substituted the product carrying costs R_{it} with generic weights w_i, representing the profit generated by the pro-

duction of a unit of product x_i, and changed the sign of the sum of shortage costs S_i.

5.3.1 Model of data uncertainty

Similarly to the Part Type Selection Model, the respect of machine tool magazine constraints and machine time availability constraints are the main issues in the Machine Loading Problem. Tool storage availability is usually decided by the machine manufacturer and it is not easy to change afterwards. Similarly, the slots required by each tool is usually a constant, standard value. Also the time available for production is relatively hard to modify, since it is related to the shift policy adopted by the company. Let us consider the processing time O_{ij}. Similarly to the analysis done in chapter 3, we can model these times as random variables $\tilde{O}_{ij}$, symmetrically distributed in the interval $\left[\bar{O}_{ij} - \hat{O}_{ij}, \bar{O}_{ij} + \hat{O}_{ij}\right]$, where $\bar{O}_{ij}$ is the expected value of the processing time, and $\hat{O}_{ij}$ is the maximum variation from it. In our formulation, this deviation represents a disruptive event that can cause the processing time to rise, from which we desire to cover.

5.3.2 Protection function

Let us then write constraint (5.4) so to highlight the problem:

$$\sum_i O_{ij}\, x_{it} + \max_{\Omega} \left\{ \sum_{i \in S_j} \hat{O}_{ij}\, x_{it} + (\Gamma_{jt} - \lfloor \Gamma_{jt} \rfloor) \hat{O}_{u_j,j}\, x_j \right\} \leq \sum_m p_{jmt} \quad \forall j, t \tag{5.13}$$

where

$$\Omega = \{S_j \cup \{u_j\};\ S_j \subseteq J_j;\ |S_j| = \lfloor \Gamma_{jt} \rfloor;\ u_j \in J_j \setminus S_j\} \tag{5.14}$$

The inner maximization problem in the equation represents the aim to be covered from the worst combination of disruptive events: among the Γ_{jt} processing times expected to rise, the ones having the worst impact on the problem will be selected. This maximization problem can be written as:

$$\beta_{jt} = \max \quad \sum_{i} \hat{O}_{ij}\, x^{*}_{it}\, z_{ijt} \tag{5.15}$$

$$s.t. \quad \sum_{j} z_{ijt} \leq \Gamma_{jt} \quad \forall j, t \tag{5.16}$$

$$0 \leq z_{ij} \leq 1 \quad \forall i, j \tag{5.17}$$

where x^{*}_{it} is a particular solution of the problem, z_{ijt} are continuous variables between 0 and 1, and Γ_{jt} is the cardinality of the subset of values of O_{ij} expected to assume their maximum value. The dual problem thus becomes:

$$\beta_{jt} = \min \quad \Gamma_{jt}\, z_{jt} + \sum_{i} q_{ijt} \tag{5.18}$$

$$s.t. \quad z_{jt} + q_{ijt} \geq \hat{O}_{ij}\, x^{*}_{it} \quad \forall i, j, t \tag{5.19}$$

$$z_{ij} \in \mathbb{R}^{+} \tag{5.20}$$

$$t_{ijt} \in \mathbb{R}^{+} \tag{5.21}$$

At this point, the dual can be inserted in the master optimization problem, that becomes:

$$\max \quad \sum_i \sum_t w_i \, x_{it} - \sum_i S_i \, C_i \tag{5.22}$$

$$s.t. \quad \sum_j T_{jmt} \, k_j \leq K_m \quad \forall m, t \tag{5.23}$$

$$\sum_t x_{it} = D_i - S_i \quad \forall i \tag{5.24}$$

$$p_{jmt} \leq T_{jmt} \, A_{mt} \quad \forall j, m, t \tag{5.25}$$

$$\sum_i O_{ij} \, x_{it} + \Gamma_{jt} \, z_{jt} + \sum_i q_{ijt} \leq \sum_m p_{jmt} \, \forall j, t \tag{5.26}$$

$$z_{jt} + q_{ijt} \geq \hat{O}_{ij} \, x_{it} \quad \forall i, j, t \tag{5.27}$$

$$\sum_j p_{jmt} \leq A_{mt} \quad \forall m, t \tag{5.28}$$

$$T_{jmt} \in \{0, 1\} \tag{5.29}$$

$$p_{jmt} \in \mathbb{R}^+ \tag{5.30}$$

$$x_{it} \in \mathbb{R}^+ \tag{5.31}$$

$$S_i \in \mathbb{R}^+ \tag{5.32}$$

$$z_{ij} \in \mathbb{R}^+ \tag{5.33}$$

$$q_{ijt} \in \mathbb{R}^+ \tag{5.34}$$

5.4 Numerical results

In this section some test cases will be presented. The robust formulation has been tested on a couple of test cases: the problems are smaller than a real one, but not trivial. This preserves the simplicity of the formulation and makes the behavior of the model easy to understand.

5.4.1 Case 1: single period, multiple machines

The first instance regards the case in which one time period is taken into consideration: the situation takes an eight-hour production shift as reference. Three machines are assumed to be processing this batch, so each machine will have the same availability of $A_{mt} = 8\,hour/shift \cdot 60\,min/hour = 480\,min$. Moreover, each machine is assumed to have a tool magazine capacity of $K_m = 10$ tools. Three tools are assumed to be required for each workpiece, and the processing time for each workpiece is $O_{ij} = 5\,min$. Each tool requires $k_j = 3$ slots. This way, machine slots capacity does not influence the solution of the problem. A constant demand of $D_i = 10$ pieces is assumed for all the workpieces. Production profits w_i are assumed to be constant, equal to $300\$/workpiece$. Shortage costs of $S_i = 600\$/workpiece$ have been considered. The production time deviations are assumed to be equal to the processing time, thus representing the situation in which a disruptive event doubles the processing time. The cardinality values that have been selected are: (1) $\Gamma_j = 0\,\forall j$, (2) $\Gamma_j = 1\,\forall j$, (3) $\Gamma_j = 2\,\forall j$, (4) $\Gamma_j = 3\,\forall j$, (5) $\Gamma_j = 4\,\forall j$, (6) $\Gamma_j = 5\,\forall j$.

Table 5.1 shows the results obtained in this case: the objective function and the total production, obtained as $X_{TOT} = \sum_i x_i$. From figures 5.2 and 5.3 we can appreciate the behavior of the model in this case. The objective function and the total production are decreasing when the number of unfortunate events increases. Table 5.2 shows the detailed results of case 1: for each cardinality value, the product types production levels are shown.

5.4.2 Case 2: multiperiod, single machine

The second test is the case where one machine and multiple time periods have been considered. The data has been taken from

Table 5.1: Case 1: Objective function and total production

Γ_j	O.F.	X_{TOT}
0	27200	96,0
1	21090	87,2
2	16000	80,0
3	11692	73,8
4	8000	68,6
5	4800	64,0

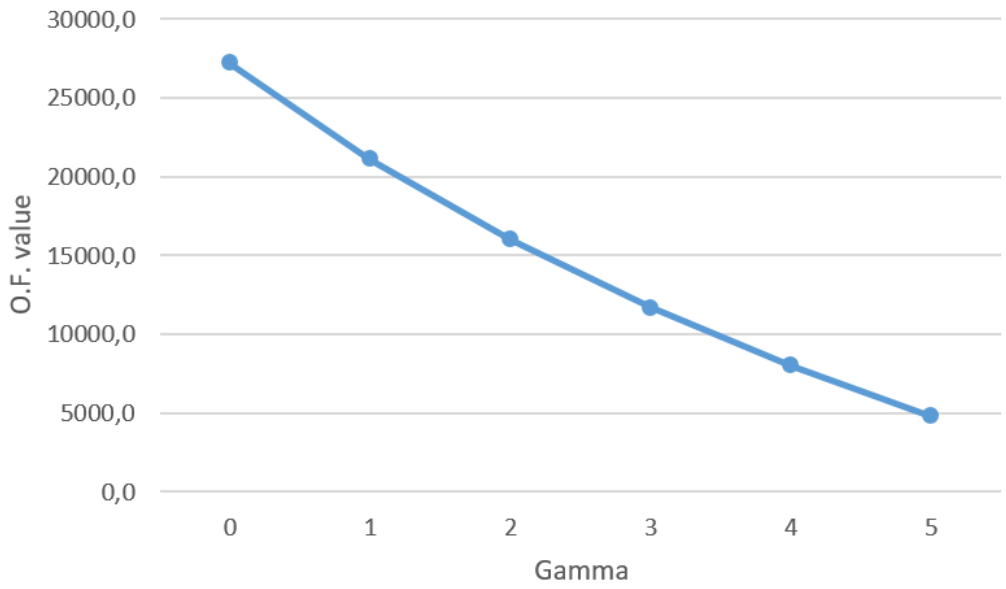


Figure 5.2: Case 1: Objective function

the real case that was analyzed by the authors of [41], where they study a Machine Loading Problem with 12 workpieces and 12 machine tools. A shift of 9 hours has been considered, so the machine availability is $A_{mt} = 9\,hour/shift \cdot 60\,min/hour = 540\,min$. A total of 5 time periods is considered, thus corresponding to a week of production. Moreover, the machine is assumed to have a tool slots capacity of $K_m = 30$ slots. The slots required from each tool k_j are the values of vector $\mathbf{k}$:

$$\mathbf{k} = \begin{bmatrix} 4 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 3 & 3 \end{bmatrix} \tag{5.35}$$

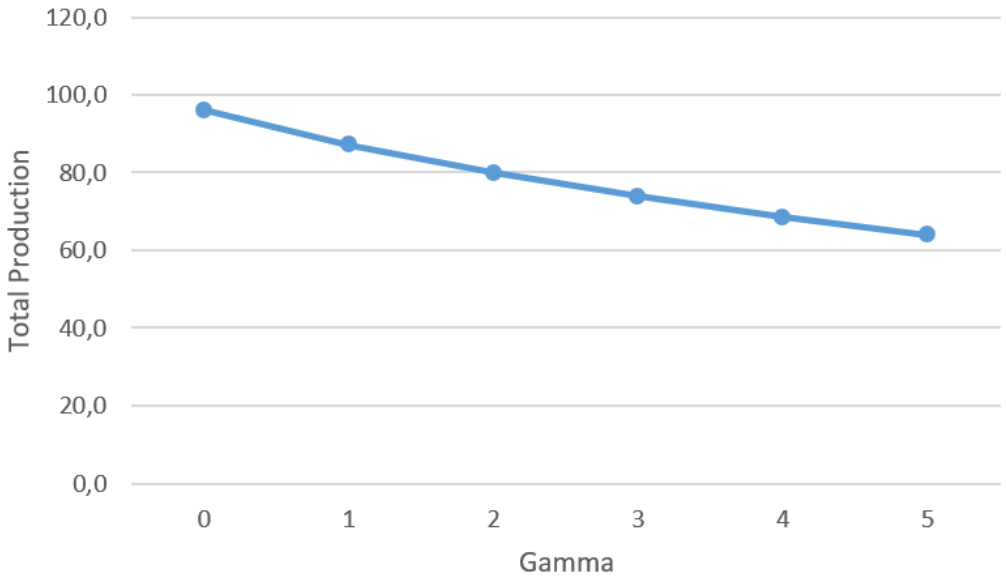


Figure 5.3: Case 1: Production level

Table 5.2: Case 1: Production levels for all the product types over different cardinality levels

Product Type	$\Gamma_j = 0$	$\Gamma_j = 1$	$\Gamma_j = 2$	$\Gamma_j = 3$	$\Gamma_j = 4$	$\Gamma_j = 5$
1	10	8.73	8.00	7.38	6.86	6.40
2	10	8.73	8.00	7.38	6.86	6.40
3	10	8.73	8.00	7.38	6.86	6.40
4	10	8.73	8.00	7.38	6.86	6.40
5	10	8.73	8.00	7.38	6.86	6.40
6	10	8.73	8.00	7.38	6.86	6.40
7	10	8.73	8.00	7.38	6.86	6.40
8	10	8.73	8.00	7.38	6.86	6.40
9	10	8.73	8.00	7.38	6.86	6.40
10	6	8.73	8.00	7.38	6.86	6.40

and the processing times O_{ij} of the tools are entries of the matrix $\mathbf{O}$. The vector of the workpieces demand values D_i is $\mathbf{D}$:

$$\mathbf{D} = \begin{bmatrix} 160 & 4 & 8 & 8 & 40 & 4 & 4 & 20 & 20 & 8 & 8 & 4 \end{bmatrix} \tag{5.36}$$

The weights w_i are assumed to be constant equal to $30\,\$/unit$

$$\mathbf{O} = \begin{bmatrix} 4.61 & 2 & 2.74 & 4.73 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2.46 & 2.51 & 0 & 4.19 & 2.7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4.4 & 0 & 0 & 2.86 & 4.62 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.12 & 1.95 & 2.64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.29 & 3.68 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.15 & 5.39 & 2.18 & 4.14 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.7 & 2.44 & 6.18 \\ 0 & 0 & 1.92 & 2.59 & 3.3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3.04 & 5.28 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4.45 & 2.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.42 & 4 & 0 & 0 & 0 & 0 & 5.48 & 5.47 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.22 & 3.62 & 3.92 & 3.72 & 0 \end{bmatrix}$$

and the shortage costs to $S_i = 40\,\$/unit$. The production times deviations are assumed to be equal to the processing time, thus representing the situation in which a disruptive event doubles the processing time. The cardinality values that have been selected are: (1) $\Gamma_j = 0\,\forall j$, (2) $\Gamma_j = 1\,\forall j$, (3) $\Gamma_j = 2\,\forall j$, (4) $\Gamma_j = 3\,\forall j$, (5) $\Gamma_j = 4\,\forall j$, (6) $\Gamma_j = 5\,\forall j$, (7) $\Gamma_j = 6\,\forall j$.

The results are shown in table 5.3, as well as in figures 5.4 and 5.5. It is straightforward to notice that the behavior of the model is similar to Case 1. An increase in the cardinality of the disruptive events causes the production levels to lower, together with the objective function.

Table 5.3: Case 2: Objective function and total production values

Γ_j	O.F.	X_{TOT}
0	7103.1	236.8
1	4851.4	161.7
2	4474.7	149.2
3	4279.4	142.6
4	4244.5	141.5
5	4226.7	140.9
6	4226.7	140.9

Tables 5.4 to 5.10 show the detailed results of Case 2. For each cardinality value, the detailed production levels for the product types is displayed.

