M 2023

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

SCHEDULING OF A MULTI-PRODUCT BATCH PROCESS IN THE CHEMICAL INDUSTRY WITH DETERIORATING EFFECTS

DIANA GONÇALVES CAMPINHO
MASTER THESIS IN CHEMICAL ENGINEERING
PRESENTED TO THE FACULTY OF ENGINEERING
OF THE UNIVERSITY OF PORTO

Master in Chemical Engineering

Scheduling of a multi-product batch process in the chemical industry with deteriorating effects

Master dissertation

of

Diana Gonçalves Campinho

Developed within the course of dissertation

held in

LSRE-LCM – Laboratory of Separation and Reaction Engineering – Laboratory of Catalysis and Materials

Supervisor at FEP: **Professor Dalila Martins Fontes**

Supervisor at FEUP: **Professor Alexandre Ferreira**

July 2023

Acknowledgment

First and foremost, I am deeply grateful to my parents for their endless love, sacrifices, and unwavering belief in my abilities. Their constant support and guidance throughout my life have been invaluable, and I am eternally grateful to them.

To my dear boyfriend, André, thank you for being my pillar of strength and always believing in me, even in the most challenging times. Your patience, understanding, and unwavering support have been a constant source of motivation for me.

I would also like to extend my sincere gratitude to my close friends from college. Your friendship has been an essential source of encouragement, laughter, and support throughout this journey. It wouldn't be the same without you.

I am grateful to my supervisors, professor Alexandre and professor Dalila, for their expert guidance, knowledge, and continuous support throughout the process of creating this work. Their expertise, mentorship, and constructive feedback have significantly shaped the development of this thesis.

Finally, I would like to dedicate a special message to my late grandfather, Luís Campinho. Although he is no longer with us, his unwavering belief in my potential and his constant encouragement have been a guiding light throughout my life. This thesis is a tribute to your memory.

I extend my heartfelt gratitude to all those who have played a significant role in my academic journey. Your support, encouragement, and belief in my abilities have been immeasurable, and I am eternally grateful for your presence in my life.

Alexandre Ferreira, supervisor of this dissertation, is a member of LSRE-LCM - Laboratory of Separation and Reaction Engineering – Laboratory of Catalysis and Materials, financially supported by LA/P/0045/2020 (ALiCE), UIDB/50020/2020 and UIDP/50020/2020 (LSRE-LCM), funded by national funds through FCT/MCTES (PIDDAC).

Abstract

This work focuses on optimizing the scheduling of a multi-product batch process, considering deterioration effects. These effects compromise the processing time of several operations and can contribute to delays and waiting times, and larger total running time, among others.

The main goal is to study the effects of deterioration in a job-shop scheduling problem and raise awareness of this topic within the industry. The primary motivation for this study is the limited number of articles addressing deterioration in job-shop problems. Most existing research focuses on less complex configurations and the development of models for solving these problems.

The method proposed to solve this problem is Biased Random Key Genetic Algorithm. This algorithm has a problem-independent portion in which encoded solutions are sorted based on an objective function, and new solutions are generated through crossover and mutation. On the other hand, the problem-dependent portion decodes the solutions, converting them into feasible schedules. Additionally, a multi-population strategy is applied, in which multiple populations evolve in parallel and exchange high-quality chromosomes after a pre-defined number of generations. The algorithm proved to be efficient in solving traditional job-shop scheduling problems.

This work considers two types of deterioration: job and machine deterioration. Job deterioration relates to the processing time dependency on the waiting time between two operations of the same job, while machine deterioration relates to the dependency on machine running time. These effects are described by linear, exponential, and a sigmoid function. The sigmoid function is proposed to model the deterioration-restricted impact on processing time.

The results indicate that optimization considering deterioration improves the makespan in comparison to considering them only after optimizing. In the case of job deterioration, optimization manages to mitigate deterioration effects by 60-80 %. However, for machine deterioration, optimization mitigates about 10 % of the effects. Therefore, maintenance was introduced to study how it can be used to improve the makespan, in the latter case.

This study highlights the importance of addressing deterioration in industrial processes. Optimization techniques that account for deterioration can reduce costs, minimize disruptions, and enhance sustainability in industrial operations. In addition to improving the decoder for machine deterioration, future research can explore additional complex scenarios and improve the understanding and application of deterioration effects in real-life production environments.

Keywords: scheduling, job-shop; deterioration effects; BRKGA; chemical industry.

Resumo

Este trabalho centra-se na otimização do escalonamento de um processo multiproduto *batch*, considerando os efeitos de deterioração. Estes efeitos comprometem o tempo de processamento das várias operações e podem impor atrasos e tempos de espera, aumentar o tempo total de funcionamento, entre outros.

O principal objetivo é estudar os efeitos de deterioração num problema de escalonamento *job-shop* e sensibilizar a indústria para esta temática. A motivação reside na escassez de investigação na literatura sobre a aplicação de deterioração em problemas *job-shop*. A maioria dos estudos existentes foca-se em configurações menos complexas e no desenvolvimento de modelos para a resolução destes problemas.

O método proposto para resolver o problema é um *Biased Random Key Genetic Algorithm*. Este algoritmo tem uma parte independente do problema em que as soluções codificadas são ordenadas com base numa função objetivo, e são geradas novas soluções através de operações de cruzamento e mutação. Por outro lado, a parte dependente do problema descodifica as soluções, convertendo-as em escalonamentos viáveis. Além disso, é aplicada uma estratégia de multi-populações, na qual várias populações evoluem em paralelo e trocam cromossomas de elevada qualidade após um número pré-definido de gerações. O algoritmo provou ser eficiente na resolução de problemas de escalonamento *job-shop* tradicionais.

Este trabalho considera dois tipos de deterioração: deterioração do tarefa e da máquina. A deterioração da tarefa relaciona-se com a dependência do tempo de processamento em relação ao tempo de espera entre duas operações da mesma tarefa, enquanto a deterioração da máquina relaciona-se com a dependência em relação ao tempo de funcionamento da máquina. Estes efeitos são descritos por funções lineares e exponenciais, e é também proposta uma função sigmoide para modelar o impacto restrito da deterioração no tempo de processamento.

Os resultados indicam que a otimização considerando a deterioração melhora o tempo total de produção, em comparação com a consideração destes efeitos após a otimização. No caso da deterioração da tarefa, a otimização consegue mitigar os efeitos de deterioração entre 60 % a 80 %. Para a deterioração da máquina, a otimização apenas consegue mitigar 10 % dos efeitos. Assim, a manutenção foi introduzida no processo de otimização, de modo a estudar o seu impacto na melhoria do tempo total de operação.

Este estudo destaca a importância de abordar a deterioração nos processos industriais. A implementação de técnicas de otimização que tenham em consideração a deterioração pode contribuir para a redução de custos, minimizar interrupções e melhorar a sustentabilidade nas operações industriais. Além de melhorar o descodificador para a deterioração das máquinas, pesquisas futuras devem explorar cenários mais complexos e aprimorar a compreensão e aplicação dos efeitos da deterioração em ambientes de produção reais.

Declaration

I hereby declare, under word of honor, that this work is original and that all non-original contributions is indicated, and due reference is given to the author and source.

Diana Gonçalves Campinho

03/07/2023

Index

List of Figures

List of Tables

Notation and Glossary

a	real constant	
F	total number of stages	
GAP	percent optimality gap	%
I	makespan improvement percentage	%
J	total number of jobs	
k	actual generation	
k_m	number of exchanged solutions	
k_p	frequency for population exchange	
M	total number of machines	
N	number of genes in a chromosome	
n_1	total number of operations of job 1	
n_2	total number of operations of job 2	
O	total number of operations	
O_{ij}	operation i from job j	
p	population size	
$p_{[ij]}$	real processing time of operation i of job j	*
p_e	size of the elite population	
p_{ij}	reference processing time of operation i of job j	*
p_m	size of the mutant population	
S_{dl}	refers to the operation d from job l	
S_{ij}	refers to the operation i from job j	
t	CPU time	s
$t'_{(i-1)j}$	completion time of operation $i-1$ of job j	*
t_{ij}	starting time of operation i of job j	*

* arbitrary units of time

Greek Letters

α_{ij}	deterioration rate of operation i from job j	
ρ_e	elite allele inheritance probability	
α	deterioration rate	
π	number of parallel populations	
σ	percentage of makespan deviation	%

Indexes

d	refers to the job
i	refers to the operation
j	refers to the job
k	refers to the biased random key genetic algorithm current generation
l	refers to the operation
m	refers to the machine

List of Acronyms

BRKGA	Biased random-key genetic algorithm
BRKGA+CS	Biased random key genetic algorithm with clustering search
BRKGA+MP	Multi-population biased random key genetic algorithm
FJSP	Flexible job-shop scheduling problem
GA	Genetic algorithm
HGA+LS	Hybrid genetic algorithm with local search
JSP	Job-shop scheduling problem
NP-hard	Non-deterministic-polynomial-time hard
Opt.	Optimum solution
RKGA	Random key genetic algorithm
w/ maint.	With maintenance
w/ opt.	With optimization
w/out opt.	Without optimization

1 Introduction

1.1 Framing and presentation of the work

Batch processes play a crucial role in the chemical industry, enabling the production of a wide range of products. These processes involve scheduling multiple products through different stages, each with unique processing time and resource requirements.

To enhance productivity, reduce costs, and increase competitiveness, it is vital to optimize the total time required to complete a set of processes (makespan). By minimizing it, companies can boost production capacity, meet customer demands, and allocate resources efficiently. Thus, considering deterioration effects is particularly important when scheduling chemical processes as it can lead to a more realistic and accurate representation of the scheduling environment, significantly impacting overall efficiency and profitability. These effects result from factors such as equipment wear and tear, impurity accumulation, and variations in raw material quality directly affecting operations' processing time.

In conversations with several companies, it has become apparent that they are unaware of the impacts of deterioration effects, as they do not actively seek to identify the causes of failures during the production process. This lack of awareness results in production disruptions, lower productivity, and increased waste that, additionally, must be discarded. Consequently, considering deterioration in chemical processes schedule optimization is crucial for cost reduction, minimizing production disruptions, and enhancing social and environmental sustainability. Thus, it represents an immensely significant topic within the chemical industry.

Therefore, the main goal of this work is to study the effect of deterioration in a generic job-shop problem, which can represent a chemical process and create awareness for the topic within the industry.

The main motivation for this study is the limited number of articles addressing deterioration in job-shop scheduling problems. Most existing research has focused on applying deterioration effects in less complex configurations. Therefore, this thesis also aims to fill the gap in the literature by proposing a new approach to solve job-shop scheduling problems that incorporate deterioration effects.

Despite the existence of some articles considering the deterioration effects in the flexible job-shop scheduling problem (FJSP), there are not many articles that consider them in the job-shop scheduling problem (JSP). Although the FJSP is an extension of the JSP, where it is possible to choose which machine performs the operations and thus potentially achieve better results, studying the effects of deterioration in a JSP is more relevant. This is because the majority of

companies are small or medium-sized enterprises, which do not have significant investment capacity to acquire multiple machines.

1.2 Contribution of the author to the work

My contribution to the work involved implementing the Biased Random Key Genetic Algorithm developed by Gonçalves *et al.* (2010). I wrote and optimized the necessary code in Python, including the decoder, and implemented multiprocessing though the *joblib* library.

To conduct the experiments, I employed the Lawrence (1984) instances, which were obtained from http://jobshop.jjvh.nl/. Furthermore, to introduce deterioration into the instances, I designed a function that randomly selected deterioration rates and machines/operations with deterioration.

1.3 Organization of the dissertation

In addition to Chapter 1, which introduces the research topic, including the motivation for the study, this dissertation is organized into five additional chapters.

In Chapter 2 – Context and State of Art – the literature related to scheduling problems and job-shop scheduling problems is reviewed, emphasizing deterioration effects. The efficiency of genetic algorithms to solve this type of problem is also reviewed, with a specific focus on Biased Random Key Genetic Algorithms.

In Chapter 3 – Methods and Methodology – the methodology used to conduct the study is described, including the Biased Random Key Genetic Algorithm solution approach. The detailed description is provided, including the assumptions made as well as the specific deterioration functions used.

In Chapter 4 – Results and Discussion – the methodology efficiency is proved, comparing the computational results with those of other studies in the literature. The computational results for deterioration are presented, analysed, and their implications are discussed.

In Chapter 5 – Conclusion – the main achievements of the research are summarized, as well as the limitations of the study and potential lines for future research.

Finally, in Chapter 6 – Assessment of the work done – a small assessment of the present work is performed.

2 Context and State of the art

In this chapter, the literature and state-of-the-art in the field of scheduling problems will be reviewed, focusing on job-shop problems with deterioration effects, and how genetic algorithms can help to solve these problems.

2.1 Batch process plants

Batch processes can handle variations in feedstock and product specifications due to their dynamic nature. This flexibility is essential for multiproduct or multipurpose facilities, making them well-suited for producing low-volume, high-value products (Mokeddem *et al.*, 2009).

In batch processing, products that require repeated process steps may require less production equipment compared to continuous operations. This is because, in continuous processing, each process step typically requires its own set of equipment. In contrast, in batch processing, the same equipment can be used or employed in multiple processes (Mokeddem *et al.*, 2009).

Batch chemical processing is commonly used and presents certain benefits over continuous operations in specific situations. These include manufacturing processes requiring flexibility in the product rate or mix, limited capacity, chemistries incompatible with continuous operations, or cases in which lot integrity must be held (Korovessi *et al.*, 2006). Industries in which batch operations are common include pharmaceuticals, specialty chemicals (adhesives, catalysts, coatings and paints, industrial gases, plastic additives, water management chemicals, and lubricants), and food.

Thus, batch processes play a crucial role in chemical manufacturing systems, producing large volumes of chemical products that meet the demands of everyday life. With increasing globalisation, there is rising global competition in the traditional chemical process industry. To remain competitive in the global marketplace, companies must optimize their production management and establish responsive systems to deal with market fluctuation.

2.2 Scheduling problems

Scheduling is the core of production management in a chemical process. It is a decision-making process that deals with the allocation of resources to tasks over given periods to achieve a specific objective, such as maximisation of profit or minimisation of makespan (Pinedo, 2012). Production scheduling, which typically covers a period of a week to a month, provides a short-term outlook on the requirements for each product to be manufactured, with the time scale tailored to fit manufacturing needs (Korovessi *et al.*, 2006).

When considering the machine scheduling problem, which originally emerged in manufacturing systems, a job consists of one or more activities, and a machine represents a resource capable of performing at least one activity at a time (Chen *et al.*, 1998). The total number of jobs is denoted by J and the total number of machines by M. Generally, the subscript j is used to denote the job, while the subscript i is assigned to the operation (Pinedo, 2012). Machines' different configurations are possible.

In a production system, a job can require one or multiple operations, depending on whether the system is single or multi-stage. Single-stage systems involve one or M machines operating in parallel. The simplest case is a single machine, which is a special case of more complex machine environments (Pinedo, 2012; Brucker *et al.*, 2012).

When considering multi-stage systems (shop scheduling problems), there are three primary types to consider. These systems consist of F stages, each with a distinct function. In a flow-shop with F stages, every job follows a predetermined sequence, processing through stages 1 to F in a fixed order (Chen *et al.*, 1998; Brucker *et al.*, 2012). In an open shop, each job also goes through every stage once; however, the routing, *i.e.*, the sequence of steps, can vary between jobs or is part of the decision-making process (Chen *et al.*, 1998). In a job-shop, each job has a specified routing through the stages, incorporating specific precedence constraints, and the routing may differ from one job to another (Chen *et al.*, 1998; Brucker *et al.*, 2012). Furthermore, there are multiprocessor variations of multi-stage systems where each stage has several parallel machines, resulting in flexible job-shops and flexible flow-shops (Pinedo, 2012)

In addition, machines may have the capability to process multiple jobs simultaneously. The reason for grouping jobs into batches is to increase efficiency, as it may be more cost or time-efficient to process jobs in a batch rather than individually. The literature identifies two types of batching machines: serial and parallel (Pei *et al.*, 2022). A serial batching machine has a batch length that equals the sum of the processing times of its job, while a parallel batching machine has a batch length that equals the largest processing time of its jobs.

Analogous to the JSP, a multi-product batch process in the chemical industry requires the production of different products (jobs), each with its own recipe and processing requirements, on a shared set of processing equipment (machines). Thus, adopting the job-shop perspective, the multi-product batch process can be effectively modelled as a scheduling problem.

2.3 Job-Shop Scheduling Problem

The JSP is an optimization problem in which multiple jobs with different routings are allocated to machines at a given time while trying to optimize an objective function (Dimopoulos *et al.*, 2000). It consists of ordering the operations of the J jobs to be processed in M machines, in which each job involves various machining operations.

Due to its great application value, the JSP has always been a concern and has been studied widely, with applications in semiconductors (Jiang *et al.*, 2023), machine manufacturing (Li *et al.*, 2020), automobile manufacturing, chemicals and pharmaceuticals (Azzouz *et al.*, 2017), and food (Bouazza *et al.*, 2017).

The JSP is one of the most complicated and important problems, recognized as non-deterministic-polynomial-time hard (NP-hard) - the solution time is an exponential function of its problem size - and a very challenging combinatorial optimization problem since the 1950s (Çaliş *et al.*, 2015). The problem complexity arises from the fact that there are many possible ways to schedule the jobs on the machines, and finding the optimal schedule requires searching through a vast space of possible schedules (Çaliş *et al.*, 2015). However, the JSP is an important practical problem in many manufacturing contexts, as optimizing its solution can significantly improve production efficiency and reduce costs.

According to Xiong *et al.* (2022), in the classic model of this type of problem, a set of assumptions is typically considered: (1) all jobs are released at time zero, cannot be cancelled during processing, and the number of jobs is predetermined and fixed; (2) all machines are available at time zero; (3) all jobs have equal weight and no specify due dates; (4) there is no arbitrary priority among jobs; (5) each operation has a predetermined processing time; (6) each job follows a fixed processing routing; (7) jobs are independent from each other; (8) each machine can handle only one operation at a time; (9) waiting between adjacent operations of the same job is allowed; (10) machines operate independently; (11) operations cannot be interrupted once stared; (12) pre-emption is not permitted; (13) setup times and transit times are negligible; (14) machine breakdowns and maintenance are not taken into account; (15) re-processing and emergency job insertion are not considered; (16) operator constraints are not considered; (17) buffer, warehouse and container capacities are assumed to be unlimited; (18) auxiliary tools and appliances are always available.

The relaxation of any of these assumptions can result in variations of the problem, which can be classified into different groups, as Xiong *et al.* (2022) explained. For instance, relaxing assumption 13 means that machine availability (Tamssaouet *et al.*, 2018; Lin *et al.*, 2021) and/or job transport time (JSP with transportation) are considered (Fontes *et al.*, 2022; Fontes *et al.*, 2023b), and dismissing assumption 6 allows for considering alternative routings or machines (Moon *et al.*, 2008; Chaudhry, 2012). If machines are in different locations, transport times and costs are taken into account (Homayouni *et al.*, 2023), leading to distributed JSP (Luo *et al.*, 2022). This problem deals with the assignment of jobs to factories geographically distributed and with determining a good operation schedule of each factory, resulting in a flexible problem (FJSP). Relaxing the 16th assumption explicitly involves the operator in the processing resource (Mencía *et al.*, 2015), meaning that both machine and operator must be

available for the operation to be processed (dual-resource constraints). Besides, there is a class of JSPs that does not rely on relaxing the previous assumptions. Researchers have recently started incorporating energy conservation and environmental protection criteria into scheduling decisions, leading to energy-efficient JSP (Fontes *et al.*, 2023c; Abedi *et al.*, 2020).

The processing time can also be nondeterministic or nonconstant (relaxation of fifth assumption). This problem can be classified as deterioration effects (Abedi *et al.*, 2020), controllable processing times (Gao *et al.*, 2022), fuzzy processing times (Bustos-Tellez *et al.*, 2018), or random distribution processing times (Zhang *et al.*, 2011). Figure 2.1 summarizes the scheduling problems and JSP types of problems.