5.4.3 Computational times

Tables 5.11 and 5.12 show the computational times of the case studies. The tests have been performed on a notebook Dell XPS13 with i7 Intel Core @2.5Ghz and 8GB RAM (WIN10). Table 5.11 shows the results in the deterministic case, so when $\Gamma = 0$; table 5.12 displays the CPU time in the robust case with the maximum value for Γ, that is $\Gamma = 5$ for Case 1 and $\Gamma = 6$

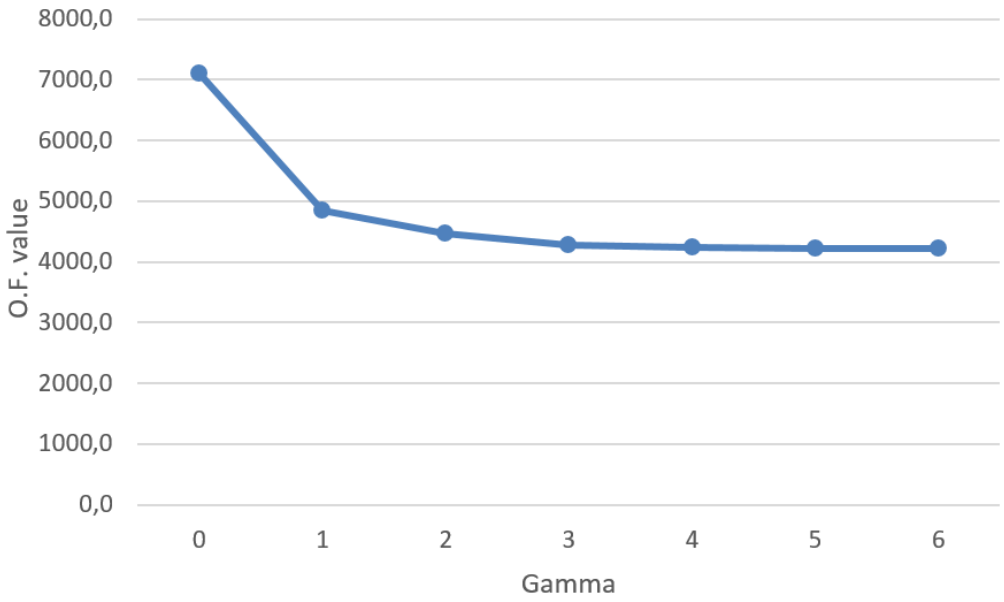


Figure 5.4: Case 2: Objective function

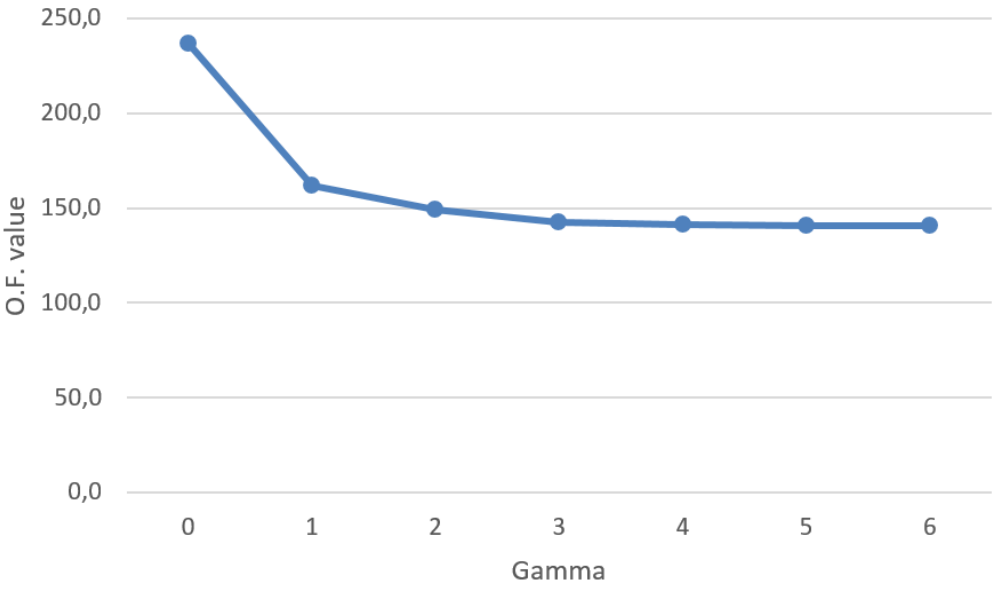


Figure 5.5: Case 2: Production level

for Case 2. The computational times are visibly similar between Case 1 and Case 2. Moreover, the robust version of the problem

Table 5.4: Case 2: Results with $\Gamma_j = 0$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	22.81	12.76	15.71	35.14	38.35
2	0.00	4.00	0.00	0.00	0.00
3	0.00	8.00	0.00	0.00	0.00
4	0.00	8.00	0.00	0.00	0.00
5	0.00	0.00	40.00	0.00	0.00
6	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	4.00	0.00
8	0.00	20.00	0.00	0.00	0.00
9	20.00	0.00	0.00	0.00	0.00
10	8.00	0.00	0.00	0.00	0.00
11	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.00	0.00	0.00

Table 5.5: Case 2: Results with $\Gamma_j = 1$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	3.57	0.00	8.17	10.58	14.88
2	0.00	0.00	4.00	0.00	0.00
3	3.35	0.00	4.65	0.00	0.00
4	3.98	0.00	0.00	4.02	0.00
5	3.90	33.88	0.00	2.22	0.00
6	2.66	0.00	0.00	1.34	0.00
7	2.75	0.00	0.00	1.25	0.00
8	5.09	0.00	11.66	2.33	0.92
9	0.00	0.00	5.72	0.95	13.33
10	3.40	0.00	4.60	0.00	0.00
11	3.10	0.00	0.00	1.42	0.00
12	1.91	0.00	0.00	2.09	0.00

seems not to influence the time needed to solve the instance.

Table 5.6: Case 2: Results with $\Gamma_j = 2$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	6.27	8.82	1.30	10.49	0.50
2	4.00	0.00	0.00	0.00	0.00
3	6.04	0.09	1.36	0.00	0.51
4	0.00	0.00	0.00	8.00	0.00
5	0.24	0.28	12.47	0.00	27.01
6	0.25	0.19	2.83	0.00	0.73
7	0.33	0.17	2.52	0.00	0.97
8	5.18	12.26	1.85	0.00	0.71
9	3.27	7.90	1.17	7.22	0.45
10	5.97	0.09	1.35	0.09	0.50
11	0.15	0.08	1.12	0.00	0.43
12	0.22	0.26	2.88	0.00	0.64

Table 5.7: Case 2: Results with $\Gamma_j = 3$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	4.43	4.80	19.18	2.24	0.00
2	1.47	0.00	0.00	2.53	0.00
3	0.82	5.76	0.00	1.42	0.00
4	0.00	0.00	0.00	8.00	0.00
5	19.20	0.00	0.00	0.00	20.80
6	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	4.00	0.00
8	2.38	8.77	0.00	8.86	0.00
9	1.17	4.30	0.00	2.01	12.52
10	0.81	5.79	0.00	1.40	0.00
11	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.00	0.00	0.00

5.5 Single workpiece tracing

It is worth to notice that, differently from the Part Type Selection model, in the Machine Loading model the detailed production for each time period t is chosen. Consequently, considering a disruptive event that affects the whole batch might not reach

Table 5.8: Case 2: Results with $\Gamma_j = 4$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	0.11	3.27	19.18	6.65	0.27
2	0.00	3.70	0.00	0.00	0.30
3	0.00	5.41	0.00	2.14	0.45
4	8.00	0.00	0.00	0.00	0.00
5	8.37	0.00	0.00	0.00	31.63
6	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	4.00	0.00
8	0.00	5.98	0.00	13.53	0.49
9	16.83	2.93	0.00	0.00	0.24
10	0.00	8.00	0.00	0.00	0.00
11	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.00	0.00	0.00

Table 5.9: Case 2: Results with $\Gamma_j = 5$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	9.71	19.18	0.00	0.00	0.00
2	3.22	0.00	0.00	0.78	0.00
3	8.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	8.00
5	0.00	0.00	7.28	32.72	0.00
6	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.30	0.00	3.70
8	0.00	0.00	20.00	0.00	0.00
9	0.00	0.00	0.00	0.00	20.00
10	0.00	0.00	8.00	0.00	0.00
11	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.00	0.00	0.00

a sufficient level of detail. A more desirable formulation would consider the possibility of disruptive events affecting each single workpiece. At this point, one could desire to have a robust formulation able to cover from the possible disruptions he/she expects to impact on a certain number of workpieces, by means of an elongation of their machining time.This corresponds to the

Table 5.10: Case 2: Results with $\Gamma_j = 6$

Product Type	Day 1	Day 2	Day 3	Day 4	Day 5
1	0.00	1.63	8.08	19.18	0.00
2	0.00	4.00	0.00	0.00	0.00
3	8.00	0.00	0.00	0.00	0.00
4	8.00	0.00	0.00	0.00	0.00
5	0.00	6.12	0.00	0.00	33.88
6	0.00	0.00	0.00	0.00	0.00
7	4.00	0.00	0.00	0.00	0.00
8	0.00	0.00	20.00	0.00	0.00
9	8.17	11.83	0.00	0.00	0.00
10	0.00	8.00	0.00	0.00	0.00
11	0.00	0.00	0.00	0.00	0.00
12	0.00	0.00	0.00	0.00	0.00

Table 5.11: Computational times in the deterministic cases

	Time periods	Machines	Cardinality Γ	CPU time [s]
CASE 1	1	3	0	5,13
CASE 2	5	1	0	5,82

Table 5.12: Computational times in the robust cases with the maximum cardinality values adopted

	Time periods	Machines	Cardinality Γ	CPU time [s]
CASE 1	1	3	5	5,17
CASE 2	5	1	6	5,38

assignment of a cardinality to the set of the single workpieces of a specific type i, instead of to the number of part order types. In order to reach this level of detail the model has to be modified, so to include the single units of production, thus becoming:

$$\max \quad \sum_i \sum_t \sum_k w_{it}\, x_{itk} - \sum_i S_i\, C_i \tag{5.37}$$

$$s.t. \quad \sum_j T_{jmt}\, k_j \leq K_m \quad \forall m, t \tag{5.38}$$

$$\sum_t \sum_k x_{itk} = D_i - S_i \quad \forall i \tag{5.39}$$

$$p_{jmt} \leq T_{jmt}\, A_{mt} \quad \forall j, m, t \tag{5.40}$$

$$\sum_i v_{ijt} \leq \sum_m p_{jmt} \quad \forall j, t \tag{5.41}$$

$$\sum_i \sum_k O_{ij}\, x_{itk} + \Gamma_{ijt}\, z_{ijt} + \sum_k q_{ijtk} \leq v_{ijt} \quad \forall i, j, t \tag{5.42}$$

$$z_{ijt} + q_{ijtk} \geq \hat{O}_{ij}\, x_{itk} \quad \forall i, j, t, k \tag{5.43}$$

$$\sum_j p_{jmt} \leq A_{mt} \quad \forall m, t \tag{5.44}$$

$$T_{jmt};\ x_{itk} \in \{0, 1\} \quad \forall j, m, t \tag{5.45}$$

$$S_i;\ p_{jmt};\ z_{ij};\ q_{ijtk};\ v_{ijt} \in \mathbb{R}^+ \tag{5.46}$$

In this new formulation, the decision variables x_{itk} are boolean variables assuming value 0 if workpiece k belonging to the order type i is produced in the time period t, and 0 otherwise. Unfortunately, the solution of this type of formulation requires much more computational power, given the much higher number of 0-1 variables it includes. A test has been performed with 10 worpieces for each of 10 product types, with 1 machine and 1 time period. Using IBM ILOG CPLEX v12.5 on a notebook Dell XPS13 @2.5GHz and 8GB RAM (WIN10), the computational time resulted to be 3 hours and 42 minutes. Therefore, a model with such a formulation can be of no help for a real size production environment, where much bigger dimensions are common.

5.6 Conclusions

A robust version of the Machine Loading model has been developed and tested on some exemplar cases. The robust model provides the production quantities to be produced in each considered time period, as well as the tools to be loaded on the machines, in order to be robust against a certain number of unfortunate events that the production planner is expecting. At this point, we can emphasize the advantage of the Cardinality-Constrained approach: the method in fact only requires a simple input parameter – the cardinality of the disruptions – that can be easily estimated. In fact, white collars are likely able to tell how many events occur in a week or in a month of production; consequently, they can use this knowledge for making reasonable estimations over which cardinality values to adopt when applying the robust models. It has been also proposed a more detailed model, that permits to track each single workpiece thus giving the users one additional degree of freedom in the choice of sets cardinality: unfortunately in this case computational times become an issue, since they turn out to be of the same order of magnitude longer of the time horizon themselves.

Chapter 6

Buffer Allocation Problem

Buffer Allocation is one of the most studied problems in manufacturing: the selection of the proper buffer capacity to be configured on a flow line is directly related to the failures occurring during the service of the production line. Uncertainty on the data regarding the failures of the machines will require the application of a robust approach. In this chapter is proposed how to find the bounds of the Buffer Allocation solution: given a failures pattern scenario, the bounds represent the buffer configurations that cover from the failures that impact the least (lower bound) and from the failures that impact the most (upper bound). This has been done with the attempt to use the Cardinality-Constrained approach following the framework of Bertsimas and Sim.

An overview on the Buffer Allocation Problem and the respective literature is provided in section 6.1. In section 6.2, we present the models for solving the Buffer Allocation Problem that we have taken as reference; in section 6.3, the development of a robust formulation by means of the Cardinality-Constrained framework is pursuited. Then, we give reasons why the method proves to be ineffective in this case. Conclusions are drawn in section 6.4.

6.1 The Buffer Allocation Problem

The Buffer Allocation Problem is a NP-hard optimization problem in the design of production flow lines. The problem concerns finding the optimal buffer sizes to be allocated in predefined areas along the line, in order to meet a specific objective. Having buffer storage allows machines to work independently one another, meaning that blocking and starving phenomena happen with a lower frequency, thus the line has less down-time and the production rate is naturally increased. On the other hand, introducing buffers along a line may prove to be very costly: this is due to the cost of floor space itself, but also to the capital cost of work in progress, that increases together with the buffer space. As a result, the Buffer Allocation is a well-known trade-off that many manufacturing system designers face. Due to its importance, the problem has been studied for more than half century by many researchers, and numerous publications are available in literature.

6.1.1 Problem Description

The problem concerns the allocation of a certain amount of buffer, N, among the $K-1$ intermediate buffer spaces in a flow line with K machines, in order to achieve a specific objective. In the literature, the problem can be usually found in three main forms, depending on which objective functions are to be considered: (1) maximization of the throughput rate of the line (also known as the Dual problem), (2) minimization of the total buffer size (Primal problem), (3) minimization of the average work-in-progress inventory. The objective functions can be expressed in the form of costs or profits, which can also involve monetary criteria since they are of major importance in the production practice.

The most important widely studied problems are the primal and the dual problem, that will be presented in the following sections. In [30] is possible to find a comprehensive analysis of the problems and the related literature.

Primal

The Primal problem deals with the minimization of the total cost of the allocated buffer capacity, constrained by the goal of reaching a minimum throughput of the line. It can be briefly formulated as follows:

$$\min \quad a^T C \tag{6.1}$$
$$s.t \quad E[T] \geq P^* \tag{6.2}$$

where P is the system throughput, P^* is the minimum throughput value and a is a cost vector with the costs of allocating a unitary buffer space; C is the allocated buffer capacity.

Dual

The Dual Buffer Allocation Problem maximises the average system throughput constrained to a maximum budget available for the buffer allocation, thus expressed as:

$$\max \quad E[P] \tag{6.3}$$
$$s.t \quad a^T C \leq a^* \tag{6.4}$$

where a^* is the available budget.

As indicated by Chow [31], the buffer allocation is difficult for two reasons:

- The lack of an algebraic relation between the throughput of the line and the buffer sizes.
- The combinatorial optimization inherent with the problem: for a production line with K machines and N total buffer capacity, the number of possible buffer configurations for the Dual Buffer Allocation Problem is:

$$\binom{N+K-2}{K-2} = \frac{(N+1)(N+2)\cdots(N+K-2)}{(K-2)!} \quad (6.5)$$

  Equation (6.5) gives an idea of the computational difficulty of even small problems, and justifies the use of numerical approaches.

In order to overcome the computational effort issues, many solution techniques have been employed and have been proposed in the literature. A comprehensive survey on the matter has been published by Demir et al. [32].

6.1.2 General solving procedure

Typically, solution approaches to solve the Buffer Allocation Problem involve the application of a generative method and an evaluative method in an iterative way. The evaluative method is used to obtain the value of the objective function for a set of input data, which is then passed to the generative method to search for an optimal solution.

Evaluative methods

Evaluative methods provide the prediction of various performance measures (e.g. the throughput rate and the mean queue lengths). They are based on analytical methods and simulation. The analytical methods can be classified as exact and approximate methods. Exact analytical results are difficult to obtain,

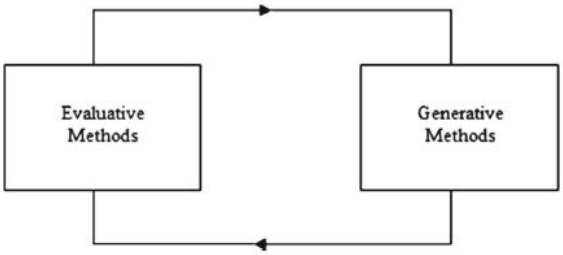


Figure 6.1: Solving procedure for BAP: generative and evaluative methods work in a iterative way

and are only available for short production lines. For long lines, approximate evaluative methods are employed: among them, the most frequently used are the decomposition method, the aggregation method, and the generalized expansion method.

The decomposition method [33] is the most widely used: the idea is to decompose the analysis of the original model into the study of a set of smaller sub-systems that are easier to deal with. The main advantage of this method is its computational efficiency and its accuracy to reach the solution.

The aggregation method has also been successfully employed to evaluate the performance of buffer allocation decisions in unreliable lines. The idea is to replace a two-station-one-buffer line with an equivalent station, which is subsequently combined with another buffer and station, and the process is repeated until the last station is reached [34].

Another approximation method is the generalized expansion method [35]. Similarly to the decomposition method, which has strict assumptions, the generalized expansion method can be used for generally distributed service times and reliable machines and it can be applied both to split and merge configurations and serial configurations.

In comparison to analytical methods, Simulation can present many advantages. The goal is to realistically model a large and complex system. The main disadvantage of simulation is that is

very time consuming.

Generative methods

Generative methods focus on finding optimal buffer sizes to improve the system performance. The simplest method is the complete enumeration; however, since the number of feasible solutions grows exponentially when the buffer size to be allocated increase, in some cases it would be impossible to search through the whole solution space. Therefore, the method is only applicable for small systems.

There are search methods that aim to deal with the abundance of alternative solutions by quickly shifting through many alternative buffer vectors, in order to discover those which yield close to optimal results. Traditional search methods have two main disadvantages: they cannot jump over local optimal solutions in search of global ones, and with these methods it is difficult to observe the effects of small changes in buffer sizes on the system performance. Meta-heuristics are search methods which use strategies that guide the search process and explore the search space in order to find optimal or near-optimal solutions. Usually these methods are approximate and non-deterministic in the search. The main advantage of meta-heuristics is that they can jump over local optimal solutions in search of the global optimal ones. The main disadvantage is that they are not problem specific.