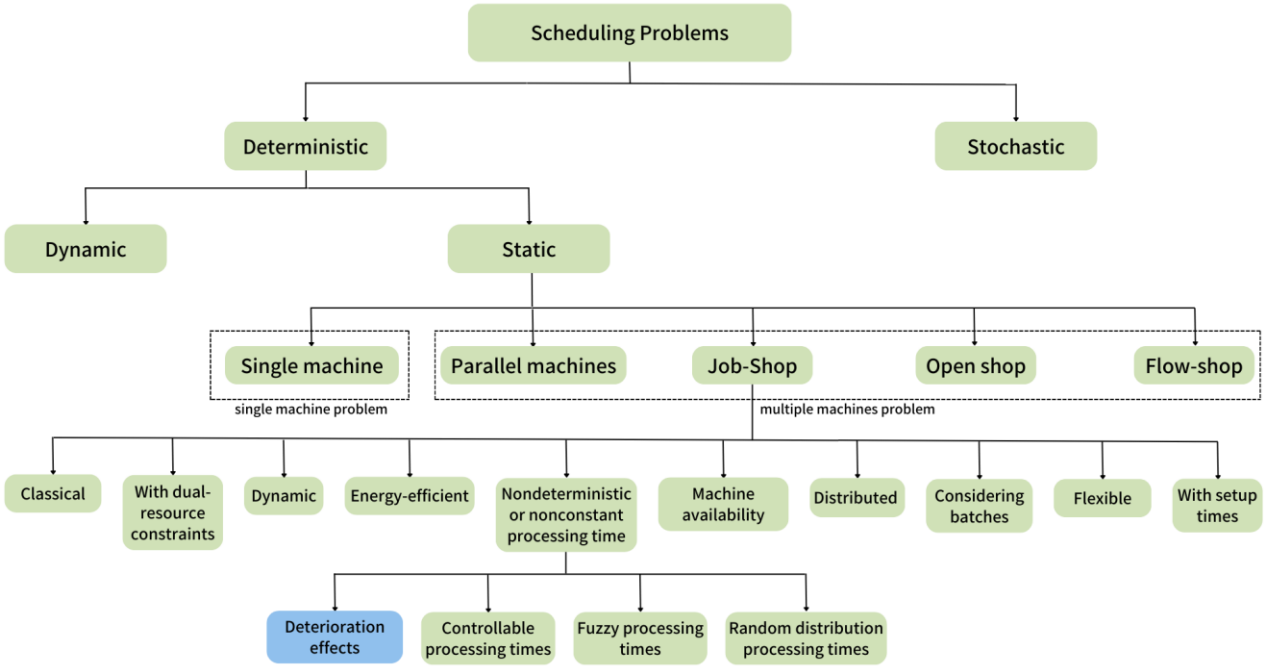

Figure 2.1. Scheduling problems classification.

In the basic model of scheduling problems, the processing time of a job is assumed to be constant. However, processing times change due to several conditions. The case in which the processing time increases over waiting, start, or release time concerns the deterioration problem (Renna, 2015). This is the JSP addressed in this work.

2.4 Deterioration effects

One of the main problems most industrial sectors face is the uncertainty of its parameters, such as processing times. In addition, raw materials waiting in a queue for processing may deteriorate over time. In some cases, the deterioration results in a complete loss of the raw materials, while in others, they are salvageable, however requiring additional processing time. This phenomenon, introduced by Gupta *et al.* (1988) and Browne *et al.* (1990), is known as the "deterioration effect."

This effect is observed in a wide range of situations, ranging from production, *e.g.*, steel rolling mills, to rescue, *e.g.*, fire-fighting response scheduling (Kunnathur *et al.*, 1990; Gupta *et al.*, 1988; Browne *et al.*, 1990). For instance, the time and effort required to control a fire increases if the containment process is delayed. Hence, the processing time depends on the starting time of the job. Similarly, in steel rolling mills, the processing time of an ingot waiting in a queue between two processes depends on the waiting time, as must be reheated if its temperature falls below a critical value.

Web of Science™ database was selected as the research resource to identify studies related to the research topic. Initially, it was necessary to determine the most suitable keywords commonly employed in the literature to approach this theme. The search was subsequently restricted to articles incorporating the following combinations of generic keywords within their title, abstract, or list of keywords: ("scheduling") AND ("deteriorati* job*" OR "deteriorati* effect*" OR "job* deteriorati*"). In total 542 documents were collected until 2022. After filtering by articles written in English until 2022, a set of 470 research articles remained. The findings suggest that research on this type of problem has generated much interest in recent years. Figure 2.2 shows the number of documents published each year, from 1990 to 2022.

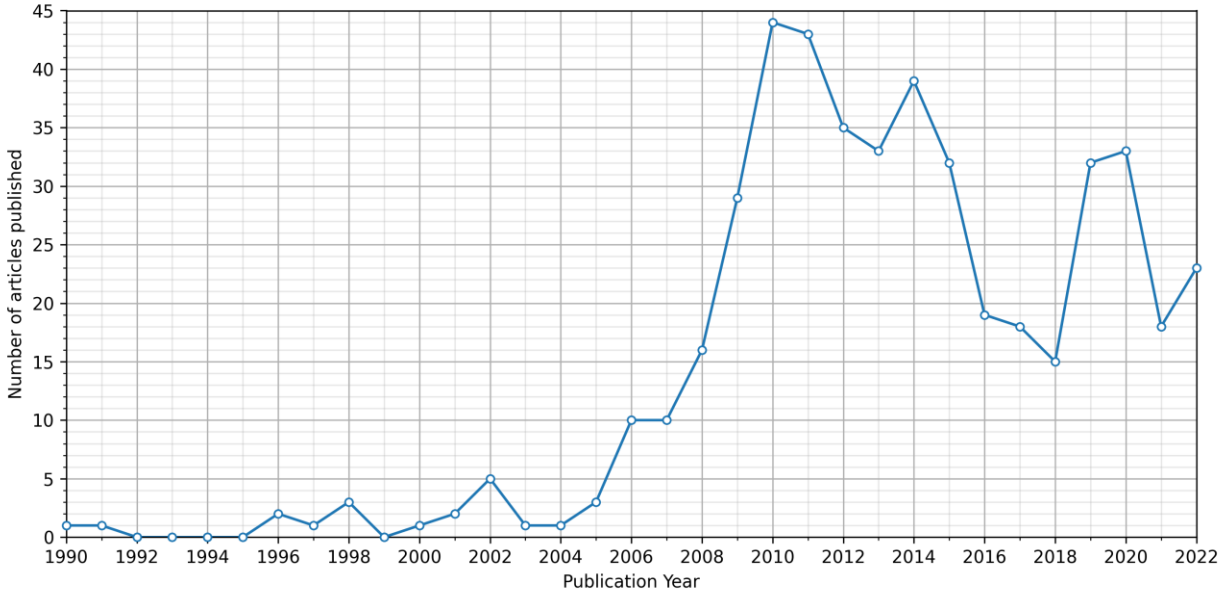

Figure 2.2. Number of articles concerning scheduling with deterioration until 2022.

Numerous configurations have been investigated, with the single-machine being the most extensively studied. Nevertheless, as elucidated by Pinedo (2012), scheduling problems in complex configurations can be decomposed into sub-problems involving single machines. However, more recently, researchers have increasingly searched for more complex configurations (Shabtay *et al.*, 2022), which are more common and realistic in the

manufacturing industry. Nevertheless, research on scheduling problems in more complex settings is still at an early stage.

Refining the search with generic keywords ("job-shop" OR "jobshop" OR "job shop"), 13 articles were presented. Among the articles found, 5 of them considered deterioration in flexible job-shop problems, and 5 others did not address JSP. In addition, 10 out of the 13 papers presented were published between 2019 and 2022, indicating the recent interest of the literature in this type of problem. It follows a brief description of the most relevant works found.

Wu *et al*. (2020) focused on a multi-objective FJSP considering deterioration effects. The authors consider machine deterioration and model it as a piecewise linear function. The deterioration rate of each machine was randomly generated using the uniform distribution $U[0.005, 0.150]$. In a previously article, Wu *et al*. (2019) argued that the deterioration effect has a limited impact on processing time in real-world production. Since, the processing time cannot be increased indefinitely. Therefore, describing the real processing time using a limited function would be more rational. In both articles, the authors concluded that consider the deterioration effects in the optimization model reduces the impact of these effects on the practical production.

Azzouz *et al*. (2020) proposed a formulation for the FJSP with sequence-dependent setup and deterioration effects. The authors modelled the deterioration effect by resorting to a linear function. However, they do not provide any information regarding the deterioration rate.

Jiang *et al*. (2022) designed a modified animal migration optimization algorithm to solve an energy-conscious FJSP with transportation time and deterioration effect simultaneously. To describe the deterioration effects, the authors proposed a linear function. To analyse the deterioration effect, the authors considered three scenarios with different levels of deterioration rates, which were randomly generated following three uniform distributions $U[0.005, 0.150]$, $U[0.150, 0.250]$ and $U[0.250, 0.400]$.

Tao *et al*. (2021) proposed a disruption management method based on cumulative prospect theory for the urgent with deteriorating effect arrival in FJSP, and an optimized multi-phase quantum particle swarm algorithm was addressed to select the processing route. A linear function describes the actual processing time for each job with deterioration. The authors did not specify the deterioration rate used.

On the other hand, Abedi *et al*. (2020) studied a multi-objective energy-efficient JSP involving speed scaling machines affected by cumulative deterioration, in which maintenance activities are needed. However, the processing time of operations depended on the jobs' sequence, not the time.

Likewise, Renna (2015) studied rate-modifying activities in a job-shop manufacturing system. The study considers that the generic workstation experiences a maximum level of deterioration, followed by the requirement for maintenance activity. The author proposed an exponential function to describe the deterioration of each machine. The deterioration rate was assumed to be 0.1.

Mosheiov (2002) addressed JSP with deteriorating jobs to minimise the makespan, assuming a simple linear deterioration function. Although, it is stated that the deterioration rate used is job and machine-dependent, no values are provided. The work mainly focused on analysing the complexity of shop problems.

In a recent comprehensive review on deteriorating and learning effects, Pei *et al.* (2022), reports on works addressing the single and parallel machine and flow shop scheduling problems, but none on the JSP.

Numerous studies have confirmed that the problem of deterioration is classified as NP-hard (Renna, 2015; Bachman *et al.*, 2002; Mosheiov, 2002). Since the classical JSP is also NP-hard, it follows that the JSP with deterioration effects is likewise NP-hard. Given that metaheuristics have demonstrated their effectiveness solving NP-hard combinatorial problems (Çaliş *et al.*, 2015), this work proposes a metaheuristic approach.

2.5 Metaheuristics

Combinatorial optimization models can represent various practical challenges such as routing, scheduling, inventory control, production, and location. To obtain satisfactory solutions quickly, heuristics are frequently used in production methods.

While simple heuristics are sufficient for solving simple problems, more complex problems require more elaborate solution approaches. As a result, many heuristics and metaheuristics have been developed over the years (García-Martínez *et al.*, 2018).

Metaheuristics are problem-independent and can incorporate or be combine with simple heuristics. By exploring the problem's search space, they can offer solutions of higher quality compared to those obtained through simpler optimization algorithms. They are specifically designed to find satisfactory solutions for complex problems, as is often the case in combinatorial optimization, with greater efficiency (García-Martínez *et al.*, 2018).

There are two main classes of metaheuristics: single-solution and population-based metaheuristics (Boussaïd *et al.*, 2013). Single-solutions metaheuristics start with a single initial solution and try to improve on it, typically, by resulting to some sort of local search, while population-based metaheuristics start with an initial population of solutions and generate new populations iteratively. Examples of single-solution metaheuristics include simulated

annealing, tabu search, and greedy randomized adaptive search procedure, while genetic algorithm (GA), ant colony optimization, and particle swarm optimization are examples of population-based metaheuristics (Boussaïd *et al.*, 2013).

2.5.1 Genetic Algorithms

The main characteristic of a GA is its ability to operate directly on the problem structure without being restricted by derivative or function continuity limitations. It also has a global search capability and can adjust search directions and self-adaptation. This algorithm has proven to be one of the most effective evolutionary techniques for solving JSP and deterioration problems (Çaliş *et al.*, 2015).

GAs can be compared to biological evolution, where the genetic code changes in each generation and evolves along generations. Additionally, the idea of *survival of the fittest* is employed in this type of algorithms, to identify an optimal or near-optimal solution (García-Martínez *et al.*, 2018).

In a GA, an analogy is made between a solution and an *individual* in a *population*. Each individual is represented by a *chromosome* that encodes the solution. A chromosome consists of a gene sequence, each representing an *allele*. Moreover, each chromosome is associated with a *fitness* value, which is correlated to the objective function value of the solution it encodes (Gonçalves *et al.*, 2010).

A GA creates a group of individuals, known as a population, which evolves over multiple generations. In each generation, a new population is formed by combining elements from the existing population. Moreover, random mutations are introduced to avoid local minima. The algorithm follows the principle of survival of the fittest when selecting individuals for mating and producing offspring. While individuals are chosen randomly, those with higher fitness are prioritized over less suitable individuals (Gonçalves *et al.*, 2010; García-Martínez *et al.*, 2018).

2.5.2 Biased Random-Key Genetic Algorithm

The random-key genetic algorithm (RKGA) was introduced by Bean (1994) to address scheduling and optimization classic problems. The chromosomes in RKGA are represented as a vector of randomly generated real numbers in the interval $[0,1[$. A deterministic algorithm, *decoder*, is used to associate these vectors of random numbers with a solution to the problem (Gonçalves *et al.*, 2010).

Biased-random key genetic algorithm (BRKGA) is a variant of RKGA introduced by Gonçalves *et al.* (2010), in which parents are chosen by combining an element randomly selected from the elite partition (individuals with the best fitness values) with one selected from the non-elite partition.

In a BRKGA, the mutation occurs with a predetermined probability defined in the framework, similar to what happens in a RKGA. BRKGA additionally enables to define of the offspring's probability to inherit specific characteristics from the elite parent by employing *parametrized uniform crossover* (Gonçalves *et al.*, 2010).

Through the evolutionary dynamics of the GA, the system acquires knowledge about the correlation between random key vectors and solutions with good fitness values (Oliveira *et al.*, 2010). Another advantage of the random key representation is the ability to employ conventional genetic operators. This characteristic allows the adoption of the GA for other optimization problems, requiring only minor adjustments to routines associated with the specific problem (Oliveira *et al.*, 2010).

BRKGA has been used with success on real-world problems, such as packing (Gonçalves *et al.*, 2010), scheduling (Gonçalves *et al.*, 2014; Homayouni *et al.*, 2023; Fontes *et al.*, 2023c; Homayouni *et al.*, 2022), and vehicle routing (Lopes *et al.*, 2016), among others (Gonçalves *et al.*, 2010). There are very few articles in the literature concerning BRKGA to solve deterioration problems.

Cunha *et al.* (2018) proposed a BRKGA to minimise the total weighted completion time of the incidents to be attended, considering fuzzy processing times. The results obtained showed that the solutions were 2.17 % better than those obtained with constructive heuristics. In their work, elite population accounted for 27 % of the total population, while the mutant population represented 17 %. Moreover, the probability of inheriting an allele from the elite parent was found to be 70 %. The population size was 1000, and the stopping criterion was 1000 generations.

Some authors combined BRKGA with other algorithms. Ma *et al.* (2018) investigated the problem of scheduling step-deteriorating jobs with release times on a single parallel-batching machine. They proposed a hybrid metaheuristic algorithm, combining BRKGA and variable neighbourhood search. Kong *et al.* (2020a) introduced a parallel-batching scheduling problem on parallel machines, where the actual processing time of a job is subject to the phenomena of deterioration and learning. The researchers proposed a hybrid algorithm, combining BRKGA and differential evolution, which proved to be efficient. Later, the same authors addressed the deterioration in the hot rolling mill using BRKGA (Kong *et al.*, 2020b).

None of the articles found concerned the job-shop problem with deterioration. However, there are some articles concerning job-shop configuration without deterioration. Gonçalves *et al.* (2014) proposed BRKGA to solve JSP, using a decoder with three phases. The initial phase produced an active schedule, the second phase attempted to improve it with a local search, and, in the last phase, the chromosome was adjusted to reflect the solution found by the previous phases. The results obtained were better by combining the BRKGA with the three

phases, with the second phase having the most significant contribution. The researchers chose 10 individuals from the total population to include the elite population, as well as the mutant population. Furthermore, they established a probability of 85 % for inheriting the allele from the elite parent. The fitness function was the modified makespan (combining the makespan with a measure of the potential for improvement of the schedule), and the stopping criterion was 20 generations.

Fontes *et al.* (2023c) addressed the energy-efficient JSP with transport resources by considering speed adjustable machines (where jobs are processed) and vehicles (that transport the jobs around the shop-floor). The goal was to find solutions balancing makespan and total energy consumption. The computational experiments have shown it to be effective and efficient. The authors defined the elite population as 20 % of the total population, and the mutant population as 10 % of the total population and established a probability of 70 % for inheriting the allele from an elite parent.

Besides that, Won *et al.* (2018) designed a genetic algorithm based on BRKGA for the JSP with cutting and parallel operations. The results showed that this algorithm could find an optimal solution for a small-size problems. The authors used a population of 30 individuals and a limit of 500 generations.

Finally, Rizzi *et al.* (2015) proposed a hybrid metaheuristic combining BRKGA and clustering search to solve JSP instances. The authors defined the population as 1000 individuals, with the elite population accounting for 10 % and the mutant population for 20 %. The probability of inheriting the allele from the elite parent was set at 70 %, and each instance was executed 20 times. They used as a stop criterion 400 generations or finding the optimal solution.

3 Methods and Methodology

3.1 Biased random key genetic algorithm

BRKGA heuristics are based on a general-purpose metaheuristic framework. This framework has two parts, one problem-dependent and another problem-independent, as can be seen in Figure 3.1 (Gonçalves *et al.*, 2010). The latter has no information about the problem being solved. The only connection to the combinatorial optimization problem under study is the problem-dependent part of the algorithm. In it, a decoder is employed to generate feasible solutions from the vectors of random-keys and to calculate the quality of the solutions generated, *i.e.*, the solutions' fitness (Gonçalves *et al.*, 2010). Hence, to define a BRKGA heuristic, one must focus on the abstract solution representation (encoding) and the construction of the solutions from a given abstract representation (decoding).

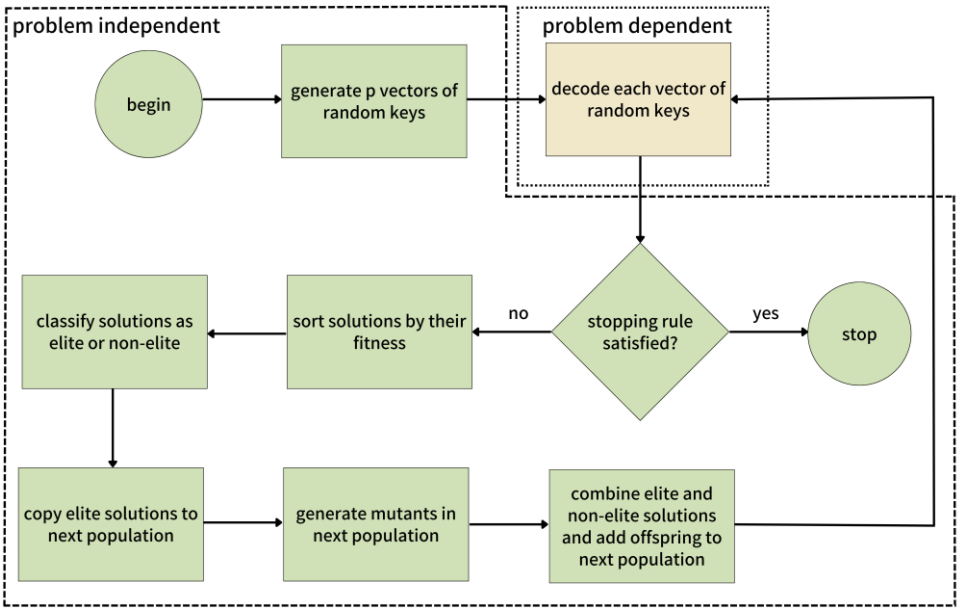

Figure 3.1. Flowchart of a BRKGA. Adapted from Gonçalves et al. (2010).

The next subsections explain in detail the problem-independent and dependent parts.

3.1.1 Problem-independent part

The problem-independent part of a BRKGA involves generating and evolving a set of random keys.

Initial population

The initial population has p vectors of random-keys, with each allele being independently generated at random within the real interval $[0, 1[$ (Gonçalves *et al.*, 2010). After the decoder calculates the fitness of each individual, the population is divided into two distinct groups: a

group of p_e *elite* individuals, consisting of the individuals with the best fitness values, and the remaining set of $p - p_e$ *non-elite* individuals, in which p_e is smaller than $p - p_e$ (Gonçalves *et al.*, 2010, 2018a).

New generation

To evolve the population, a new generation of individuals must be produced. The new generation is obtained by combining of the elite individuals of the current generation, the offspring generated by reproduction (crossover), and the mutants. The general process is represented in Figure 3.2.

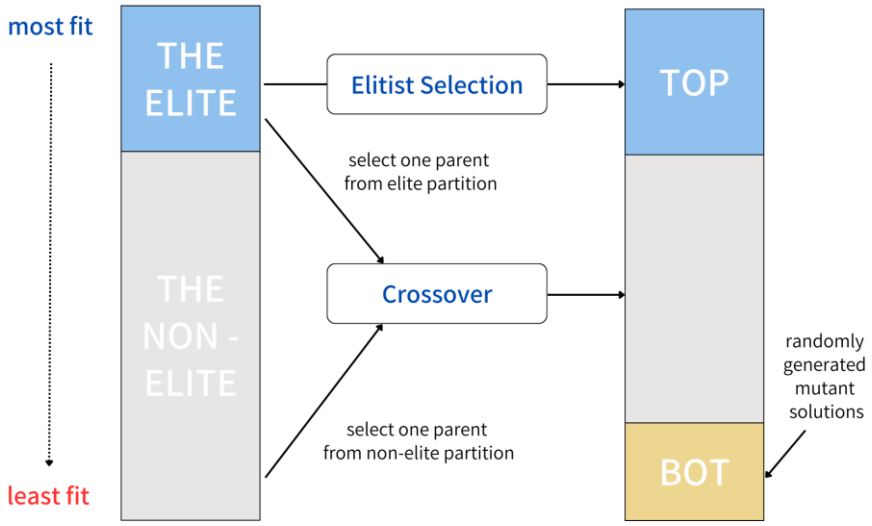

Figure 3.2. Transition from generation k to generation $k + 1$ in a BRKGA. Adapted from Gonçalves et al. (2010).