6.2 Mathematical Programming for Simulation

Simulation is a wide-spread method that is used in a lot of environments and corporations in order to study the behavior of a system. Commonly, it is applied in situations when is not pos-

sible to derive closed-form solution equations of the behavior of the system. In fact, one of the main characteristics of simulation is the possibility of predicting the system performance in an implicit way, without requiring the mathematical model of the system. Discrete event simulation is commonly used to analyze the behavior of manufacturing systems with the goal of estimating their major performance measures, such as throughput, resources utilizations, etc. A discrete event system can be defined as a system where one can identify specific events that occur in specific moments. Schruben [42] has proposed an alternative way of modeling discrete event systems, in which the system is mapped in a mathematical programming formulation and the optimal solution represents the trajectory of the discrete events: the behavior of the system is represented by an optimization model in which the sum of finishing and starting events is minimized constrained to the routing of parts – or customers – flowing in the system. Within this framework, optimization and performance evaluation are decoupled: simulation is computing the system performance with a certain configuration; on top of simulation is placed an optimization model that searches for the best configuration of the system. Indeed, here we find the solving technique as depicted in figure 6.1. Simulation is in fact an evaluative method, and the optimization model is a generative method.

In [43], Alfieri and Matta proposed a mathematical programming model of discrete event systems that has the advantage of simultaneously evaluating the system performances and optimizing its configuration. In their work, the authors study production flow lines and propose a mathematical programming formulation to solve the Buffer Allocation Problem.

In the next part of this section, the formulations of the primal and dual Buffer Allocation Problem, as proposed by the authors, will be presented, both in their mixed-integer form and in their

linear approximate form.

6.2.1 Assumptions and performance measures

Let us consider an open flow line composed by K machines separated one another by $K-1$ inter-operational buffers. The sequencing of parts is known a priori: the workpiece i arrives at the system at his arrival time A_i and it is processed sequentially from the first machine till the last one: the processing time of workpiece i on machine j is $t_{i,j}$. Transportation times are assumed to be negligible or already included in the operations times. If a machine M_j is busy, the parts wait at the buffer B_{j-1}. Each buffer B_j has a finite capacity C_j. The Blocking Before Service Control rule is assumed for the machines: a machine will process a workpiece only if there is at least a free slot in the downstream buffer. The first machine is never starved, and the last one is never blocked.

6.2.2 MILP formulation

In this section, the primal and dual buffer allocation problem proposed in [44] will be presented, referring to their mixed-integer form (MILP).

Primal

The primal Buffer Allocation Problem can be formulated as follows:

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} x_{j,k} k \tag{6.6}$$

$$s.t. \quad F_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{6.7}$$

$$F_{i+1,j} - F_{i,j} \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{6.8}$$

$$F_{i,j+1} - F_{i,j} \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{6.9}$$

$$F_{i+k_j,j} - F_{i,j+1} \geq t_{i+k_j,j} x_{j,k} - (1 - x_{j,k})M \quad \forall k, i = 1, ..., N-k_j \quad j = 1, ..., K-1 \tag{6.10}$$

$$\sum_{k=L_j}^{U_j} x_{j,k} = 1 \quad \forall j \tag{6.11}$$

$$F_{N,K} \leq T^* \tag{6.12}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{6.13}$$

$$x_{j,k} \in \{0, 1\} \tag{6.14}$$

In this formulation, $F_{i,j}$ are the finishing times of the i-th part on the j-th machine, $x_{j,k}$ is a binary variable equal to one if a capacity k (with $k = (L_j, ..., U_j)$) is selected for the buffer B_j. L_j and U_j are the bounds defined by the analyst of the problem for the j-th buffer. Constraints (6.7) impose that the service at the first machine cannot finish before the arrival time of the part plus its processing time. Contraint (6.8) impedes that a machine

can process 2 different parts at the same time. Constraint (6.9) state that a part cannot be processed by two machines at the same time. Constraint (6.10) imposes that a part cannot leave a machine if the downstream buffer is full: the buffer allocation is hereby performed thanks to the decision variable $x_{j,k}$: indeed, when $x_{j,k} = 1$ all the constraints related to the j-th buffer with an assigned capacity k are activated; otherwise a large value M is subtracted from the right side, thus making the constraint redundant. Constraint (6.11) imposes that only a capacity k can be chosen for each buffer B_j. Constraint (6.12) imposes a lower bound on the system throughput: this can be understood by noticing that the throughput of the system can be expressed as:

$$P = \frac{N}{F_{N,K}} \tag{6.15}$$

where N is the number of parts that the system is set to produce. Thus, if the number of parts has been already fixed, imposing a limit on the moment the last part will leave the system is equivalent to limiting the throughput from below. Constraints (6.13) and (6.14) impose the non-negativity of finishing time and the boolean nature of the variables $x_{j,k}$.

The model proposed by Alfieri and Matta provides both the performance evaluation of the system, by estimating the maximum flowtime given a buffer space configuration, and the optimization by choosing the right buffer space among machines. It is important to notice that the model provides a solution that is optimal only with respects to the particular sample path, and not in general. Moreover the authors warn that, since the model includes both continuous variables ($F_{i,j}$) and binary ones ($x_{j,k}$), computational times might be an issue.

Dual

The MILP formulation of the dual problem differs from the primal one in the following points:

- The objective function becomes

$$\min \quad F_{N,K} \tag{6.16}$$

  that corresponds to the maximization of the expected production rate, thus replacing the constraint (6.12) of the primal.

- A main system constraint is to be added, that is:

$$\sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} k \cdot x_{j,k} = a^* \tag{6.17}$$

  where a_j is the cost for the allocation of a unit of buffer space in the j-th buffer. This constraint corresponds to the budget constraint on buffer space.

6.2.3 LP approximation

The authors of [43] proposed an alternative way of deactivating the constraints (6.10) by introducing a continuous variable s that can make the constraints redundant when necessary: this can consistently reduce the computational time. The variable is called *time buffer*, and represents the time length a customer can start its processing in advance on a server before the successive one becomes available. The fact that a part can start in advance means an item can begin its process at a stage even if the next one has not freed a buffer slot because it has not yet finished a part; hence, it represents the fact that the next stage has not

yet saturated its buffer. When the weighted sum of the surplus variables is minimized, a fast approximate solution to the Buffer Allocation Problem can be found.

Primal

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} s_{j,k} w_k \tag{6.18}$$

$$s.t. \quad F_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{6.19}$$

$$F_{i+1,j} - F_{i,j} \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{6.20}$$

$$F_{i,j+1} - F_{i,j} \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{6.21}$$

$$F_{i+k_j,j} - F_{i,j+1} \geq t_{i+k_j,j} - s_{j,k} \tag{6.22}$$

$$i = 1, ..., N-k_j \quad j = 1, ..., K-1 \quad \forall k \tag{6.23}$$

$$F_{N,K} \leq T^* \tag{6.24}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{6.25}$$

$$s_{j,k} \geq 0 \quad \forall j, k \tag{6.26}$$

The approximate primal problem is different from its mixed-integer formulation in:

- the objective function: minimizing the time buffer corresponds to minimize the total buffer space; w_k are weights that can be chosen so that a large buffer space is penalized.
- the constraint (6.10) becomes (6.22): the variable $s_{j,k}$ will deactivate it when needed.
- constraint (6.11) is no longer required since in the new formulation the decision variables $s_{j,k}$ are continuous.

In this new formulation, the constraint corresponding to the buffer B_j with capacity k will be deactivated when $s_{j,k}$ are positive means that a buffer of capacity k is necessary. In order to

find the optimal buffer space, a simple heuristic rule is applied: since the larger the $s_{j,k}$ the higher is the importance having a buffer capacity of k, the selected buffer quantity for the j-th buffer will then be the largest index k at which $s_{j,k}$ are positive.

Dual

The dual version of the Buffer Allocation Problem in the LP approximate formulation can be solved with the same changes to the primal as in the MILP formulation presented in section 6.2.2. In this case though, the sum of the time variables must be limited to a threshold value α: the higher the α and the higher the expected production rate will be. As a result, the objective function of the LP approximate dual problem is the same as in equation (6.16) and the following constraint will be added:

$$\sum_{j=1}^{K-1} \sum_{k=L_j}^{U_j} s_{j,k} \leq \alpha \tag{6.27}$$

As in the primal formulation, this approximation of the dual problem, thanks to the introduction of continuous variables $s_{j,k}$ can effectively reduce the computational effort to solve the problem.

6.3 Robust formulation

In this section the attempt to formulate the buffer allocation problem as of section 6.2.2 in accordance to the framework of Bertsimas and Sim will be presented.

6.3.1 Model of data uncertainty

In a flow line, buffer space is allocated in order to cover from system disruptions deriving from failures in the machines com-

posing the line. In our case, the solution of the BAP by means of the Cardinality-Constrained approach implies – given a cardinality Γ to the failures of the machines – finding the solution when the failures we consider are distributed in such a way that their impact on the solution of the problem is maximized. As pointed out in [30], the more variability exist in the failures of the machines in a flow line, the more buffer space is needed. It is interesting to notice that this phenomenon can also be easily verified with the models presented in section 6.2.2 : the more variability is inserted in the technological matrix entries $t_{i,j}$, the more buffer space will be selected. In the integrated simulation-optimization approach as in section 6.2.2, the main input parameter of this formulation of the Buffer Allocation Problem is the processing times $t_{i,j}$. It is now necessary to model the occurrence of failures: it is possible to include the failures of the machines in the variability of the processing times t_{ij}. Thus, let $\bar{t}_{i,j}$ be the nominal duration of the operation on the i-th workpiece on the j-th machine, and $\bar{t}_{i,j} + \hat{t}_{i,j}$ be the operation time when on the j-th machine the processing of the i-th piece has required a longer duration, due to a failure or any sort of event that causes the machine to remain idle for that amount of time (i.e. incorrect positioning of workpieces, man errors, longer time to repair, etc.).

6.3.2 Lower bound

The lower bound corresponds to the solution of the case in which the lowest impact of the machine failures is considered. This can be easily found by writing the primal problem in the following way:

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} s_{j,k} w_k \tag{6.28}$$

$$s.t \quad F_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{6.29}$$

$$F_{i+1,j} - F_{i,j} \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{6.30}$$

$$F_{i,j+1} - F_{i,j} \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{6.31}$$

$$F_{i+k_j,j} - F_{i,j+1} \geq t_{i+k_j,j} - s_{j,k} \tag{6.32}$$

$$\forall k,\ i = 1, ..., N - k_j \quad j = 1, ..., K-1$$

$$\sum_j z_{i,j} = \Gamma_i \quad \forall i \tag{6.33}$$

$$\sum_i z_{i,j} = \zeta_j \quad \forall j \tag{6.34}$$

$$F_N \leq T^* \tag{6.35}$$

$$\bar{t}_{i,j} + z_{i,j}\hat{t}_{i,j} = t_{i,j} \quad \forall i, j \tag{6.36}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{6.37}$$

$$s_{j,k} \geq 0 \quad \forall i, j \tag{6.38}$$

$$t_{i,j} \geq 0 \quad \forall i, j \tag{6.39}$$

$$z_{i,j} \in \{0, 1\} \tag{6.40}$$

In this formulation $t_{i,j}$ are decision variables of the optimization problem, $\bar{t}_{i,j}$ and $\hat{t}_{i,j}$ are the nominal processing times and their maximum fluctuation due to failures, respectively. $z_{i,j}$ are boolean decision variables that are set to select the failures in the constraint (6.36). Constraints (6.33) bound the set of failures that we expect a workpiece will encounter on the machines to

respect the cardinality Γ_i. This means the i-th workpiece during all the time it spends in the system will encounter a maximum number of Γ_i failures (or, if we want, general disruptive events). Similarly, the number of failures that we expect can happen on a generic machine can be bounded by ζ_j in constraint (6.34). With this formulation, the lower bound solution of the Buffer Allocation Problem will be implicitly solved. The reason is that while finding a solution that minimizes the buffer space, the solver will implicitly select the processing times that cause the minimum impact on the solution. It is valuable to notice that the lower bound formulation, differently from the upper bound one, cannot be applied to the MILP formulation of section 6.2.2, since it would result in a nonlinear formulation, for both $t_{i,j}$ and $x_{j,k}$ would be decision variables of the problem. An important remark is the fact that since the solution of the buffer allocation problem depends on the variability of the processing times, this variability will be implicitly minimized in the choice of the processing times in constraint (6.36).

6.3.3 Upper bound

The upper bound corresponds to the situation in which the failures constrained to the desired cardinality have the maximum impact on the solution. Let us first formulate the primal version of the Buffer Allocation Problem in the LP approximate form as proposed in section 6.2.3, so that it can fit within the framework of our robust approach. It is worth to notice that in this LP formulation we are not modeling uncertainty the same way as in section 2.2. In fact, the uncertainty is affecting a known term of the model, and not a coefficient of a decision variable. In the next part we can derive an equivalent formulation in order to fit in the framework of the Cardinality-Constrained approach.

Nominal formulation

In the first step, we must express the problem in the nominal form of a maximization problem, that is:

$$\begin{aligned} \max \quad & \mathbf{c'\,x} \\ s.t. \quad & \mathbf{A\,x} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned} \tag{6.41}$$

Let us then re-write the problem as follows:

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} s_{j,k} \cdot w_k \tag{6.42}$$

$$s.t. \quad \alpha_{i,1} F_{i,1} \leq -1 \quad \forall i \tag{6.43}$$

$$\alpha_{i+1,j}(F_{i+1,j} - F_{i,j}) \leq -1 \quad \forall j, i = 1, ..., N-1 \tag{6.44}$$

$$\alpha_{i,j+1}(F_{i,j+1} - F_{i,j}) \leq -1 \quad \forall i, j = 1, ..., K-1 \tag{6.45}$$

$$\alpha_{i+k,j}(F_{i+k_j,j} - F_{i,j+1}) + \alpha_{i+k,j} s_{j,k} \leq -1 \tag{6.46}$$

$$\forall k, i = 1, ..., N - k_j \quad j = 1, ..., K-1$$

$$F_N \leq T^* \tag{6.47}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{6.48}$$

$$s_{j,k} \geq 0 \quad \forall j, k \tag{6.49}$$

where the $\alpha_{i,j}$ coefficients are defined as:

$$\alpha_{i,j} = -\frac{1}{t_{i,j}} \tag{6.50}$$

and have been derived by simply dividing each term of equations (6.19) - (6.24) by $t_{i,j}$. Since the derived formulation is in the nominal form of a Mathematical Programming model, it is now possible to introduce the model of data uncertainty as in

section 2.2; uncertainty will impact on the coefficients $\alpha_{i,j}$: the coefficients must be able to match the situation in which failures occur. So, let $\bar{\alpha}_{i,j}$ be the nominal value of the coefficient, defined as:

$$\bar{\alpha}_{i,j} = -\frac{1}{\bar{t}_{i,j}} \tag{6.51}$$

and let $\hat{\alpha}_{i,j}$ be the maximum fluctuation of the coefficient, occurring in the event of a failure:

$$\hat{\alpha}_{i,j} = -\frac{1}{\bar{t}_{i,j} + \hat{t}_{i,j}} + \frac{1}{\bar{t}_{i,j}} \tag{6.52}$$

so that when $\alpha_{i,j}$ goes to $\bar{\alpha}_{i,j} + \hat{\alpha}_{i,j}$, it will assume the value $-\frac{1}{\bar{t}_{i,j}+\hat{t}_{i,j}}$.

Case with 2 machines and 2 workpieces

In order to better appreciate the structure of the formulation, the simplest problem will be displayed: this way the size of the matrixes is still small enough to be easily read. In the case of $K = 2$,$N = 2$, $L_j = 1$ and $U_j = 2$, the problem becomes:

$$\mathbf{x} = \begin{bmatrix} F_{1,1} \\ F_{2,1} \\ F_{1,2} \\ F_{2,2} \\ s_{1,1} \\ s_{2,1} \\ s_{1,2} \\ s_{2,2} \end{bmatrix} \tag{6.53}$$

$$\mathbf{A} = \begin{bmatrix} \alpha_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ -\alpha_{2,1} & \alpha_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\alpha_{2,2} & \alpha_{2,2} & 0 & 0 & 0 & 0 \\ -\alpha_{1,2} & 0 & \alpha_{1,2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\alpha_{2,2} & 0 & \alpha_{2,2} & 0 & 0 & 0 & 0 \\ 0 & -\alpha_{2,1} & \alpha_{2,1} & 0 & \alpha_{2,1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.54}$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ T^* \end{bmatrix} \tag{6.55}$$

General case

The formulation for a general size of N pieces and K machines will maintain the same form. where the decision variables vector $\mathbf{x}$ is:

$$\mathbf{x} = \begin{bmatrix} \mathbf{F}_1 \\ \vdots \\ \mathbf{F}_K \\ \mathbf{S}_{L_j} \\ \vdots \\ \mathbf{S}_{U_j} \end{bmatrix} \tag{6.56}$$

where:

$$\mathbf{F}_1 = \begin{bmatrix} F_{1,1} \\ \vdots \\ F_{N,1} \end{bmatrix} \tag{6.57}$$

$$\mathbf{F}_K = \begin{bmatrix} F_{1,K} \\ \vdots \\ F_{N,K} \end{bmatrix} \tag{6.58}$$

$$\mathbf{S}_{L_j} = \begin{bmatrix} s_{1,L_j} \\ \vdots \\ s_{N,L_j} \end{bmatrix} \tag{6.59}$$

$$\mathbf{S}_{U_j} = \begin{bmatrix} s_{1,U_j} \\ \vdots \\ s_{N,U_j} \end{bmatrix} \tag{6.60}$$

The matrix $\mathbf{A}$ is:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{B}_1 & \mathbf{E}_1 \\ \mathbf{A}_2(2) & \mathbf{B}_2 & \mathbf{E}_2 \\ \vdots & \mathbf{B}_3 & \mathbf{E}_3 \\ \mathbf{A}_2(N-1) & \mathbf{B}_4 & \mathbf{E}_4 \\ \mathbf{A}_3(2) & \mathbf{B}_5 & \mathbf{E}_5 \\ \vdots & \mathbf{B}_6 & \mathbf{E}_6 \\ \mathbf{A}_3(K-1) & \mathbf{B}_7 & \mathbf{E}_7 \\ \mathbf{F} & \mathbf{C} & \mathbf{E}_8 \\ \mathbf{D} & 1 & \mathbf{E}_9 \end{bmatrix} \tag{6.61}$$

where:

$$\mathbf{A}_1 = \begin{bmatrix} \alpha_{1,1} & & & \\ & \alpha_{2,1} & & \\ & & \ddots & \\ & & & \alpha_{N,1} \end{bmatrix}$$

$$\mathbf{A}_2(2) = \begin{bmatrix} -\alpha_{2,1} & \alpha_{2,1} & & & \\ & -\alpha_{2,2} & \alpha_{2,2} & & \\ & & \ddots & \ddots & \\ & & & -\alpha_{2,K} & \alpha_{2,K} \end{bmatrix}$$

$$\mathbf{A}_2(N-1) = \begin{bmatrix} -\alpha_{N-1,1} & \alpha_{N-1,1} & & & \\ & -\alpha_{N-1,2} & \alpha_{N-1,2} & & \\ & & \ddots & \ddots & \\ & & & -\alpha_{N-1,K} & \alpha_{N-1,K} \end{bmatrix}$$

$$\mathbf{A}_3(2) = \begin{bmatrix} -\alpha_{1,2} & \alpha_{1,2} & & & \\ & -\alpha_{2,2} & \alpha_{2,2} & & \\ & & \ddots & \ddots & \\ & & & -\alpha_{N,2} & \alpha_{N,2} \end{bmatrix}$$

$$\mathbf{A}_3(K-1) = \begin{bmatrix} -\alpha_{1,K-1} & \alpha_{1,K-1} & & & \\ & -\alpha_{2,K-1} & \alpha_{2,K-1} & & \\ & & \ddots & \ddots & \\ & & & -\alpha_{N,K-1} & \alpha_{N,K-1} \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} -\alpha_{2,1} & \alpha_{2,1} & 0 & \alpha_{2,1} & & & \\ & -\alpha_{3,2} & \alpha_{3,2} & 0 & \alpha_{3,2} & & \\ & & \ddots & \ddots & \ddots & \ddots & \\ & & & -\alpha_{N-K_j,K-1} & \alpha_{N-K_j,K-1} & 0 & \alpha_{N-K_j,K-1} \end{bmatrix}$$

$$\mathbf{B}_1 = zeros[N, N(K + U_j - L_j - 1) + 3]$$

$$\mathbf{B}_2 = zeros[K, N(K + U_j - L_j) - K - 4]$$

$$\mathbf{B}_3 = zeros[K(N-4), N(K + U_j - L_j) - K - 4]$$

$$\mathbf{B}_4 = zeros[K, N(K + U_j - L_j) - K - 4]$$

$$\mathbf{B}_5 = zeros[N, N(K + U_j - L_j - 1) - 4]$$

$$\mathbf{B}_6 = zeros[N(K - 4), N(K + U_j - L_j - 1) - 4]$$

$$\mathbf{B}_7 = zeros[N, N(K + U_j - L_j - 1) - 4]$$

$$\mathbf{F} = zeros[K + 1, 1]$$

$$\mathbf{D}_7 = zeros[1, N(K + U_j - L_j) - 4]$$

$$\mathbf{E}_1 = zeros[N, 3]$$

$$\mathbf{E}_2 = zeros[K, 3]$$

$$\mathbf{E}_3 = zeros[K(N - 4), 3]$$

$$\mathbf{E}_4 = zeros[K, 3]$$

$$\mathbf{E}_5 = zeros[N, 3]$$

$$\mathbf{E}_6 = zeros[N(K - 4), 3]$$

$$\mathbf{E}_7 = zeros[N, 3]$$

$$\mathbf{E}_8 = zeros[K + 1, 3]$$

$$\mathbf{E}_9 = zeros[1, 3]$$

and the vector $\mathbf{b}$ is:

$$\mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \\ -1 \\ -1 \\ \vdots \\ -1 \\ T^* \end{bmatrix} \tag{6.62}$$

For the sake of simplicity we refer from here on to the 2-machine, 2-workpiece problem, since it does not compromise the generality of our considerations.

At this point, it appears possible to apply the Cardinality-Constrained approach to derive a robust solution that covers from the occurrence of failures, which will be causing the α coefficients to vary to $\hat{\alpha}$. The same steps shown in section 2.5 can be done:

$$\sum_j \alpha_{ij}\, x_j + \max_{\Omega} \left\{ \sum_{j \in S_i} \hat{\alpha}_{ij}\, x_j + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{\alpha}_{i,t_j}\, x_j \right\} \leq b_i \quad \forall i \tag{6.63}$$

where

$$\Omega = \{S_i \cup \{t_i\};\ S_i \subseteq J_i;\ |S_i| = \lfloor \Gamma_i \rfloor;\ t_i \in J_i \setminus S_i\} \tag{6.64}$$

represents the set of coefficients undertaking their maximum value. It is now possible to notice a very important characteristic of this formulation, which is also its main limitation.

When reformulating the problem in an outlook such as the one in (6.41), it was necessary to split the decision variables of the problem in the vector $\mathbf{x}$. As a consequence, all the constraints of the problem, which are referring to both indexes i and j, will be expressed in the matrix $\mathbf{A}$, one row at a time. This will cause the matrix to assume such a structure that only one or two of the α coefficients will be present in each row, and in the latter case, the coefficients will be the same one to be found in 2 different positions with the opposite sign. This represents an strong limitation to the robust formulation of the problem. The reason stands in the fact that the robust approach of Bertsimas and Sim considers the cardinality of the set of coefficients in a row of the problem that are subject to uncertainty, thus letting the user choose the number of coefficients he/she expects to vary for each row. In this case, each row will represents a particular couple i, j, that is the location of a single workpiece on a specific machine of the line. Giving a cardinality to the row will then be limited to two choices only: whether we expect a failure to occur at that particular location for a defined workpiece or not. This not only forces the decision maker to choose a very high number of Γ coefficients, but also loses its purpose of selecting which failures causes the worst effect on the system, because only one coefficient will be present to be chosen on each row.

An alternative constraint

In order to avoid the problem just mentioned, it is possible to think a constraint of the kind:

$$\sum_i a_{i,j} F_{i,j} \leq T_j \quad \forall j \tag{6.65}$$

where T_j represents a time budget on the j-th machine and $a_{i,j}$ are coefficients which will need to vary properly representing the

occurrence of a failure at the i-th workpiece on the j-th machine. It is possible to notice that in order to represent the occurrence of a failure on the system, the decision variables $F_{i,j}$ would need to vary consequently the variation of the $a_{i,j}$ coefficients. In order for this to happen, the coefficients shall be linked such as:

$$a_{1,j}(F_{1,j})\, F_{1,j} + \ldots + a_{i,j}(F_{1,j}, \ldots F_{i,j})\, F_{i,j} \leq T_j \quad \forall j \tag{6.66}$$

This formulation is clearly non-linear, and it cannot even be solved with a relaxation because of the explicit link between coefficients and variables.

Splitting the variables

In order to solve the previously exposed situation, one option is to split the decision variables in two parts: $F_{i,j}$ that, as in the nominal formulation, represent the finishing times of workpiece i on machine j, and $y_{i,j}$ representing the increase of the finishing time F due to a failure on the machine j while working on workpiece i. This way the nominal formulation becomes:

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} x_{j,k} k \tag{6.67}$$

$$s.t. \quad F_{i,1} + y_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{6.68}$$

$$F_{i+1,j} + y_{i+1,j} - (F_{i,j} + y_{i,j}) \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{6.69}$$

$$F_{i,j+1} + y_{i,j+1} - (F_{i,j} + y_{i,j}) \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{6.70}$$

$$F_{i+k_j,j} + y_{i+k_j,j} - (F_{i,j+1} + y_{i,j+1}) \geq t_{i+k_j,j} x_{j,k} - (1 - x_{j,k})M \tag{6.71}$$

$$\forall k, i = 1, ..., N - k_j \quad j = 1, ..., K-1$$

$$\sum_{k=L_j}^{U_j} x_{j,k} = 1 \quad \forall j \tag{6.72}$$

$$F_N \leq T^* \tag{6.73}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{6.74}$$

$$y_{i,j} \geq 0 \quad \forall i, j \tag{6.75}$$

$$x_{j,k} \in \{0, 1\} \tag{6.76}$$

In order to obtain a robust formulation, it is possible to add the constraint:

$$\sum_i F_{i,j} + \max_{\Omega} \left\{ \sum_i y_{i,j} \right\} \leq T_j \quad \forall j \tag{6.77}$$

where we can recognize the protection function:

$$\beta_j = \max \quad \sum_i y_{i,j}\, z_{i,j} \tag{6.78}$$

$$s.t. \quad \sum_i z_{i,j} \leq \Gamma_j \tag{6.79}$$

$$0 \leq z_{i,j} \leq 1 \quad \forall i \tag{6.80}$$

This formulation is not enough to correctly express the occurrence of failures. In order to achieve it it is necessary to modify the inner maximization problem so to express the fact that, when a failure occurs it will have the duration of $\hat{t}_{i,j}$, and, when it does not, no additional time is added to the formulation.

$$\beta_j = \max \quad \sum_i y_{i,j}\, z_{i,j} \tag{6.81}$$

$$s.t. \quad \sum_i z_{i,j} \leq \Gamma_j \tag{6.82}$$

$$\hat{t}_{i,j}\, z_{i,j} \leq y_{i,j} \quad \forall i \tag{6.83}$$

$$-M\, z_{i,j} \leq y_{i,j} \quad \forall i \tag{6.84}$$

$$0 \leq z_{i,j} \leq 1 \quad \forall i \tag{6.85}$$

where equation (6.83) forces $y_{i,j}$ to take the value $\hat{t}_{i,j}$ when $z_{i,j} = 1$ and equation (6.84) constrains $y_{i,j}$ to 0 until $z_{i,j} = 0$. M is a very big number.

Let us now write the dual form of the j-th β-maximization problem: at this point, it is useful to write it in explicit form, with the corresponding dual variables π:

$$\begin{aligned}
\min \quad & \Gamma_j\,\pi_1 + (\pi_2 + \cdots + \pi_N) + (\pi_{N+1}\,y_1 + \cdots + \pi_{2N}\,y_N) + \\
& + (\pi_{2N+1}\,y_1 + \cdots + \pi_{3N}\,y_N) && (6.86)\\
s.t. \quad & \pi_1 + \pi_2 + \hat{t}_{1,j}\,\pi_{N+2} + \cdots \geq y{*}_{1,j} && (6.87)\\
& \pi_1 + \pi_3 + \hat{t}_{2,j}\,\pi_{N+2} + \cdots \geq y{*}_{2,j} && (6.88)\\
& \vdots && (6.89)\\
& \pi_{i,j} \geq 0 \quad \forall i && (6.90)
\end{aligned}$$

where $\mathbf{y}^*$ is a particular solution of the master problem. At this point it is possible to see that the dual problem is non-linear. This is due to the fact that in the primal problem there are decision variables $y_{i,j}$ as right side terms, thus belonging to the vector $\mathbf{b}$ of the formulation (6.41). It is possible to think of linearising this problem by replacing:

$$\gamma \leq M\,y \tag{6.91}$$
$$\gamma \geq \hat{t}_i \pi_i - M\,(\hat{t}_i - y^*) \tag{6.92}$$

This substitution comes from observing that the y variables can either take the value 0 or the value $\hat{t}$. Unfortunately the substitution is not enough to obtain a dual problem that correctly represents the inner maximization problem, that meaning the solutions coinciding at the optimum. The reason stands in the fact that by substituting the variables γ the dual problem has a greater number of dual variables than it would have had without the substitution, thus presenting more degrees of freedom than its previous version. The result is that the Strong Duality Theorem is no longer valid: the solution of the dual problem no longer represents the inner optimization problem and the corresponding protection function. To the best of our knowledge,

it is not clear how to further bound the γ variables in a way so to force the dual problem to take the same optimal value of the primal.

6.4 Conclusions

Let us summarize the reasons why we have opted for not applying the Cardinality-Constrained approach in this case:

- The direct application of the approach in the mathematical programming formulas for simulation could not give us the advantage of being able to choose the cardinality for each row of the problem. In fact, only one or two coefficients are present in each row, forcing the decision maker to identify the workpiece and machine where he expects a failure to happen: this is distant from the initial purpose for the application of the approach.
- By writing a constraint to gather more coefficients on a row, those coefficients would depend directly on the decision variables of the master problem: thus, we obtain a non linear formulation.
- In order to solve the previously mentioned difficulties, an alternative formulation would split the variables of the problem, and uses the increase in the processing times as a decision variable. This process however results in a non-linear dual problem, where little can be done to further linearise.

These difficulties shall not discourage from the attempt of finding the bounds of the Buffer Allocation Problem. The limitations are mainly due to the application of the Cardinality-Constrained problem in its analytical form. However, the study was useful to practice the concept of cardinality applied to machine failures. In chapter 7 this will be used in the construction

of a heuristic algorithm to find the bounds that have not been derived in this part.

Chapter 7

A matheuristic approach for the Buffer Allocation Problem

In this chapter an alternative solution to the upper bound of the Buffer Allocation Problem is proposed. The purpose is to find the robust solution to the Buffer Allocation Problem which was not derived for the reasons listed in section 6.4. The scope remains unchanged: finding the failure-pattern – constrained to a specific cardinality – that causes the worst impact on the system, and the buffer space configuration that better covers from it. The possibility of solving this problem derives from the observation that it is possible to input failure patterns in the optimization model (section 7.1). Therefore, it is conceivable to evaluate the objective functions corresponding to each pattern, and from there building the heuristic: a Tabu Search algorithm has been developed, and permits to find an approximate solution to the upper bound (section 7.2). Numerical results of a test case are shown in section 7.3.