The algorithm employs an *elitist strategy* by directly transferring all the elite individuals from generation k to generation $k + 1$, without any modifications. This approach effectively preserves the high-quality solutions found during the iterations of the algorithm, leading to a consistently improving heuristic (Gonçalves *et al.*, 2010).

This algorithm incorporates mutation by introducing *mutants* into the population. A mutant is a vector of random keys generated with the method employed to generate the elements of the initial population. During each generation, a small number p_m of mutants is introduced into the population.

To create the next generation, after incorporating the p_e elite individuals and the p_m mutants, an additional $p - p_e - p_m$ individuals must be generated to complete the population of generation $k + 1$. This is achieved by producing $p - p_e - p_m$ offsprings through the mating process (Gonçalves *et al.*, 2018a).

Following on Gonçalves et al. (2010), a parent can be selected multiple times for mating. The probability of selecting a specific elite individual for mating $(1/p_e)$ is greater than that of a

given non-elite individual ($1/(p-p_e)$), since $p_e < p - p_e$. Consequently, elite individuals have higher likelihood to transmit their characteristics to future generations than non-elite individuals (Gonçalves *et al.*, 2018a).

The mechanism used to implement mating in BRKGA is parametrized uniform crossover (Gonçalves *et al.*, 2010). The crossover probability, denoted as ρ_e, represents the likelihood of an off-spring inheriting an allele from its elite parent, and is a user-defined parameter. The offspring's allele takes on the value of the elite parent with a probability ρ_e, while it assumes the value of the non-elite parent with probability $1 - \rho_e$. Hence, the offspring is more likely to inherit characteristics of the elite parent than those of the non-elite parent.

Figure 3.3 depicts the crossover process for two-random key vectors, each consisting of five genes (Gonçalves *et al.*, 2010). In this particular example, ρ_e is set to 0.7, indicating that the offspring inherits the elite parent's allele with a probability of 0.7. To simulate this, a randomly generated number within the interval of $[0, 1[$ is used to simulate the outcome of a biased coin toss. If the resulting number is less than or equal to 0.7, the child inherits the allele of the elite parent. Otherwise, it inherits the allele of the non-elite parent.

Elite parent	0.1	0.5	0.2	0.4	0.3
Non-elite parent	0.3	0.4	0.1	0.2	0.6
Random number	0.3	0.8	0.2	0.9	0.1
Relation to crossover probability of 0.7	<	>	<	>	<
Offspring chromosome	0.1	0.4	0.2	0.2	0.3

Figure 3.3. Parametrized uniform crossover: mating in BRKGAs.

Once the new generation is completely created, *i.e.*, it has p individuals with fitness values calculated for all newly created random-key vectors, the population is again divided into elite and non-elite individuals, and the process restarts. This iterative process continues until a specific stopping criterion is met. Several stopping criteria can be employed, such as stopping after a time limit is reached, after a fixed number of generations from the beginning, or after a solution at least as good as a certain value is found (Gonçalves *et al.*, 2010).

Multi-population strategy

This work employs a multi-population strategy, where multiple populations evolve parallel and exchange high-quality chromosomes after a predefined number of generations. Fontes *et al.* (2013) observed that excessive and frequent chromosome exchanges can disrupt the evolutionary process. Thus, following on Fontes *et al.* (2023a) a strategy that replaces the worst

two elite chromosomes from each independent population with the best two chromosomes from the combined populations after 30 generations was implemented. Algorithm 1 presents the primary procedures for the proposed multi-population BRKGA.

Algorithm 1. Proposed multi-population BRKGA

```
Initialization of the control parameters for the BRKGA
Generate a random population using a new seed for the random number generator
    while stopping criteria is not satisfied, do
        for each independent population:
            Decode solutions for the current population,
            Evaluate the solutions (compute the makespan),
            Update the two best solutions,
            Partition the population into elite and non-elite according to fitness values,
            Generate offspring by parametrized uniform crossover,
            Generate mutants,
            Copy the elite group onto the next population,
            Add mutants and offspring to the new population,
            If it is time to exchange chromosomes, then:
                Replace the two worst individuals of the elite group by the two best
                individuals among the elite solutions of all populations,
    end while
Terminate the algorithm and print the makespan and time
```

Parameter setting

The BRKGA has some parameters (which influence its performance) that need to be set: the population size (p), the number of genes in a chromosome (N), the size of the mutant solution population (p_m), the size of the elite population (p_e), and the elite allele inheritance probability (ρ_e). Additionally, for the multi-population strategy, it is necessary to set the number of parallel populations (π), the frequency of population interchange (k_p), and the number of exchanged solutions (k_m).

The values recommended by Gonçalves *et al.* (2010, 2018b) are described in Table 3.1. Note that a is not a parameter, however, it directly influences the population size.

The parameters values' used in this work, which are based on previous work by Gonçalves *et al.* (2005), Fontes *et al.* (2023c), and Rizzi *et al.* (2015), were empirically selected by testing them on a subset of benchmark instances. The experiments were conducted using the setup specified in Table 3.1.

Table 3.1. Recommended parameter value settings.

Parameter	Recommended value	Value used in this work
p	$p = aN,\ 1 \leq a\ \in \mathbb{R}$	$\min\{500, 2N\}$
p_e	$0.10p \leq p_e \leq 0.25p$	$0.2p$
p_m	$0.10p \leq p_m \leq 0.30p$	$0.1p$
ρ_e	$0.5 \leq \rho_e \leq 0.8$	0.7
π	$\pi \in \{1, \ldots, 5\}$	3
k_p	$k_p \in \{50, \ldots, 100\}$	30
k_m	$k_m \in \{1,2,3\}$	2

Since metaheuristic algorithms are stochastic, each instance was solved ten times. The stopping criteria selected include 100 generations without improvement or run 500 generations, whichever happens first.

3.1.2 Problem-dependent part

One of the features that differentiates conventional GAs is that the algorithm does not directly handle the problem's solutions but instead operates on a solution representation known as the chromosome. The algorithm evolves the population by evolving encoded solutions rather than "real" solution. The BRKGA encodes a solution through a chromosome with N alleles, where N is the number of production operations required. Each allele is a real value between $[0, 1[$ and is initially (first generation) randomly generated. In a JSP, a solution can be encoded as a vector where 1 to n_1 refer to operations of job 1, elements $n_1 + 1$ to $n_1 + n_2$ the ones of job 2, and so on, as represented in Figure 3.4 (Homayouni *et al.*, 2023).

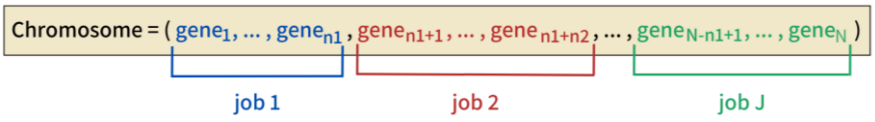

Figure 3.4. Chromosome structure for a JSP, with J jobs and N operations.

Thus, to perform a mapping between the space of random keys and the real space of the problem, a deterministic algorithm called a decoder is used, which is responsible for returning a feasible solution and its corresponding objective value or fitness from a vector of random keys (Gonçalves *et al.*, 2014).

Decoding procedure

The decoding process is illustrated by resorting the example considered by Gonçalves *et al.* (2014), which is described in Table 3.2.

Table 3.2. Example of an instance with 4 jobs and 3 machines (Gonçalves et al., 2014).

	Job 1		Job 2		Job 3		Job 4	
Order	Machine	Process time	Machine	Process time	Machine	Process time	Machine	Process time
1	a	2	B	3	c	5	b	2
2	b	3	C	2	b	2	a	4
3	c	4	A	3	a	3	c	2

A solution to a JSP with four jobs, (*e.g.*, four products), each requiring three production operations, can be represented (encoded) as given in Figure 3.5a. Elements 1 to n_1 (3) refer to the operations of job 1, elements $n_1 + 1$ (4) to $n_1 + n_2$ (6) to the ones of job 2, and so on.

Figure 3.5. Decoding a chromosome into a list of ordered operations.

The vector is filled with random keys, *i.e.*, a real number in the interval $[0, 1[$, as in Figure 3.5a. The decoding method starts by sorting the vector in ascending order (Figure 3.5b). Then, it converts the indices of the sorted vector into a sequence of repeated jobs. Subsequently, this sequence of repeated jobs is translated into a sequence of operations by replacing (from left to right) the i^{th} appearance of job j by operation O_{ij}, ensuring, this way, that the solutions are always feasible (Homayouni *et al.*, 2023), as represented in Figure 3.5c.

Once a list of ordered operations is obtained, a schedule is created by sequentially scheduling each operation, starting with the first operation, then the second, and so on. To process an operation, the previous operation of the same job must be completed, unless, it is the first operation of a job, and the machine it is processed on must be free. Each operation is allocated to the earliest starting time as long as the two conditions are satisfied (just the latter if it a job first operation). This process is iterated until all operations have been scheduled. The Gantt chart corresponding to the chromosome in Figure 3.5a can be found in Figure 3.6. The scheduling is not optimized, but it is feasible.

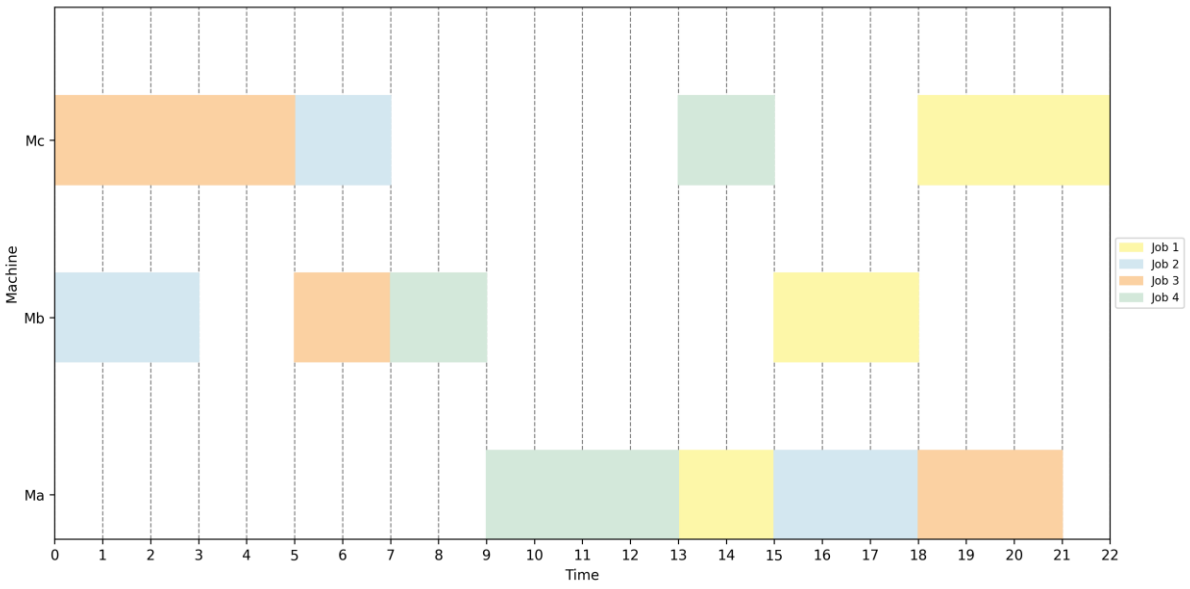

Figure 3.6. The Gantt chart illustrating the example defined in the Table 3.2 and corresponding to the chromosome depicted in the Figure 3.5a.

Fitness function

The objective of the fitness function is to assess each chromosome's quality, enabling the evolutionary process to distinguish between different chromosomes and to directs the search for better solutions (Gonçalves *et al.*, 2018a). Depending on the problem, different measures can be employed, such as makespan (as implemented in this work), the sum of the total tardiness and earliness, the number of tardy jobs, and mean completion times, among others.

3.2 Deterioration effects

In the chemical industry, the production of certain products may contain components that deteriorate over time, causing an increase in the processing time. For instance, in the production of porcelain craftworks, the raw material, made up of clay and special coagulants, becomes harder over time, which may increase the time required to shape a craftwork. Other examples can be found in the food industry: powders absorb water when handled in high-humidity environments. Due to liquid bridges between particles (Teunou *et al.*, 1999), they become less flowable, and hence, longer times are required to process, transport and store them. Also, the mixing time is affected by the mixture's viscosity, which is affected by the time the mixture spends waiting for the mixing step since it may lead to a decrease in temperature or solvent evaporation.

Furthermore, production systems often undergo changes in the physical conditions of their machines as a result of overheat, fouling, impurities, or other phenomena, leading to degradation. These changes cause a reduction in machine efficiency and an increase in processing times (Wang *et al.*, 2018). For example, if a reaction must meet a given conversion,

the presence of fouling on the catalyst will cause an increase in processing time. Also, machine fouling (for instance, in the of case of powder in the food industry) results in lower production rates, and hence, it takes longer to produce the required amounts, and/or reduces the heat transfer.

To study such impacts in production, deterioration effects were considered. The two types of deterioration mentioned before were studied: (1) job deterioration, in which processing times depend on the waiting time between two consecutive operations of the same job (components' degradation); (2) machine deterioration, in which processing times depend on machine's running time (loss of machine/system efficiency).

Typically, when machines start to deteriorate, maintenance is necessary to maintain acceptable production levels (Renna, 2015). Although the study's primary objective is not maintenance, it was incorporated into the machine deterioration optimization process to explore its potential impact on improving makespan in the context of machine deterioration.

To study the influence of deterioration, three types of functions to mode the impacts on processing time were considered, namely: linear, exponential, and sigmoid functions. Note that deterioration effects considering sigmoid functions are here studied for the first time. However, sigmoid functions are of particular relevance since in many processes the conditions leading to processing time's deterioration tend to approach equilibrium, thereby causing processing times to approach a limit. For example, when the temperature of an agitator increases, its efficiency decreases and consequently, the time to reach the same level of mixing increases. However, the agitator temperature stabilizes after a certain period, and the processing time remains constant.

Following the work by Wu *et al.* (2019), it was considered that deterioration has a limited impact on processing time. The obviously case in real world production systems is the example of powder flow properties degradation by humidity, described above. The absorbed humidity reaches an equilibrium, affecting the processing time in a restricted way. For the sigmoid function, it was considered acceptable to assume that the processing time stabilizes at twice its processing time. For the exponential function, empirical tests were performed to define the limit.

The processing times were expressed as a function of the job waiting time (job deterioration) and the machine running time (machine deterioration). Equations (1) to (3) model the processing time under job deterioration, and equations (4) to (6) model the processing time under machine deterioration. In all equations, $p_{[ij]}$ refers to the actual processing time of operation i of job j, while p_{ij} refers to the reference processing time of the same operation, *i.e.*, without deterioration.

$$p_{[ij]} = p_{ij} + \alpha_{ij}(t_{ij} - t'_{(i-1)j}) \tag{1}$$

$$p_{[ij]} = \begin{cases} p_{ij}e^{\alpha_{ij}\left(t_{ij} - t'_{(i-1)j}\right)}, & if\ (t_{ij} - t'_{(i-1)j}) \leq \dfrac{p_{ij}}{10} \\ p_{ij}e^{\frac{\alpha_{ij} \times p_{ij}}{10}}, & if\ (t_{ij} - t'_{(i-1)j}) > \dfrac{p_{ij}}{10} \end{cases} \tag{2}$$

$$p_{[ij]} = \frac{2 \times p_{ij}}{1 + e^{-\alpha_{ij}\left(t_{ij} - t'_{(i-1)j}\right)}} \tag{3}$$

$$p_{[ij]} = p_{ij} + \alpha_m \sum_{d \in J} \sum_{l \in O_d : m_{dl} = m_{ij} \wedge S_{dl} < S_{ij}} p_{[dl]} \tag{4}$$

$$p_{[ij]} = \begin{cases} p_{ij}e^{\alpha_m \sum_{d \in J} \sum_{l \in O_d : m_{dl} = m_{ij} \wedge S_{dl} < S_{ij}} p_{[dl]}}, & if \sum_{d \in J} \sum_{l \in O_d : m_{dl} = m_{ij} \wedge S_{dl} < S_{ij}} p_{[dl]} \leq \dfrac{p_{ij}}{10} \\ p_{ij}e^{\frac{\alpha_m \times p_{ij}}{10}}, & if \sum_{d \in J} \sum_{l \in O_d : m_{dl} = m_{ij} \wedge S_{dl} < S_{ij}} p_{[dl]} > \dfrac{p_{ij}}{10} \end{cases} \tag{5}$$

$$p_{[ij]} = \frac{2 \times p_{ij}}{1 + e^{-\alpha_m \sum_{d \in J} \sum_{l \in O_d : m_{dl} = m_{ij} \wedge S_{dl} < S_{ij}} p_{[dl]}}} \tag{6}$$

In equations (1) to (3), the waiting time between two consecutive operations of the same job is given by the difference between the starting time of operation i of job j (t_{ij}) and the time at which the operation $i-1$ of job j has been completed ($t'_{(i-1)j}$). In these equations, α_{ij} represents the deterioration rate of job j due to the waiting time imposed before starting operation i, which is machine independent.

In equations (4) to (6), the running time is expressed by the sum of the processing time of the operations since the machine has started operation. Hence, for each machine the running time is calculated as the sum of the processing times over all operations it processes and that have been started on or before t_{ij}. In these equations, α_m represents the deterioration rate of each machine, which is job independent.

Prediction the deterioration rate is very complex in batch processes with multiple products. Different products require different batch recipes, such as varying raw material ratios, additives, and operating conditions, resulting in distinct deterioration processes (Wu *et al.*, 2018). Consequently, predicting the deterioration rate becomes challenging and requires real-time data. Thus, in this study, the deterioration rates were randomly drawn from the uniform distribution $U[0.005, 0.200]$, as is standard in the literature (Wu *et al.*, 2020; Li *et al.*, 2021).

In an attempt to bring this study closer to reality, deterioration was not considered to affect machines or all operations. Indeed, it was decided to have a fixed percentage of machines and of operations to be affected by deterioration. Hence, 25 % of the machines and of the operations were randomly selected.

4 Results and discussion

The computational experiments were conducted on the 10 JSP instances proposed by Lawrence (1984), since they are commonly used in many JSP studies. The configurations of each instance, *i.e.*, processing times and order of operations, can be found in Annex A, along with the deterioration rate of operations affected by this effect. Annex B contains the deteriorating machines and their respective rates for each instance and in Annex C the deterioration functions addressed are represented.

The tests were conducted on a computer with an Intel(R) Core(TM) i5-2400 Processor (3.10 GHz) running on the Windows 10 operating system, and the algorithm was implemented in Python® 3.11.

4.1 BRKGA efficiency

To demonstrate the effectiveness of the BRKGA to solve JSP, instances from Lawrence (1984) were solved. Table 4.1 shows the instances' settings, as well as the optimal total production time (makespan) from literature (Opt.), the best solution found, the percent optimality gap (GAP), the percentage of makespan deviation ($\sigma = (average - best)/average \times 100$) and the CPU time needed to find the best solution.

Table 4.1. Lawrence (1984) instances and computational results for BRKGA.

Characteristics			BRKGA			
Instance	M-J-O	Opt.	Best	GAP	σ/%	t/s
la04	5-10-50	590	597	1.2	3.3	17
la08	5-15-75	863	**863**	0.0	0.1	11
la12	5-20-100	1039	**1039**	0.0	0.0	6
la16	10-10-100	945	946	0.1	5.2	29
la17	10-10-100	784	796	1.5	3.6	30
la22	10-15-150	927	989	6.7	3.5	50
la23	10-15-150	1032	1045	1.3	3.0	84
la28	10-20-200	1216	1269	4.4	4.5	157
la31	10-30-300	1784	**1784**	0.0	0.2	95
la36	15-15-225	1268	1340	5.7	4.1	159
Mean		1044.8	1066.8	2.1	2.7	64

Overall, the BRKGA performs well in these instances, with the mean best solution of 1066.8, which is only 2.1 % away from the mean optimal solution. The algorithm also has a relatively

low standard deviation, indicating consistent performance, *i.e.*, robustness. The CPU time required to find the optimal solution varies depending on the instance's configuration, as instances with a higher number of operations take longer. Nevertheless, its average value is only 64 seconds.

Then, the multi-population was implemented (BRKGA+MP), and the results obtained compared to those of i) Rizzi *et al.* (2015) that employed clustering search in BRKGA (BRKGA+CS), and those of ii) Gonçalves *et al.* (2005) that purposed a hybrid genetic algorithm with local search (HGA+LS). The results are summarized in Table 4.2. It is important to take into account that the literature values were obtained from different computers in different years, which may influence on the reported resolution times.

Table 4.2. Computational results for BRKGA+MP and comparation with literature values.