7.1 The upper bound case with failures as input

Let us start from the formulation of the Buffer Allocation Problem - lower bound case, as described in section 6.3.2. From here on the LP approximate case will be taken as reference. This is due to the convenience of having results directly comparable to the lower bound case derived in chapter 6. Let us re-write the lower bound formulation:

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} s_{j,k} w_k \tag{7.1}$$

$$s.t \quad F_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{7.2}$$

$$F_{i+1,j} - F_{i,j} \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{7.3}$$

$$F_{i,j+1} - F_{i,j} \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{7.4}$$

$$F_{i+k_j,j} - F_{i,j+1} \geq t_{i+k_j,j} - s_{j,k} \quad \forall k, i = 1, ..., N-k_j \quad j = 1, ..., K-1 \tag{7.5}$$

$$F_N \leq T^* \tag{7.6}$$

$$\sum_j z_{i,j} = \Gamma_i \quad \forall i \tag{7.7}$$

$$\sum_i z_{i,j} = \zeta_j \quad \forall j \tag{7.8}$$

$$\bar{t}_{i,j} + z_{i,j}\hat{t}_{i,j} = t_{i,j} \quad \forall i, j \tag{7.9}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{7.10}$$

$$s_{j,k} \geq 0 \quad \forall i, j \tag{7.11}$$

$$t_{i,j} \geq 0 \quad \forall i, j \tag{7.12}$$

$$z_{i,j} \in \{0, 1\} \tag{7.13}$$

In this formulation, the decision variables $z_{i,j}$ are selecting the failure-pattern of the system. In the lower bound formulation, this is a decision variable of the problem. Then, we can think to

input a generic matrix $\hat{z}_{i,j}$ with a predefined pattern of failures. The model becomes:

$$\min \quad \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} s_{j,k} w_k \tag{7.14}$$

$$s.t \quad F_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{7.15}$$

$$F_{i+1,j} - F_{i,j} \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{7.16}$$

$$F_{i,j+1} - F_{i,j} \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{7.17}$$

$$F_{i+k_j,j} - F_{i,j+1} \geq t_{i+k_j,j} - s_{j,k} \quad \forall k, i = 1, ..., N-k_j \quad j = 1, ..., K-1 \tag{7.18}$$

$$F_N \leq T^* \tag{7.19}$$

$$\bar{t}_{i,j} + \hat{z}_{i,j}\hat{t}_{i,j} = t_{i,j} \quad \forall i, j \tag{7.20}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{7.21}$$

$$s_{j,k} \geq 0 \quad \forall i, j \tag{7.22}$$

$$t_{i,j} \geq 0 \quad \forall i, j \tag{7.23}$$

Notice that in this case equations (7.7) and (7.8) are not present, since $\hat{z}_{i,j}$ are now input parameters of the problem and it is not necessary to impose constraints on them. However, the cardinality of the failures has to be decided upfront to respect the conditions:

$$\sum_j \hat{z}_{i,j} = \Gamma_i \quad \forall i \tag{7.24}$$

$$\sum_i \hat{z}_{i,j} = \zeta_j \quad \forall j \tag{7.25}$$

This way it is possible to find the Buffer Allocation solution corresponding to a specific failures' pattern in the line.

7.2 The proposed algorithm

We have shown that it is possible to derive the solution of a Buffer Allocation Problem when the failure-pattern of the system is known. Ergo, it is possible to conceive an algorithm to find the failure-patterns that cause the worst impact on the system: those are the failures arranged in such a way so that the maximum buffer space is needed. Moreover, we can use the concept of cardinality to bound the total number of failures to a desired value. Notice that by doing so we let the decision maker decide the level of protection he/she wants to adopt: indeed, when a higher number of failures is expected to take place we aim at a higher level of robustness, and vice-versa. Let us arrange the optimization problem in this way:

$$\min \quad W^*(l) = \sum_{j=1}^{K-1} a_j \sum_{k=L_j}^{U_j} s_{j,k} w_k \tag{7.26}$$

$$s.t \quad F_{i,1} \geq A_1 + t_{i,1} \quad \forall i \tag{7.27}$$

$$F_{i+1,j} - F_{i,j} \geq t_{i+1,j} \quad \forall j, i = 1, ..., N-1 \tag{7.28}$$

$$F_{i,j+1} - F_{i,j} \geq t_{i,j+1} \quad \forall i, j = 1, ..., K-1 \tag{7.29}$$

$$F_{i+k_j,j} - F_{i,j+1} \geq t_{i+k_j,j} - s_{j,k} \quad \forall k, i = 1, ..., N-k_j \quad j = 1, ..., K-1 \tag{7.30}$$

$$F_N \leq T^* \tag{7.31}$$

$$\bar{t}_{i,j} + \hat{z}^l_{i,j} \hat{t}_{i,j} = t_{i,j} \quad \forall i, j \tag{7.32}$$

$$F_{i,j} \geq 0 \quad \forall i, j \tag{7.33}$$

$$s_{j,k} \geq 0 \quad \forall i, j \tag{7.34}$$

$$t_{i,j} \geq 0 \quad \forall i, j \tag{7.35}$$

where $W^*(l)$ is the objective function of the l-th generic iteration with a corresponding failure input matrix $\hat{z}^l_{i,j}$. At each iteration, the solution is found the same way as in chapter 6: that is, for

each j-th buffer, the allocated space is defined as the index of the last positive value of vector $s_{j,k}$. Notice that for simplicity we have omitted to write the iteration index l for the decision variables of the problem. Formally, they should be $F^l_{i,j}$, $t^l_{i,j}$ and $s^l_{j,k}$ since in each iteration a different optimization problem is being solved.

Figure 7.1 shows a graphical overview of the algorithm, which has been written in Matlab ® code (Appendix A). The algorithm is structured in the following way:

Step 0

The algorithm starts with an initial matrix $\hat{z}^{(0)}$of values $\hat{z}^{(0)}_{i,j}$. The matrix is compiled with binary values that respect the cardinality constraints as in equations (7.7) and (7.8) (the initialization function code is available in appendix A). At this step, all the variables (buffer space and objective function) are initialized to 0. At the first iteration ($l = 0$), we skip from step 0 to step 2.

Step 1

Step 1 evaluates the objective function and the buffer space. These results are obtained by solving the Mathematical Programming problem (7.26): this is done by implementing the equations in IBM CPLEX ® v12.5. It is interesting to notice that we can recognize again the sub-division between evaluative and generative methods. In our technique, the Matlab routine interfaces with CPLEX by an input-output Excel file, where the failure matrix $\hat{z}^{(l)}$ is written and serves as input for the solver. The same file is used to save the results at each iteration: indeed, the objective function value $eval(l)$ and the buffer space are being read by the Matlab routine at each iteration. If the calculations reported an increase in the objective function, the routine goes to step 2: the improvement point is *"tagged"* with

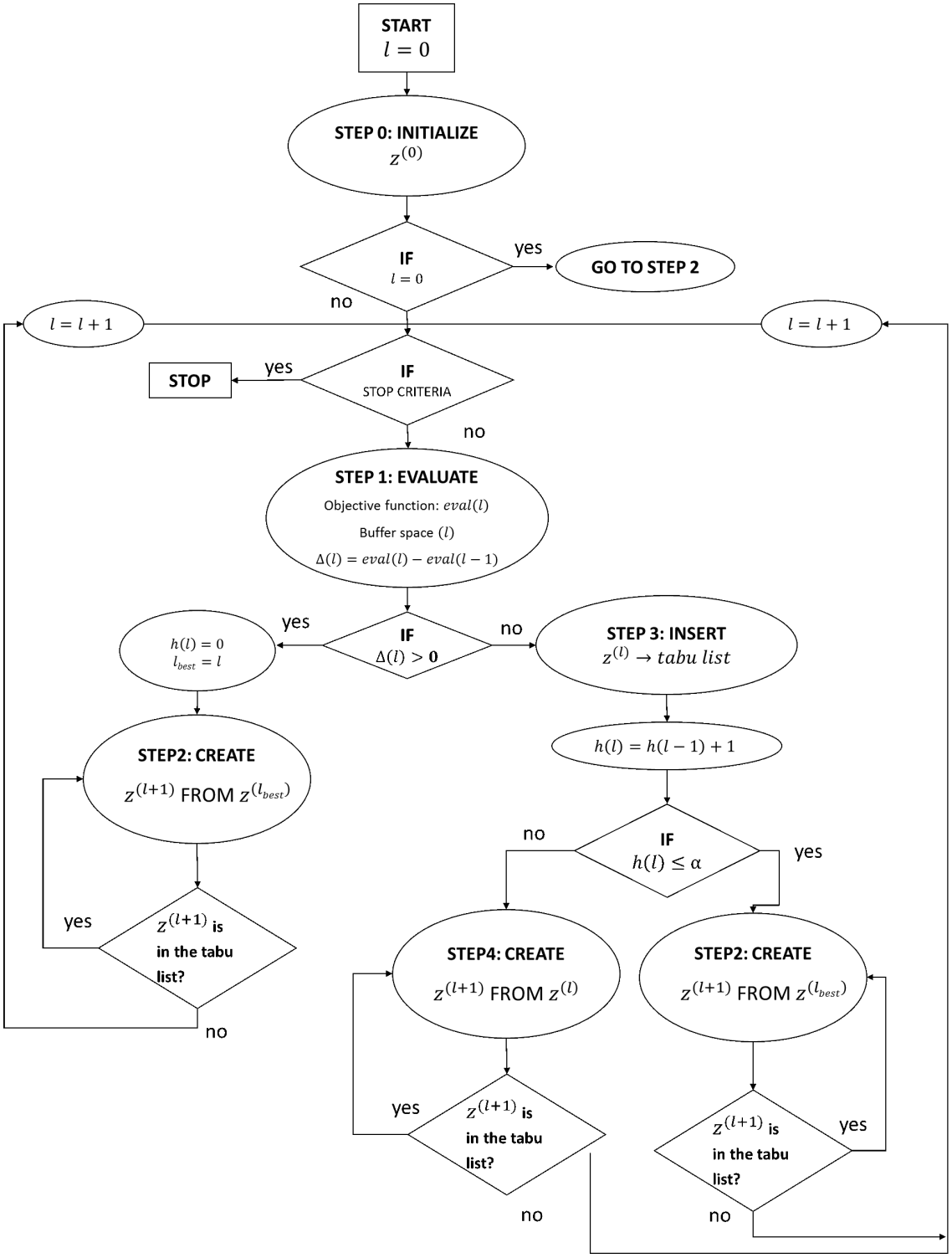


Figure 7.1: The proposed Tabu Search algorithm

l_{best}, and the counter h is set to 0. If no improvement is observed, we procede with step 3.

Creation Function

The creation of the failures' matrix for the next iteration is done by the creation function (section A.2): given an input matrix $\hat{z}_{i,j}^{(l)}$, this function switches between a 1 and a 0 along a randomly chosen column of the matrix, thus returning $\hat{z}_{i,j}^{(l+1)}$, that is the input matrix for the next iteration. Notice that in this operation the cardinality constraint on a column (7.25) are always satisfied. In this case we do not bound the cardinality on the rows, so we do not use constraint (7.24).

Step 2

Step 2 generates the failures matrix for the next iteration $\hat{z}_{i,j}^{(l+1)}$ with the creation function. As input, it uses $\hat{z}_{i,j}^{(l_best)}$, that is the matrix with which the last improvement has been found. Subsequently, it is checked whether the newly generated matrix is already in the *"Tabu list"*. The list contains all the matrixes that returned objective function values worse than previous iterations. If the check is positive, a new generation is done.

Step 3

Step 3 saves the matrix $\hat{z}_{i,j}^{(l)}$ in the *"Tabu list"*, since no improvement has been obtained with those input values. Moreover, the counter h is raised by 1. The counter permits to avoid local optimums. In particular, if h is less or equal a threshold value α, the algorithm runs again over step 2. Otherwise, it procedes with step 4.

Step 4

In step 4, a new failures matrix is created by means of the creation function. However, at this step the function uses as input the matrix $\hat{z}_{i,j}^{(l)}$, that is the failures' matrix relative to this specific iteration.

Stop criteria

Figure 7.2 shows the objective function results in a test-run. Over 1500 iterations, the algorithm was able to find better solutions until the 206th iteration. In the following iterations, no better solutions were found.

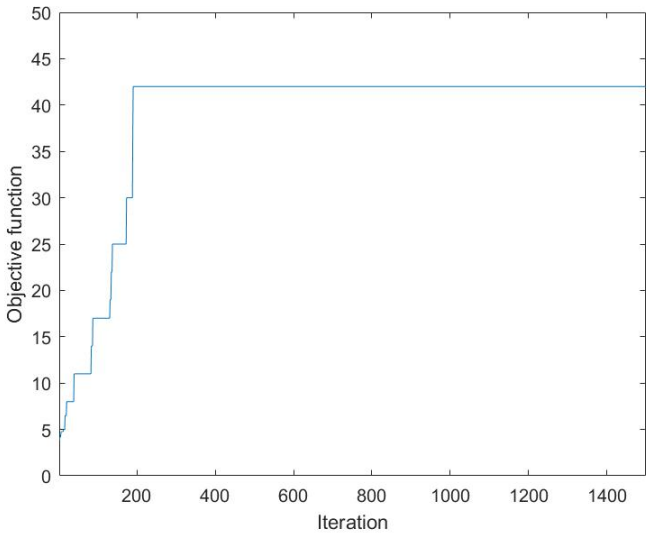


Figure 7.2: The objective function solutions of the Tabu Search algorithm in a test run with 1500 iterations

From the observation of the behavior of the objective function we can think of adding a stop criteria to the algorithm. In fact, since over 1500 iterations no improvement has been noticed (other runs have confirmed this behavior as well), it is conceivable to stop the algorithm after it has not found better solutions for a certain number of iterations. In figure 7.3, Φ is a

parameter that indicates when the routine can be stopped. The iteration at which we break the iteration loop is l_{stop}. Notice that $l_{best} + \Phi = l_{stop}$. It is important to remark that the stop criteria depends on the dimensions of the problem: indeed, in a bigger problem there are more possible combinations of failures, making it appropriate to use higher values of Φ.

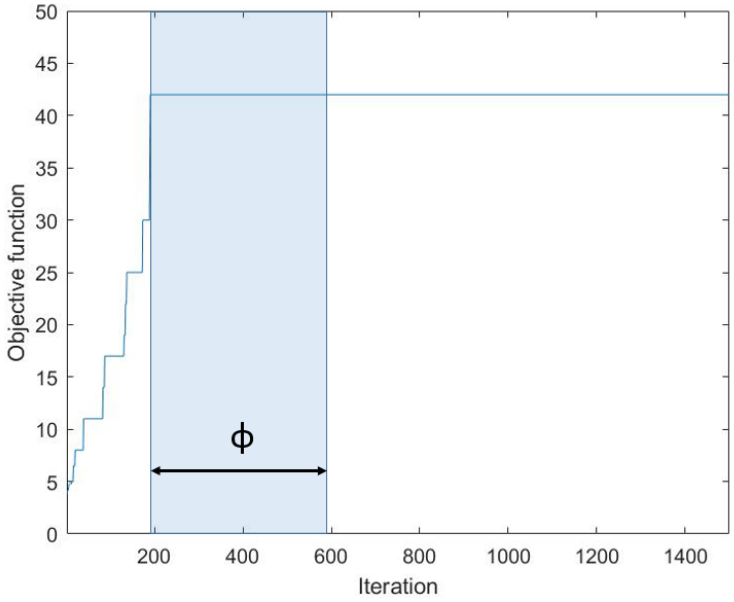


Figure 7.3: The objective function solutions of the Tabu Search algorithm in a test run with 1500 iterations

7.3 Numerical results

A test has been done in the case of a flow line with 4 machines and 3 buffers (figure 7.4) producing 1000 workpieces. The processing times $\bar{t}_{i,j}$ have been set to 1 min per workpiece, and their deviations $\hat{t}_{i,j}$ to 0.5 min per workpiece. The throughput constraint has been set to $T = 1015\,min$, corresponding to a desired throughput of 97.5%. For the sake of simplicity, the cost of each buffer slot has been set to 1 \$. Since it is the only cost in the objective function, its value does not need to be specific.

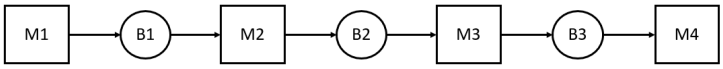


Figure 7.4: The flow line considered for the test

The cardinality values represent how many failures will affect the workpieces on the j-th machine; those are the Γ_j that satisfy $\sum_i z_{i,j} = \Gamma_j$, and have been set to:

$$\Gamma = \begin{bmatrix} 10 & 10 & \Gamma_3 & 10 \end{bmatrix} \tag{7.36}$$

where Γ_3 is kept as a parameter so to be able to evaluate different scenarios: this is due to the aim to obtain the behavior of the solution when cardinality values vary, which is one of the main purposes throughout this work. Therefore, the algorithm has been run for different cardinality values of the failures on machine M3. Figure 7.5 shows the total buffer space found for each respective cardinality value. The lower bound total buffer space remains unchanged in all cases, whereas the upper bound solutions vary together with the increase of the failures' cardinality. The graph shows evidence about the fact that buffer space in the upper bound case grows only after the cardinality of the failures at machine M3 becomes greater than the ones from the remaining machines, that is always kept at the constant level of 10, as in (7.36). In fact, when cardinality becomes greater than 10, M3 turns to be the bottleneck of the system, since the total time of production on it is longer than all the other machines' production times. Therefore, in order to avoid blockages, more buffer space is added over all the line, as it can be seen from table 7.1. Notice that the buffer space found by the algorithm does increase, but it does not follow a specific pattern (notice for example the buffer space shape between cardinality values of 16 and 17): this is due to the approximate nature of the Tabu Search heuristic, that is able to find a good

solution in reasonable times, although not guaranteeing that it is a global optimum. Appendix B shows details about the failures'configurations found for each cardinality value by running the algorithm.

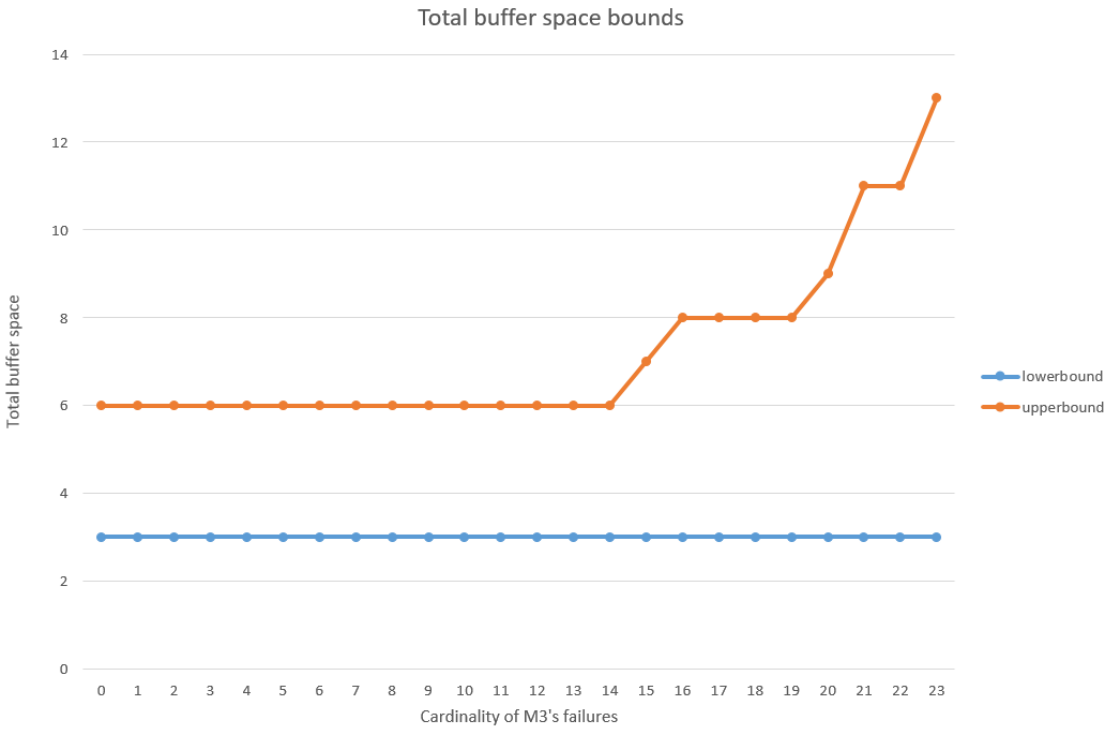


Figure 7.5: Comparison between lower and upper bound with respect to the cardinality of the failures of M3

7.3.1 Computational times

Table 7.2 lists the computational times for obtaining the lower bound solution, together with the corresponding gap. The iterations required in this case are always equal to 1 simply because the lower bound only requires one, and the failure-pattern is obtained by solving the optimization problem itself (section 7.1). Table 7.3 shows the computational times related to the upper bound case: several iterations are required, therefore the time to

Table 7.1: Comparison of the results obtained in the lower and upper bound cases

	Lower Bound				Upper Bound			
Γ_3	B1	B2	B3	Total	B1	B2	B3	Total
0	1	1	1	3	2	2	2	6
1	1	1	1	3	2	2	2	6
2	1	1	1	3	2	2	2	6
3	1	1	1	3	2	2	2	6
4	1	1	1	3	2	2	2	6
5	1	1	1	3	2	2	2	6
6	1	1	1	3	2	2	2	6
7	1	1	1	3	2	2	2	6
8	1	1	1	3	2	2	2	6
9	1	1	1	3	2	2	2	6
10	1	1	1	3	2	2	2	6
11	1	1	1	3	2	2	2	6
12	1	1	1	3	2	2	2	6
13	1	1	1	3	2	2	2	6
14	1	1	1	3	2	2	2	6
15	1	1	1	3	2	3	2	7
16	1	1	1	3	3	2	3	8
17	1	1	1	3	2	3	3	8
18	1	1	1	3	2	3	3	8
19	1	1	1	3	2	3	3	8
20	1	1	1	3	2	4	3	9
21	1	1	1	3	4	4	3	11
22	1	1	1	3	2	4	5	11
23	1	1	1	3	3	5	5	13

solve the instance is higher. Figure 7.6 shows the CPU time expressed in hours for different cardinality values Γ_3. As it can be seen from the graph, the computational time required is longer with higher cardinality values, as well as with a higher number of iterations. Nevertheless, let us remind that this is also due to different values of Φ that have been used. Moreover, since the algorithm randomly permutes the matrix $\hat{z}_{i,j}$, a variability in the computational times remains natural.

Despite the times are high, this does not discourage the application of our simple heuristic. In fact, let us remind the following: (1) theoretically, once a protection level is chosen, only one cardinality value has to be used, so the algorithm would run only once, or anyway not an excessive number of times; (2) we have not used techniques to reduce the CPU time, even though there are many that proved to be successful, such as the use of parallel cores; (3) the algorithm solves a Buffer Allocation Problem, to which in general is dedicated a sufficient amount of time, since it is part of the designing phase of a flow line.

Table 7.2: Lower bound case: computational times

Γ_3	n_{iter}	CPU time [s]	GAP[%]
0	1	80	0,32
1	1	80	0.30
2	1	80	0.27
3	1	80	0.24
4	1	80	0.20
5	1	80	0.17
6	1	80	0.14
7	1	80	0.10
8	1	80	0.70
9	1	80	0.40
10	1	80	0.00
11	1	80	0.20
12	1	80	0.40
13	1	80	0.50
14	1	80	0.74
15	1	160	0.10
16	1	80	0.10
17	1	80	0.17
18	1	200	0.19
19	1	200	0.20
20	1	200	0.23
21	1	200	0.28
22	1	200	0.31
23	1	200	0.31

Table 7.3: Upper bound case: computational times

Γ_3	n_{iter}	l_{stop}	l_{best}	CPU time [s]	CPU time [h]
0	1500	559	97	4208	1.2
1	1500	476	225	12193	3.4
2	1500	290	39	2427	0.7
3	1500	452	201	11646	3.2
4	1500	462	211	3816	1.1
5	1500	547	296	14439	4.0
6	1500	646	395	7021	2.0
7	1500	420	169	10503	2.9
8	1500	570	319	6462	1.8
9	1500	487	236	14992	4.2
10	1500	423	172	5765	1.6
11	1500	384	133	13924	3.9
12	1500	425	174	6957	1.9
13	1500	345	94	13980	3.9
14	1500	378	127	11096	3.1
15	1500	398	147	18049	5.0
16	3000	3000	148	53091	14.7
17	1500	867	366	15309	4.3
18	1500	405	154	18932	5.3
19	1500	470	219	23925	6.6
20	1500	1500	1327	17697	4.9
21	3000	3000	748	30349	8.4
22	3000	3000	985	121602	33.8
23	3000	1719	716	89093	24.7

7.4 Conclusions and remarks

This chapter has introduced an algorithm that can be used to derive the upper bound of the BAP. The computational times are in the order of 80 to 200 seconds for the lower bound case, in which only one run is required but of a MILP problem, and between 5 to 7 hours for the upper bound case with 1500 iterations. Those times are high but not problematic, since the application is a design problem that normally requires a long

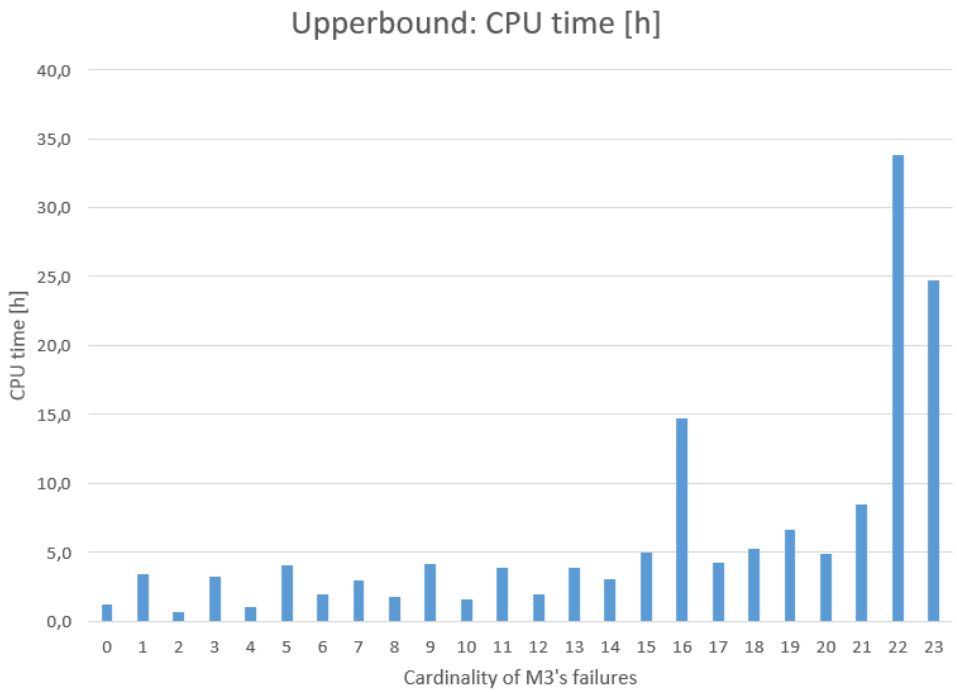


Figure 7.6: CPU time for the upper bound case, expressed in hours

time anyway. Moreover, it is certainly possible to improve the computational times. Different paths could be undertaken: parallelization seems to be the most promising, but in future developments other options could be considered as well, such as the development of a "*smarter*" creating function. For example, since the failures are a relatively small number in comparison with the workpieces flowing in the line, some patterns could produce equivalent results. The corresponding input matrixes could be excluded from the possible outcomes of the failure-patterns creating function. Alternatively, the algorithm shall recognize improvement directions. For these reasons we believe the proposed heuristic represents a promising approach to derive approximate bounds of the buffer configuration solution.

Chapter 8

Conclusions

The literature analysis done in this work shows that interest in finding efficient methods to deal with uncertainty in the manufacturing area is high. Regarding this work, the robust versions of the problems we have taken into account gave promising results both in terms of applicability and computational times.

In chapter 3, a robust model for the assessment of a Part Type Selection has been developed, by means of its integration with the Cardinality-Constrained approach. The results of tests on case studies proved to be aligned with the expected behavior of a Cardinality-Constrained robust model: the decrease of the objective function value with an increase in the cardinality. Once the relationship between objective function and cardinality can be measured, a decision maker can evaluate the cost of a certain number of disruptive events on the production campaign. Therefore, the proposed approach is apposite for an application in the manufacturing area. The application of the robust Part Type Selection model to a real case and the corresponding results are shown in chapter 4. The analysis of the results motivated the proposal of using alternative weights (i.e. dependent on results obtained with lower cardinality values) in order to obtain batching results more pertinent to a real situation. In particular, with the new formulation, the results are

easier to understand by inexperienced users.

The robust version of a Machine Loading model has been proposed in chapter 5, with the scope of providing an effective tool to help the Production Planning and tooling the machines over a specific time horizon. A further detailed model, that permits to track each single workpiece has also been suggested: unfortunately, in this case computational times constitute an issue.

The application of the Cardinality-Constrained method on the Buffer Allocation Problem resulted in a non-linear formulation when using the analytical expressions of the approach (Chapter 6). Since linearity is one of the most relevant and innovative advantage of the Cardinality-Constrained approach, we did not further develop the formulation, and opted instead to use the concept of cardinality in building a Tabu Search algorithm (Chapter 7) that could find the approximate bounds of the buffer configuration for a flow line in a test case. The computational times are in the order of few minutes for the lower bound case, in which only one run is required, and between 5 to 7 hours for the upper bound case with 1500 iterations. Those times are high but not problematic, since the Buffer Allocation is a problem to which is normally dedicated a reasonable amount of time.

As to future work, the developments of this research shall aim at reducing the computational effort in the cases we have indicated it is an issue. The robust version of the Machine Loading Problem, when tracing the single workpieces, proves to be too much time-demanding. In this case, standard approaches to reduce computational effort in Mathematical Programming problems can be endeavored. Moreover, we are confident in the possibility to improve the computational times for the matheuristic approach we developed for the BAP: parallelization seems to be the most promising, but other options can be considered as well,

such as the development of a *smarter* failure-patterns creating function. Bigger-sized problems shall be taken into consideration, aiming at a broader applicability of the method. Furthermore, a deeper analysis is required in order to find the degree of approximation of the bounds we have found. In order to do so, we have to compute the solution for all the combinations, and consequently compare the results with the approximate bounds that we already possess. Our future efforts will go in this direction.

Bibliography

[1] J.F. Shapiro. Chapter 8 mathematical programming models and methods for production planning and scheduling. *Handbooks in Operations Research and Management Science*, 4(C):371–443, 1993.

[2] S.C. Graves. Uncertainty and production planning. *International Series in Operations Research and Management Science*, 151:83–101, 2011.

[3] A. Sethi and S.P. Sethi. Flexibility in manufacturing: A survey. *International Journal of Flexible Manufacturing Systems*, 2:289–328, 1990.

[4] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004. cited By 0.

[5] J. R. Birge and F. Louveaux. Introduction to stochastic programming, 1997.

[6] D. Bertsimas, D.B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011.

[7] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming, 1973.

[8] A. Ben-Tal and A. Nemirovski. Robust solutions of linear

programming problems contaminated with uncertain data, 2000.

[9] O.L.V. Costa and A.C. Paiva. Robust portfolio selection using linear-matrix inequalities. *Journal of Economic Dynamics and Control*, 26(6):889 – 909, 2002.

[10] R.H. Tutuncu and M. Koenig. Robust asset allocation. *Annals of Operations Research*, 132(1-4):157–187, 2004.

[11] G. Calafiore and L. El Ghaoui. Robust maximum likelihood estimation in the linear model. *Automatica*, 37(4):573–580, 2001.

[12] Y.C. Eldar, A. Ben-Tal, and A. Nemirovski. Robust mean-squared error estimation in the presence of model uncertainties. *IEEE Transactions on Signal Processing*, 53(1):168–181, 2005.

[13] D. Bertsimas and A. Thiele. A robust optimization approach to supply chain management. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3064:86–100, 2004. cited By 24.

[14] A. Ben-Tal and A. Nemirovski. Robust truss topology design via semidefinite programming. *SIAM Journal on Optimization*, 7(4):991–1016, 1997.

[15] S.P. Boyd, S.-J. Kim, D.D. Patil, and M.A. Horowitz. Digital circuit optimization via geometric programming. *Operations Research*, 53(6):899–932, 2005.

[16] D. Patil, S. Yun, S.-J. Kim, A. Cheung, M. Horowitz, and S. Boyd. A new method for design of robust digital circuits. pages 676–681, 2005.

[17] K.-L. Hsiung, S.-J. Kim, and S. Boyd. Power control in lognormal fading wireless channels with uptime probability specifications via robust geometric programming. volume 6, pages 3955–3959, 2005.

[18] R.G. Lorenz and S.P. Boyd. Robust minimum variance beamforming. *IEEE Transactions on Signal Processing*, 53(5):1684–1696, 2005.

[19] A. Mutapcic, S.-J. Kim, and S. Boyd. Beamforming with uncertain weights. *IEEE Signal Processing Letters*, 14(5):348–351, 2007.

[20] D. Bertsimas and D.B. Brown. Constrained stochastic lqc: A tractable approach. *IEEE Transactions on Automatic Control*, 52(10):1826–1841, 2007.

[21] D. Bertsimas, O. Nohadani, and M.T. Kwong. Robust optimization in electromagnetic scattering problems. *Journal of Applied Physics*, 101(7), 2007.

[22] M.C.O. Moreira, J.-F. Cordeau, A.M. Costa, and G. Laporte. Robust assembly line balancing with heterogeneous workers. *Computers and Industrial Engineering*, 88:254–263, 2015.

[23] L. Borba and M. Ritt. A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. *Computers and Operations Research*, 45:87–96, 2014. cited By 14.

[24] D. J. Alem and R. Morabito. Production planning in furniture settings via robust optimization. *Computers and Operations Research*, 39(2):139–150, 2012. cited By 21.

[25] O. Hazir, E. Erel, and Y. Gnalay. Robust optimization models for the discrete time/cost trade-off problem. *Inter-*

national Journal of Production Economics, 130(1):87–95, 2011. cited By 14.

[26] C.-C. Lu, K.-C. Ying, and S.-W. Lin. Robust single machine scheduling for minimizing total flow time in the presence of uncertain processing times. *Computers and Industrial Engineering*, 74(1):102–110, 2014. cited By 4.

[27] O. Solyali, J.-F. Cordeau, and G. Laporte. Robust inventory routing under demand uncertainty. *Transportation Science*, 46(3):327–340, 2012. cited By 10.

[28] O. Hazir and A. Dolgui. Assembly line balancing under uncertainty: Robust optimization models and exact solution method. *Computers and Industrial Engineering*, 65(2):261–267, 2013. cited By 17.

[29] Y. Moon and T. Yao. A robust mean absolute deviation model for portfolio optimization. *Computers and Operations Research*, 38(9):1251–1258, 2011. cited By 12.