	BRKGA+MP				BRKGA+CS				HGA+LS		
Instance	Best	GAP	σ/%	t/s	Best	GAP	σ/%	t/s	Best	GAP	t/s
la04	**590**	0.0	1.9	36	**590**	0.0	7.8	14	**590**	0.0	42
la08	**863**	0.0	0.0	8	**863**	0.0	5.4	6	**863**	0.0	99
la12	**1039**	0.0	0.0	7	**1039**	0.0	6.1	6	**1039**	0.0	201
la16	946	0.1	4.8	49	946	0.1	10.8	22	**945**	0.0	232
la17	785	0.1	1.7	50	**784**	0.0	8.8	47	**784**	0.0	216
la22	954	2.9	3.3	74	942	1.6	12.0	25	950	2.5	629
la23	**1032**	0.0	2.3	93	**1032**	0.0	8.1	53	**1032**	0.0	594
la28	1241	2.1	4.5	152	1258	3.5	10.9	11	1250	2.8	1267
la31	**1784**	0.0	0.2	101	**1784**	0.0	4.5	67	**1784**	0.0	3745
la36	1315	3.7	3.5	85	1315	3.7	10.3	75	1303	2.8	1826
Mean	1054.9	0.9	2.2	66	1055.3	0.9	8.5	33	1054.0	0.8	885

The results show that BRKGA and BRKGA+MP performed well, but the BRKGA+MP consistently provided better solutions. This variant obtained solutions that were either optimal or very close to optimality. Additionally, it is very robust since as the standard deviation values are quite small. It is important to note that the best solution obtained for the la28 instance by BRKGA+MP is better than the best one found by both the BRKGA+CS and the HGA+LS.

In comparison, the BRKGA+CS variant performed worse than BRKGA+MP, with a little higher mean best solution and a significantly higher standard deviation (about four times larger).

Nevertheless, it required less time than the BRKGA+MP variant (about half). On the other hand, HGA+LS found slightly better solutions in most cases, with lower relative gaps compared to the other algorithms. However, this algorithm took significantly more time to find these solutions in all cases (about 14 larger than that of the BRKGA+MP). For example, for the instance with the highest number of operations (la31) it took over 1 hour. Additionally, nothing can be said about method robustness since no standard deviations are reported.

The computational time required by BRKGA and its variants was generally moderate, with most instances solved within a few minutes or less. However, the time necessary to find the best solution was higher for BRKGA+MP, which was expected since this variant involves multiple populations. Based on the results obtained, BRKGA+MP was chosen to tackle the JSP with deterioration effects.

4.2 Job deterioration

To analyse the impact of deterioration effects, the following two strategies were used to calculate the makespan. Firstly, for the best solutions found by BRKGA+MP, *i.e.*, disregarding the deterioration, the makespan was recalculated by considering the deterioration afterward, that is, without optimizing the "real" processing time (w/out opt.). Then, the instances were solved again, considering processing times that incorporate the deterioration (w/ opt.). The results of both cases are provided in Table 4.3, as well as the computational time required for the optimization, the makespan improvement percentage (I), and the standard deviation percentage. Figure 4.3 depicts the solutions obtained (Gantt chart), for both cases.

As it can be seen from Table 4.3, for all instances, the deterioration optimized solution is better. The largest improvement can be observed in the presence of the exponential deterioration; the makespan values are, on average, 38 % lower. Indeed, the deterioration effects are almost nullified as the optimized makespan is, on average, just 9 % above the makespan disregarding deterioration.

In the linear case, the improved solutions have makespans that are, on average, 4.7 % lower. Although this number seems to be small, it is indeed quite large as the deterioration impacts on makespan are only about 6 %. Hence, the optimization was able to eliminate almost 80 % of the imposed increase, as shown in Figure 4.3a.

Regarding the exponential function, an average improvement of about 38 % was obtained. In this case, by optimizing considering the deterioration effects, about 70 % of the imposed increase on the makespan is recovered (Figure 4.3b). The improvement is as high as 68.6 %, for instance la22. This case proves the extreme importance of taking deterioration into account in industrial processes since an uncontrolled schedule can lead to a significant loss of productivity and a large increase of operational costs.

Table 4.3. Computational results for linear, exponential, and sigmoid job deterioration.

	Linear					Exponential					Sigmoid				
	W/out opt.	W/ opt.	t/s	I/%	σ/%	W/out opt.	W/ opt.	t/s	I/%	σ/%	W/out opt.	W/ opt.	t/s	I/%	σ/%
la04	621.7	600.2	17	3.6	3.8	752.8	667.8	31	12.7	5.8	728.8	663.0	25	9.9	2.5
la08	959.3	865.0	118	10.9	1.4	1790.1	1256.3	82	42.5	1.5	1245.9	985.9	131	26.4	4.4
la12	1145.9	1043.1	69	9.9	0.8	1759.0	1122.4	68	56.7	1.5	1338.9	1137.0	112	17.8	2.1
la16	1014.7	966.6	175	5.0	3.0	1259.5	1024.0	30	23.0	3.7	1173.0	1024.0	57	14.5	2.8
la17	799.2	788.0	61	1.4	2.2	1064.0	819.0	81	29.9	3.1	1010.7	802.3	85	26.0	5.4
la22	986.9	959.4	149	2.9	3.4	1749.6	1037.5	152	68.6	3.8	1283.4	1030.0	192	24.6	5.2
la23	1066.2	1033.5	206	3.2	1.7	1621.3	1088.2	60	49.0	4.1	1233.5	1083.6	142	13.8	5.2
la28	1292.7	1291.1	217	0.1	4.2	2074.3	1384.6	179	49.8	6.6	1654.8	1406.7	216	17.6	3.3
la31	1940.1	1792.0	275	8.3	2.2	2605.7	1912.6	261	36.2	2.8	2255.5	1928.6	293	16.9	3.5
la36	1365.3	1335.1	222	2.3	3.9	1558.0	1386.7	247	12.4	3.9	1539.2	1412.3	190	9.0	2.2
Mean	1119.2	1067.4	151	4.7	2.7	1623.4	1169.9	119	38.1	3.7	1346.4	1147.3	144	17.7	3.7

Figure 4.1 and Figure 4.2 depict the Gantt charts for the la22 instance without optimizing and optimizing the deterioration effects, respectively. By analysing these figures, is clear that the makespan can be greatly reduced, which is accomplished by reducing the machines' idle time.

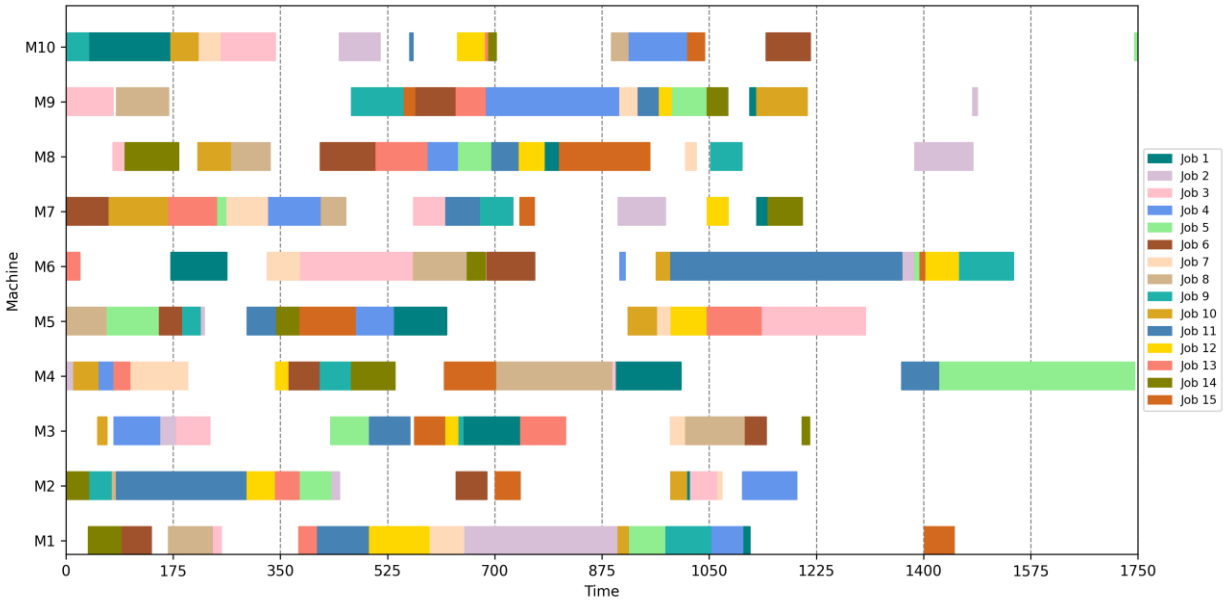

Figure 4.1. Instance la22 schedule considering exponential job deterioration without optimization.

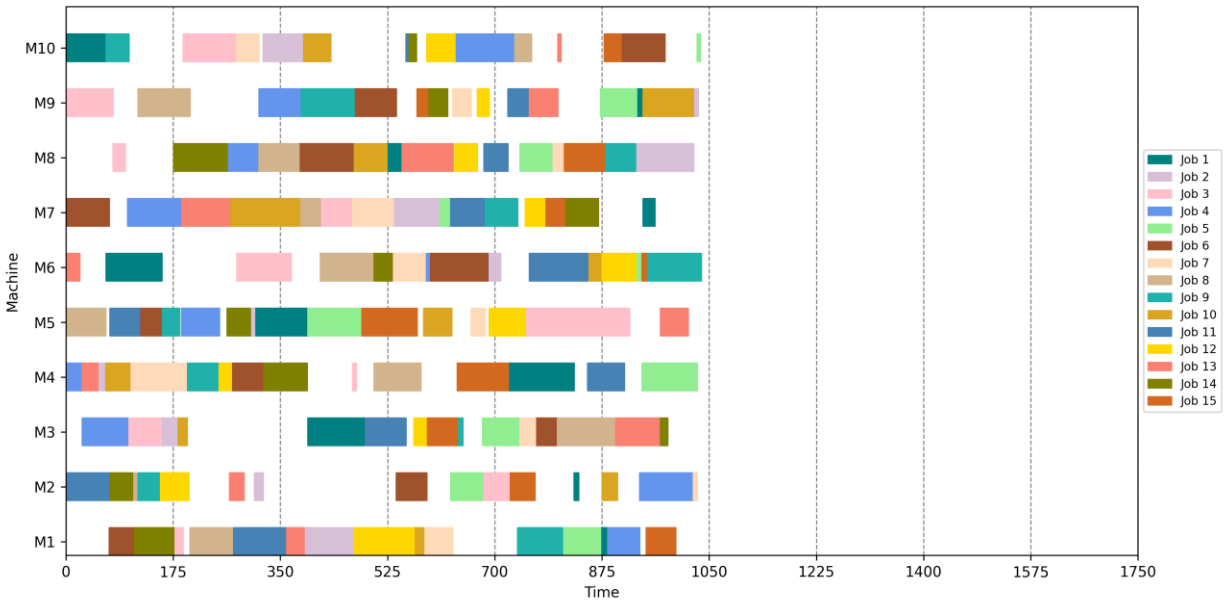

Figure 4.2. Instance la22 schedule considering exponential job deterioration with optimization.

In the sigmoid case, the average improvement amounts to 17.7 %. This type of deterioration function is the harder to compensate. Nevertheless, most of the deterioration impacts have been mitigated since about 64 % of the deterioration impacts can be avoided (Figure 4.3c).