[30] S.B. Gershwin and J.E. Schor. Efficient algorithms for buffer space allocation. *Annals of Operations Research*, 93(1-4):117–144, 2000. cited By 117.

[31] We-Min Chow. Buffer capacity analysis for sequential production lines with variable process times. *International Journal of Production Research*, 25(8):1183–1196, 1987. cited By 49.

[32] L. Demir, S. Tunali, and D.T. Eliiyi. The state of the art on buffer allocation problem: A comprehensive survey. *Journal of Intelligent Manufacturing*, 25(3):371–392, 2014. cited By 6.

[33] Stanley B. Gershwin. *Manufacturing Systems Engineering.* 1993.

[34] A. Dolgui, A. Eremeev, A. Kolokolov, and V. Sigaev. A genetic algorithm for the allocation of buffer storage capacities in a production line with unreliable machines. *Journal of Mathematical Modelling and Algorithms*, 1(2):89–104, 2002.

[35] D. Spinellis, C. Papadopoulos, and J.M. Smith. Large production line optimization using simulated annealing. *International Journal of Production Research*, 38(3):509–541, 2000.

[36] R. Jaikumar. Postindustrial manufacturing. *Harvard Business Review*, 1986.

[37] S.S. Hwang and A.W. Shogan. Modelling and solving an fms part selection problem. *International Journal of Production Research*, 27(8):1349–1366, 1989.

[38] Kathryn E. Stecke. Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. *Management Science*, 29(3):273–288, 1983.

[39] A. Grieco, Q. Semeraro, and T. Tolio. A review of different approaches to the fms loading problem. *International Journal of Flexible Manufacturing Systems*, 13(4):361–384, 2001.

[40] M.S. Sodhi, R.G. Askin, and S. Sen. Multiperiod tool and production assignment in flexible manufacturing systems. *International Journal of Production Research*, 32(6):1281–1294, 1994.

[41] K. Das, M.F. Baki, and X. Li. Optimization of operation and changeover time for production planning and scheduling in a flexible manufacturing system. *Computers and Industrial Engineering*, 56(1), 2009.

[42] Lee W. Schruben. Mathematical programming models of discrete event system dynamics. volume 1, pages 381–385, 2000.

[43] A. Alfieri and A. Matta. Mathematical programming formulations for approximate simulation of multistage production systems. *European Journal of Operational Research*, 219(3):773–783, 2012.

[44] A. Matta. Simulation optimization with mathematical programming representation of discrete event systems. pages 1393–1400, 2008. cited By 17.

Appendix A

Matheuristic code

In this appendix is reported the Matlab ® code used for the implementation of the Tabu Search algorithm described in chapter 5.

A.1 Main

```
close all
clearvars

% input file excel for interface
filename = 'C:\Users\Giovanni\opl\LPhuge\mille.xlsx';
sheet = 1;

% input cardinality constraints
gamma=[10 10 21 10];

% input iterations
n_iter=3000;

% input other data
nmach=4;
npieces=1000;
T=1025;
T_excel_pos='M2';
threshold=20;
limit=3000;
lowerbound=2.97;
of_excel_pos='E2';
s_excel_pos='G2:AJ4';
cplex_path = 'oplrun -p C:\Users\Giovanni\opl\LPhuge';

xlswrite(filename,T,sheet,T_excel_pos);

counter=0;
jbetter=0;
indextabu=1;
deltae=0;
c=1;
ub=1;

colstart=char(c+'A'-1);
colfinish=char(nmach+'A'-1);
rowstart=num2str(2);
rowfinish=num2str(npieces+1);
z_excel_pos=[colstart, rowstart,':',colfinish, rowfinish];

% insert cardinality
```

```
z0=card(npieces,nmach,gamma);

t=zeros(npieces, nmach);
of=zeros(1,n_iter);
of1=zeros(1,n_iter);
ofgood=zeros(1,n_iter);
z0(:,:,1)=z0;

ztabu=zeros(npieces,nmach,indextabu);
z_upperbound=zeros(npieces,nmach,n_iter);
buffer=zeros(n_iter,nmach-1);

for j=1:n_iter

%write failure matrix z on excel interface file
sheet = 2;
xlswrite(filename,z0(:,:,j),sheet,z_excel_pos);

% run cplex code
status = dos(cplex_path);

%read results
sheet=3;
of1(j)= xlsread(filename, sheet, of_excel_pos);
s = xlsread(filename, sheet, s_excel_pos);
buffer(j,:)=bufferspace(s);

of(j)=of1(j)+10*sum(buffer(j,:));

%check whether to continue (Tabu Search)
if j==1
        deltae=of(j)-deltae;
else
        deltae=of(j)-of;
end

if  ~any(deltae<0)

        counter=0;
        jbetter=j;
        ofgood(j)=of(j);
        z_upperbound(:,:,ub)=z0(:,:,j);
        ub=ub+1;

if indextabu==1
        div=ones(1,indextabu);
else
        div=ones(1,indextabu-1);
end

while any(div)
        z0(:,:,j+1)=replace_new(z0(:,:,j));

        if indextabu==1
        for l=1:indextabu
        div(l)=isequal(z0(:,:,j+1),ztabu(:,:,l));
        end
        else
                for  l=1:indextabu-1
                div(l)=isequal(z0(:,:,j+1),ztabu(:,:,l));
                if div(l)==1
                        break
                end
                end
        end
end
else
        ofgood(j)=ofgood(j-1);
        counter=counter+1;
        ztabu(:,:,indextabu)=z0(:,:,j);
        div=ones(1,indextabu);

        while any(div)
        if j==1
                z0(:,:,j+1)=replace_new(z0(:,:,j));
        elseif counter<threshold
                z0(:,:,j+1)=replace_new(z0(:,:,jbetter));
        else
                z0(:,:,j+1)=replace_new(z0(:,:,j-1));
        end

        for l=1:indextabu
                div(l)=isequal(z0(:,:,j+1),ztabu(:,:,l));
```

```
        if div(l)==1
                break
        end
end

end
indextabu=indextabu+1;
end

if counter > limit
        break
end

end

save('UB_workspace.mat')
```

A.2 Create function

```
function [z0]=replace_new(z0)

%replace_new is a Matlab function that
%randomly selects a column of a 0,1 matrix
%and in that column operates one
%switch between a 0 and a 1.

[r,c]=size(z0);
i=1;
while i==1
        c_rand = randi([1 c],1);
        col_z=z0(:,c_rand);
        pos_ones= find(col_z);
        pos_zeros= find(~col_z);
        i=isempty(pos_ones)|isempty(pos_zeros);
end

[r_ones,y]=size(pos_ones);
index_one = pos_ones( randi([1 r_ones],1) );

[r_zeros,u]=size(pos_zeros);
index_zero = pos_zeros( randi([1 r_zeros],1) );

clear y
clear u

z0(index_one,c_rand)= 0;
z0(index_zero,c_rand)= 1;

end
```

A.3 Cardinality initialization function

```
function [z0]=card_new(r,c,gamma)
%card_new is a matlab function that generates
%a matrix n*m, with on its
%columns ones and zeros so to respect the cardinality values.
z0=zeros(r,c);

for k=1:c
        for i=1:size(gamma(k))
                index=1:2:2*gamma(k)-1;
                z0(index,k)=1;
        end
end

end
```

Appendix B

Gantt charts: failure-patterns found by the Tabu Search algorithm

In this appendix we show the graphical representation of the results of the BAP test case outlined in chapter 7. The charts show the machines failures for each cardinality value that has been used. In each graph, the points represent which workpiece was affected by a failure over which machine. Notice that for the way we have constructed the test case, the failures over machines M1, M2, M4 are always 10, whereas the failures over M3 respect the cardinality constraint.

B.1 Upper bound

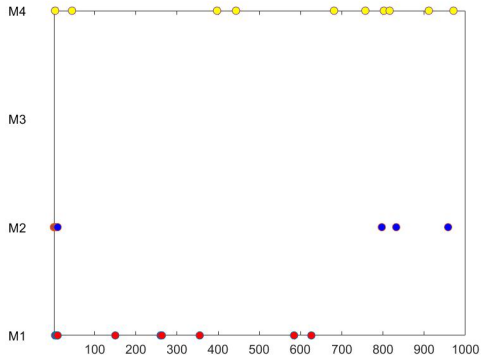


Figure B.1: Upper bound: failure-patter with $\Gamma_3 = 0$

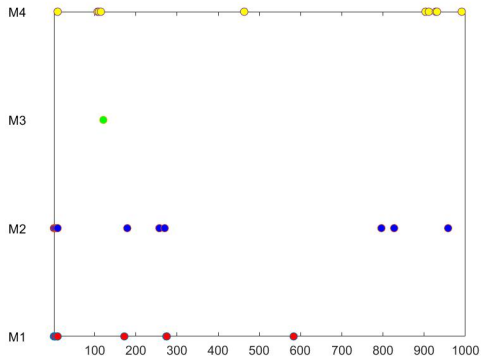


Figure B.2: Upper bound: failure-patter with $\Gamma_3 = 1$

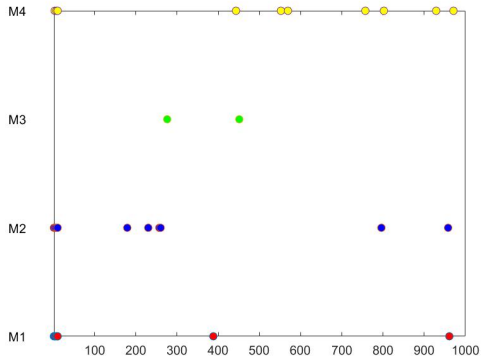


Figure B.3: Upper bound: failure-patter with $\Gamma_3 = 2$

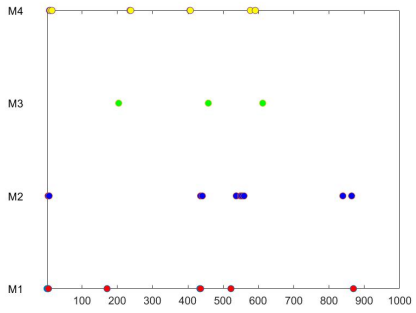


Figure B.4: Upper bound: failure-patter with $\Gamma_3 = 3$

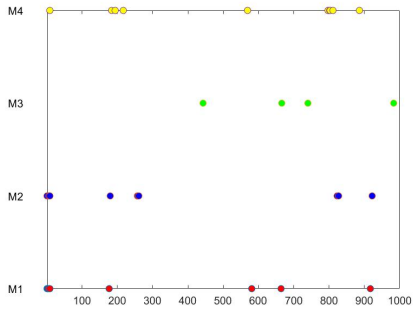


Figure B.5: Upper bound: failure-patter with $\Gamma_3 = 4$

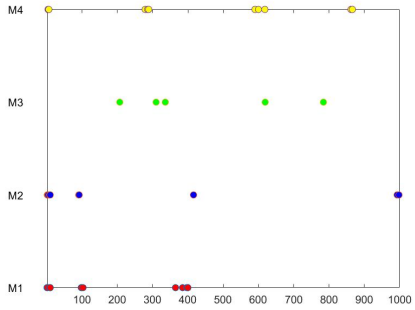


Figure B.6: Upper bound: failure-patter with $\Gamma_3 = 5$

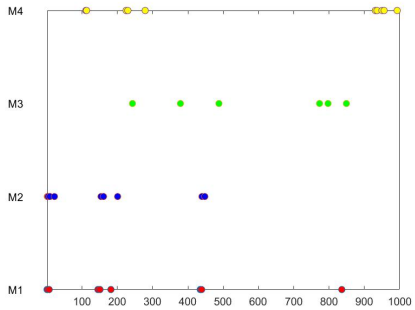


Figure B.7: Upper bound: failure-patter with $\Gamma_3 = 6$

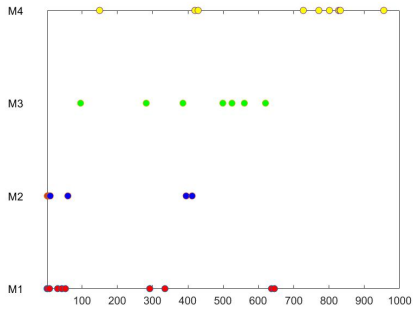


Figure B.8: Upper bound: failure-patter with $\Gamma_3 = 7$

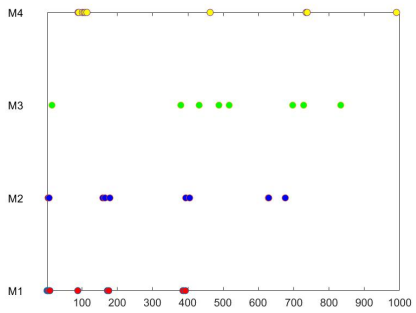


Figure B.9: Upper bound: failure-patter with $\Gamma_3 = 8$

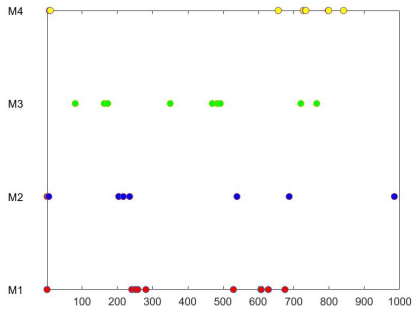


Figure B.10: Upper bound: failure-patter with $\Gamma_3 = 9$

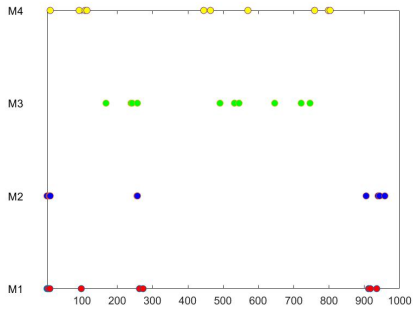


Figure B.11: Upper bound: failure-patter with $\Gamma_3 = 10$

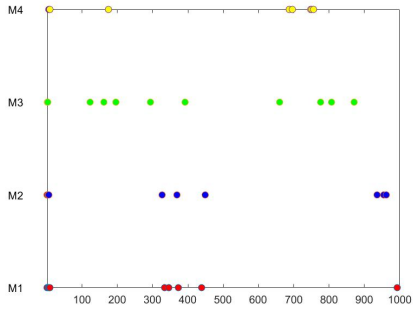


Figure B.12: Upper bound: failure-patter with $\Gamma_3 = 11$

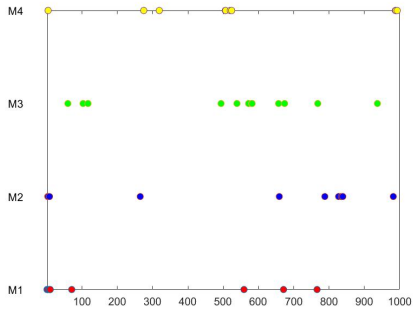


Figure B.13: Upper bound: failure-patter with $\Gamma_3 = 12$

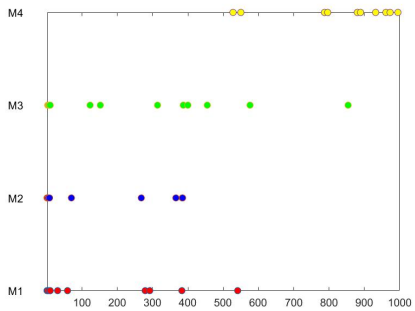


Figure B.14: Upper bound: failure-patter with $\Gamma_3 = 13$

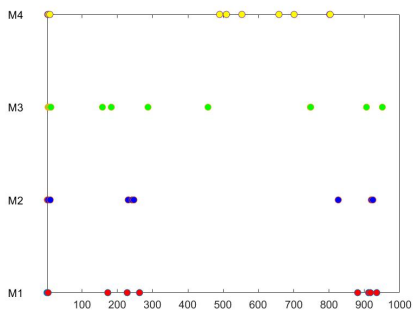


Figure B.15: Upper bound: failure-patter with $\Gamma_3 = 14$

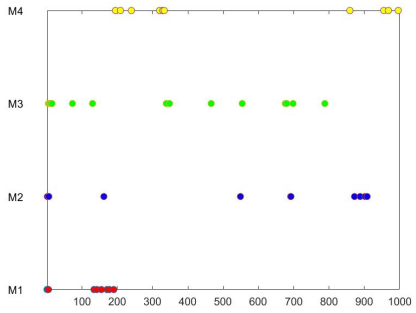


Figure B.16: Upper bound: failure-patter with $\Gamma_3 = 15$

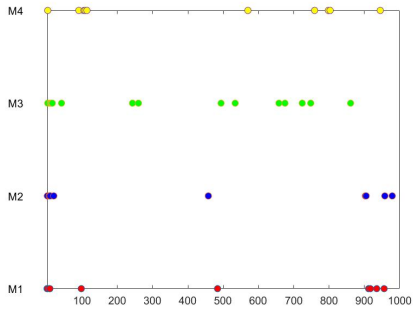


Figure B.17: Upper bound: failure-patter with $\Gamma_3 = 16$

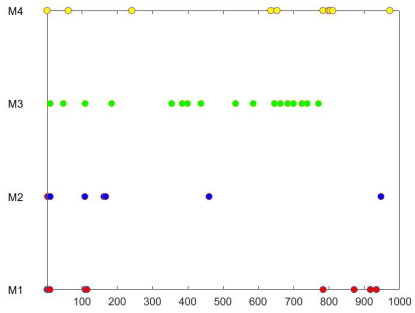


Figure B.18: Upper bound: failure-patter with $\Gamma_3 = 17$

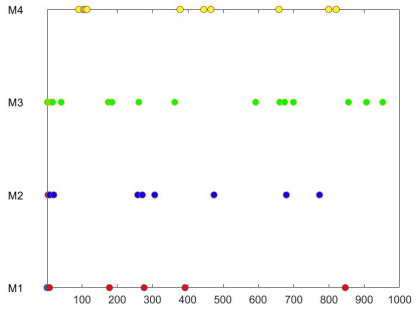


Figure B.19: Upper bound: failure-patter with $\Gamma_3 = 18$

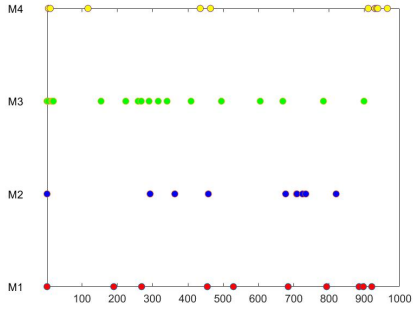


Figure B.20: Upper bound: failure-patter with $\Gamma_3 = 19$

B.2 Lower bound

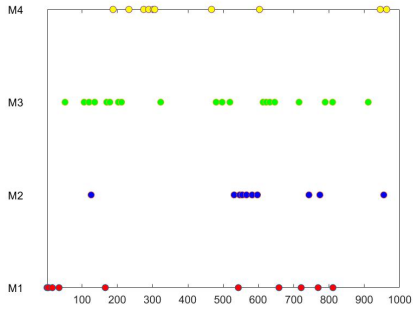


Figure B.21: Upper bound: failure-patter with $\Gamma_3 = 20$

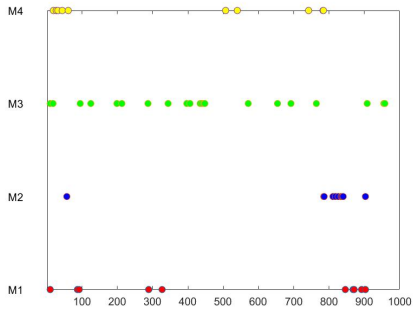


Figure B.22: Upper bound: failure-patter with $\Gamma_3 = 21$

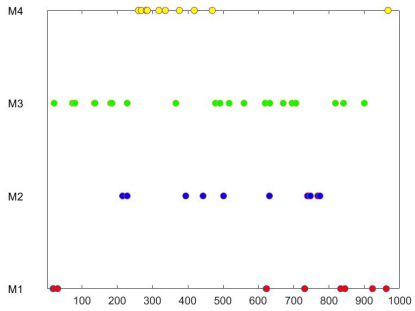


Figure B.23: Upper bound: failure-patter with $\Gamma_3 = 22$

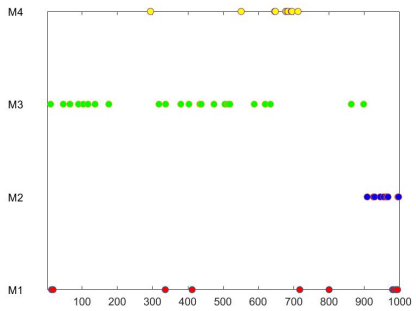


Figure B.24: Upper bound: failure-patter with $\Gamma_3 = 23$

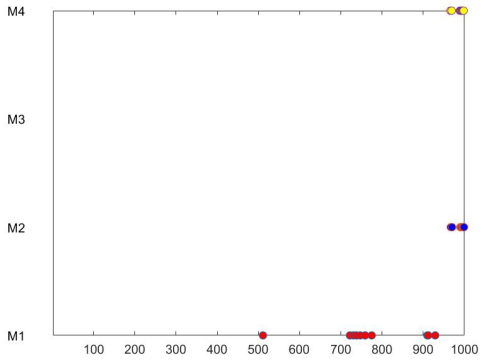


Figure B.25: Lower bound: failure-patter with $\Gamma_3 = 0$

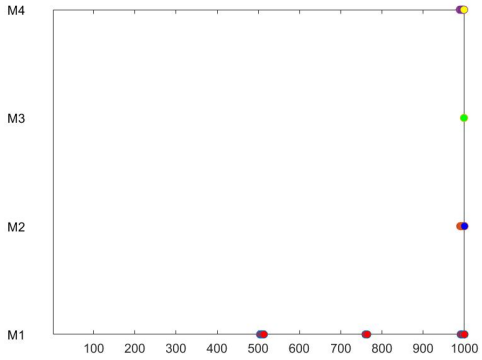


Figure B.26: Lower bound: failure-patter with $\Gamma_3 = 1$

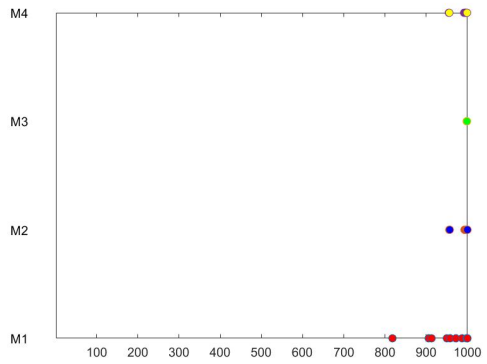


Figure B.27: Lower bound: failure-patter with $\Gamma_3 = 2$

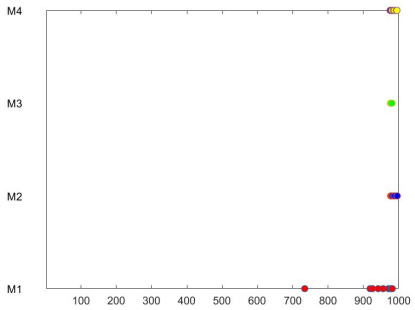


Figure B.28: Lower bound: failure-patter with $\Gamma_3 = 3$

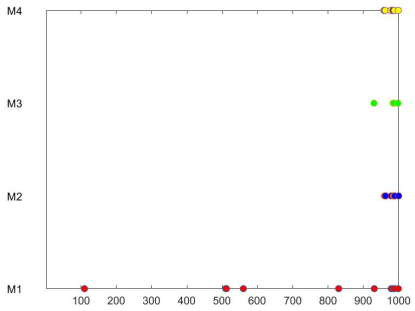


Figure B.29: Lower bound: failure-patter with $\Gamma_3 = 4$

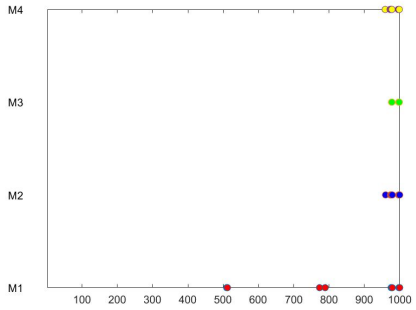


Figure B.30: Lower bound: failure-patter with $\Gamma_3 = 5$

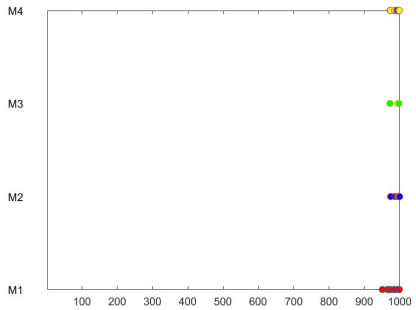


Figure B.31: Lower bound: failure-patter with $\Gamma_3 = 6$

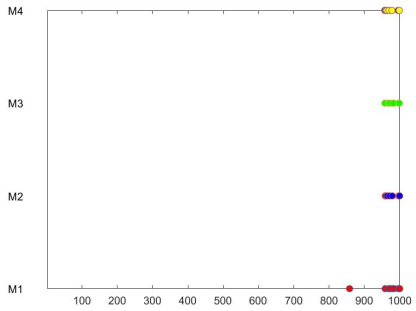


Figure B.32: Lower bound: failure-patter with $\Gamma_3 = 7$

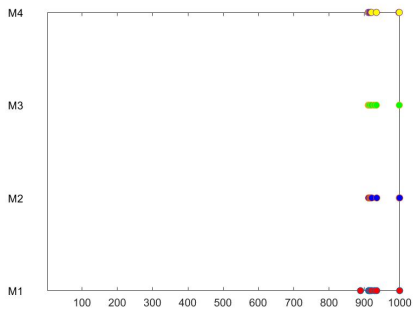


Figure B.33: Lower bound: failure-patter with $\Gamma_3 = 8$

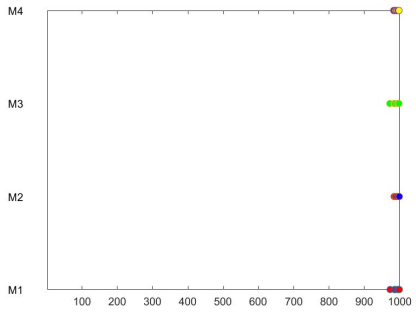


Figure B.34: Lower bound: failure-patter with $\Gamma_3 = 9$

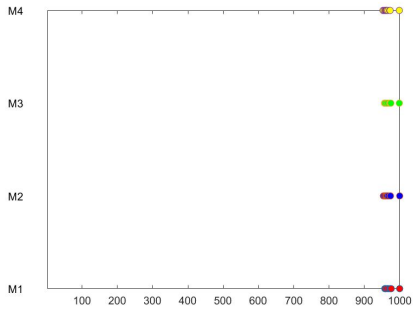


Figure B.35: Lower bound: failure-patter with $\Gamma_3 = 10$

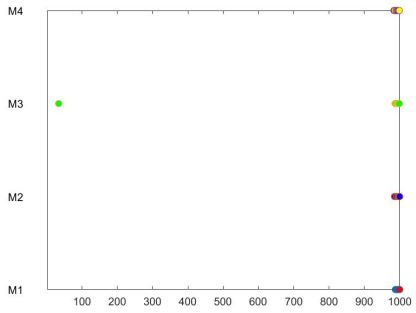


Figure B.36: Lower bound: failure-patter with $\Gamma_3 = 11$

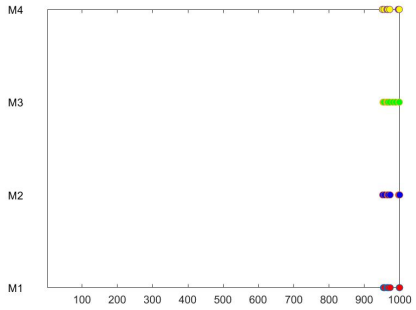


Figure B.37: Lower bound: failure-patter with $\Gamma_3 = 12$

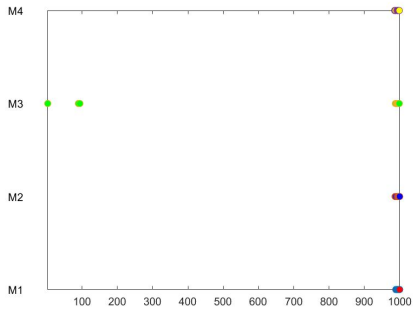


Figure B.38: Lower bound: failure-patter with $\Gamma_3 = 13$

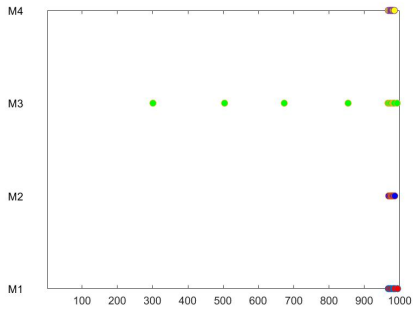


Figure B.39: Lower bound: failure-patter with $\Gamma_3 = 14$

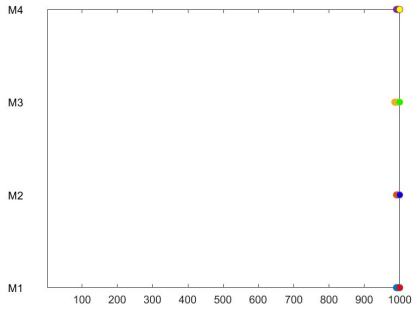


Figure B.40: Lower bound: failure-patter with $\Gamma_3 = 15$

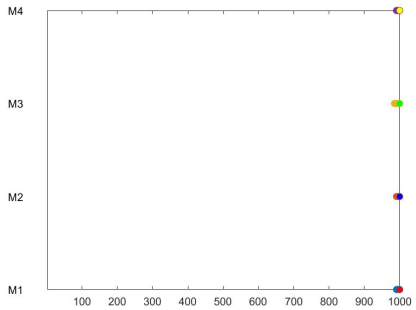


Figure B.41: Lower bound: failure-patter with $\Gamma_3 = 16$

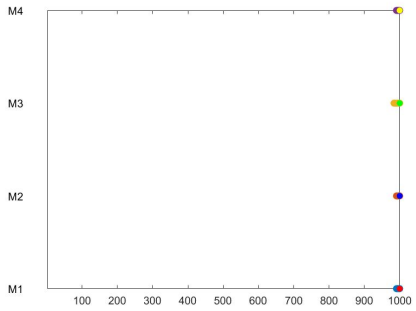


Figure B.42: Lower bound: failure-patter with $\Gamma_3 = 17$

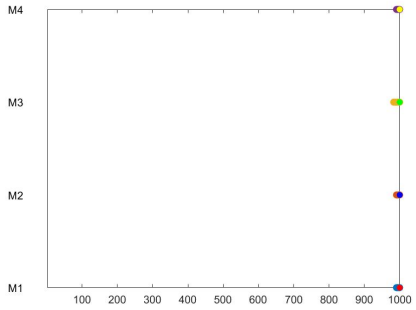


Figure B.43: Lower bound: failure-patter with $\Gamma_3 = 18$

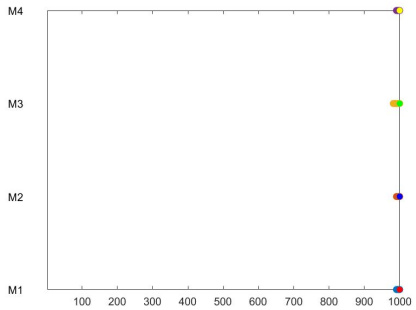


Figure B.44: Lower bound: failure-patter with $\Gamma_3 = 19$

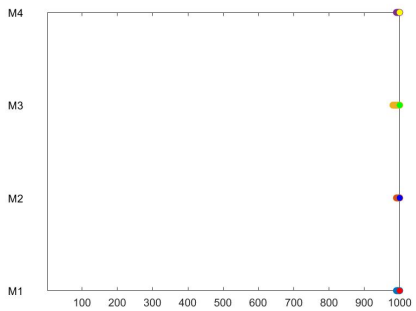


Figure B.45: Lower bound: failure-patter with $\Gamma_3 = 20$

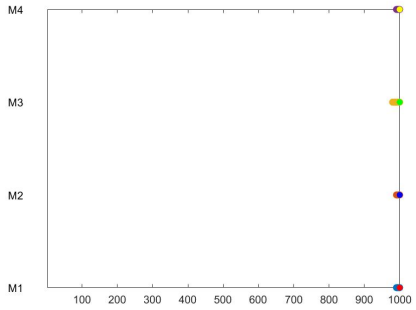


Figure B.46: Lower bound: failure-patter with $\Gamma_3 = 21$

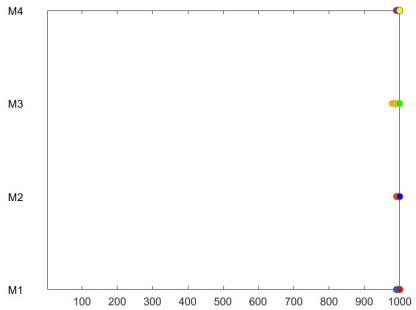


Figure B.47: Lower bound: failure-patter with $\Gamma_3 = 22$

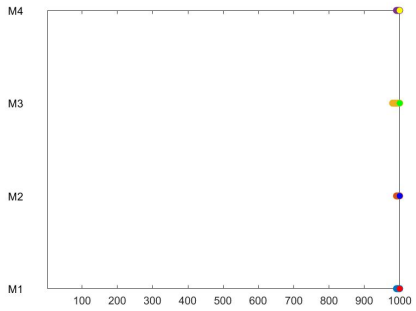


Figure B.48: Lower bound: failure-patter with $\Gamma_3 = 23$