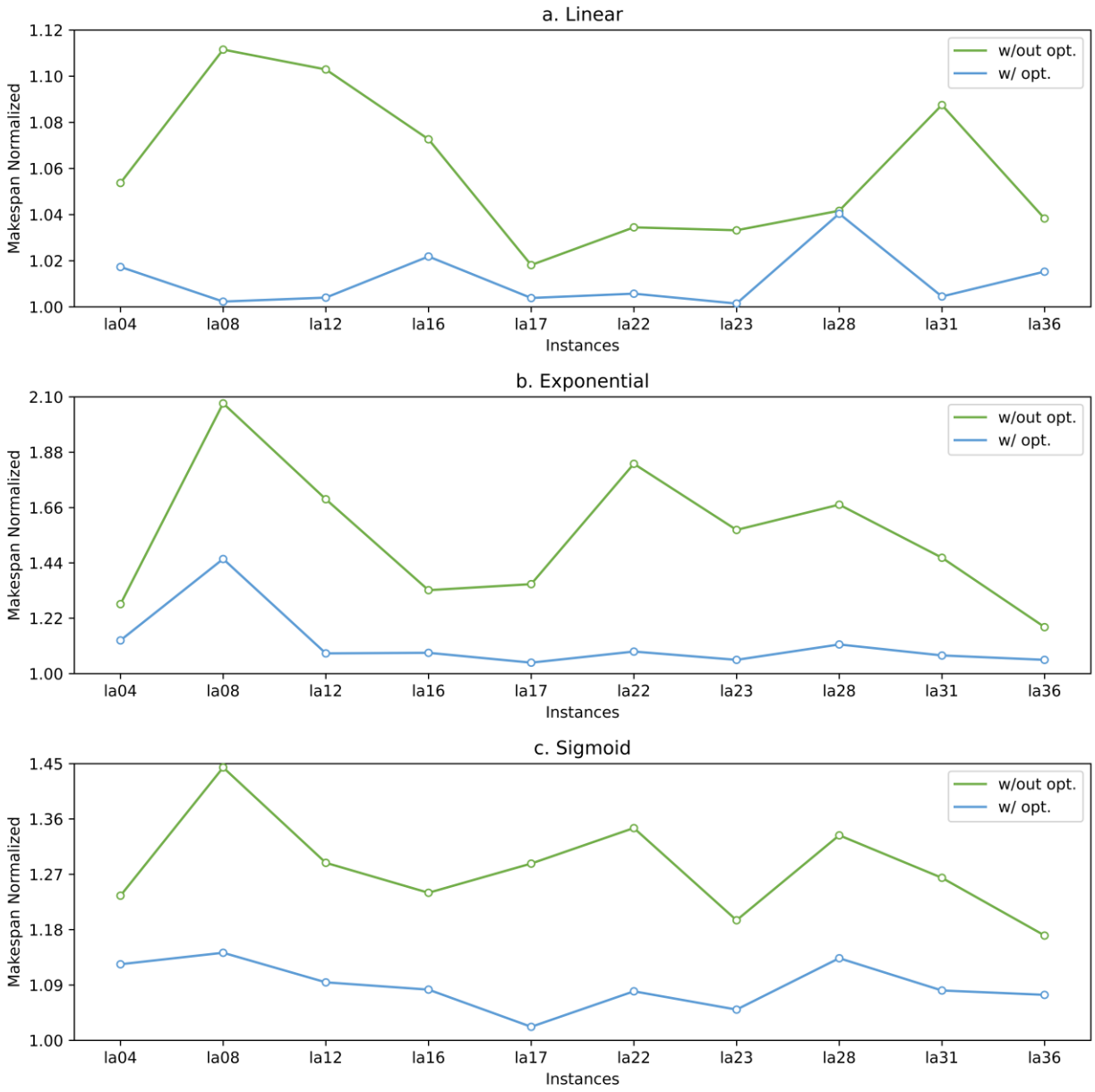

Figure 4.3. Normalized makespan for each instance with and without optimization for job deterioration.

Furthermore, for all functions, both CPU time and standard deviation increased compared to those obtained for BRKGA+MP in Section 4.1. This is due to the increased complexity of the problem when deterioration is considered.

From Figure 4.3, it can be observed that the impact of deterioration is not too much related to the total number of operations in the instances but mainly depends on the reference processing time and the deterioration rate, as explained in Annex C.

Overall, when job deterioration is considered in JSP, it allows for a more realistic modelling of the actual production process since deterioration is a common occurrence in many industries. By accounting for job deterioration, the scheduling algorithm can assign jobs more efficiently,

resulting in a shorter makespan compared to cases where deterioration is not considered but still occurs.

4.3 Machine deterioration

The same procedure was applied for machine deterioration. The results are summarized in Table 4.4, which reports the computational time required for the optimization, the makespan improvement percentage, and the standard deviation percentage.

In the case of machine deterioration, the mitigation of the deterioration effects is not as effective as that of job deterioration; nevertheless, they are still partially mitigated.

Unlike job deterioration, where the reordering of certain tasks facilitates the reduction or even elimination of waiting times between consecutive operations within the same job, machine deterioration, despite operational reorganization, maintains a fixed reference processing time, *i.e.*, the machines take longer to process the operations regardless of the operations sequence.

In this case, the only way of recovering machine efficiency is performing machine maintenance. Hence, maintenance activities were introduced. Whenever the machine processing time is twice the reference processing time, the machine is subject to maintenance (w/ maint.). For each machine, the maintenance time was determined as the average of the reference processing times associated with that particular machine. The results are summarized in Table 4.5. Figure 4.4 shows the normalised makespan for the three cases, namely: without optimizing the deterioration effects, optimizing the deterioration effects by just considering operation sequence, and optimizing the deterioration effects by also performing maintenance.

Regardless of the deterioration function under consideration, optimization addressing deterioration proved to be better than considering it after solving the problem. The optimization proved to be more efficient in the linear case, in which the makespan improved, on average, about 23 %. In this case, the deterioration had the highest impact on makespan, increasing by 234 %. Hence, the optimization eliminated about 10 % of the imposed increase, see Figure 4.4a. The improvement in the linear deterioration is as high as 92.8 %, for instance la31, which experienced the highest increase in the makespan. This case proves, again, the extreme importance of taking deterioration into account in industrial processes.

In the presence of an exponential deterioration, an improvement of about 7 %, on average, was obtained. In this case, by optimizing considering the deterioration effects, about 10 % of the imposed increase on the makespan was reduced (Figure 4.4b). In the sigmoid case, the average improvement amounts to 9 %. Still, it was able to mitigate the deterioration effects since about 13 % of the deterioration impact can be avoided (Figure 4.4c). However, this case was the case where deterioration caused a lower increase in makespan, only increasing by about 65 %.

Table 4.4. Computational results for linear, exponential, and sigmoid machine deterioration.

	Linear					Exponential					Sigmoid				
	W/out opt.	W/ opt.	t/s	I/%	σ/%	W/out opt.	W/ opt.	t/s	I/%	σ/%	W/out opt.	W/ opt.	t/s	I/%	σ/%
la04	719.9	658.2	17	9.4	1.0	783.5	748.5	3	4.7	0.0	1043.3	1008.3	5	3.5	0.0
la08	871.4	863.0	14	1.0	0.0	879.5	863.0	16	1.9	0.0	1444.2	1405.6	24	2.7	0.0
la12	4407.6	3438.8	69	28.2	0.1	2699.0	2481.9	1	8.7	0.0	2051.9	1989.0	0	3.2	0.0
la16	1116.6	1060.3	108	5.3	1.6	1339.0	1156.2	20	15.8	0.2	1357.5	1186.0	24	14.5	2.8
la17	1314.6	1197.7	73	9.8	0.4	1362.3	1269.3	17	7.3	0.0	1221.0	1128.0	27	8.2	0.4
la22	2346.8	1844.1	72	27.3	3.1	1546.9	1541.9	6	0.3	0.0	1708.6	1501.1	52	13.8	0.0
la23	2643.3	2191.8	136	20.6	1.0	2313.4	2162.4	7	7.0	0.0	1714.0	1567.0	35	9.4	0.0
la28	3174.0	2772.5	115	14.5	0.6	2089.8	2015.3	43	3.7	0.0	2225.8	2099.8	120	6.0	0.1
la31	16774.9	8699.1	235	92.8	2.0	3472.9	3187.1	11	9.0	0.0	2865.8	2717.0	70	5.5	0.0
la36	1817.7	1514.8	194	20.0	2.4	1547.3	1386.0	232	11.6	3.9	1855.0	1556.7	204	19.2	3.2
Mean	3518.7	2424.1	103	22.9	1.2	1803.4	1681.2	36	7.0	0.4	1748.7	1615.9	56	8.6	0.7

Table 4.5. Computational results for linear, exponential, and sigmoid machine deterioration with maintenance activities.

	Linear					Exponential					Sigmoid				
	W/out opt.	W/ opt.	t/s	I/%	σ/%	W/out opt.	W/ opt.	t/s	I/%	σ/%	W/out opt.	W/ opt.	t/s	I/%	σ/%
la04	719.9	643.0	10	12.0	0.5	783.5	748.5	6	4.7	0.0	1043.3	973.3	5	7.2	0.0
la08	871.4	863.0	13	1.0	0.2	879.5	863.0	0	1.9	0.0	1444.2	1405.6	38	2.7	0.0
la12	4407.6	1459.0	72	202.1	0.2	2699.0	1857.9	16	45.3	0.3	2051.9	1850.0	52	10.9	0.0
la16	1116.6	1014.2	24	10.1	1.3	1339.0	1097.6	1	22.0	0.0	1357.5	1186.0	32	14.5	0.8
la17	1314.6	926.1	46	42.0	0.6	1362.3	1025.5	40	32.8	2.3	1221.0	1100.0	25	11.0	0.8
la22	2346.8	1076.8	93	117.9	1.6	1546.9	1254.9	31	23.3	1.1	1708.6	1501.4	99	13.8	0.0
la23	2643.3	1205.4	45	119.3	1.9	2313.4	1460.3	3	58.4	0.0	1714.0	1458.0	36	17.6	0.4
la28	3174.0	1490.4	103	113.0	1.5	2089.8	1889.0	2	10.6	0.0	2225.8	2006.7	106	10.9	0.6
la31	16774.9	1983.1	238	745.9	1.0	3472.9	2563.5	0	35.5	0.0	2865.8	2548.9	217	12.4	0.8
la36	1817.7	1368.0	151	32.9	4.4	1547.3	1402.6	274	10.3	1.6	1855.0	1544.1	202	20.1	3.7
Mean	3518.7	1202.9	79	139.6	1.3	1803.4	1416.3	37	24.5	0.5	1748.7	1557.4	81	12.1	0.7

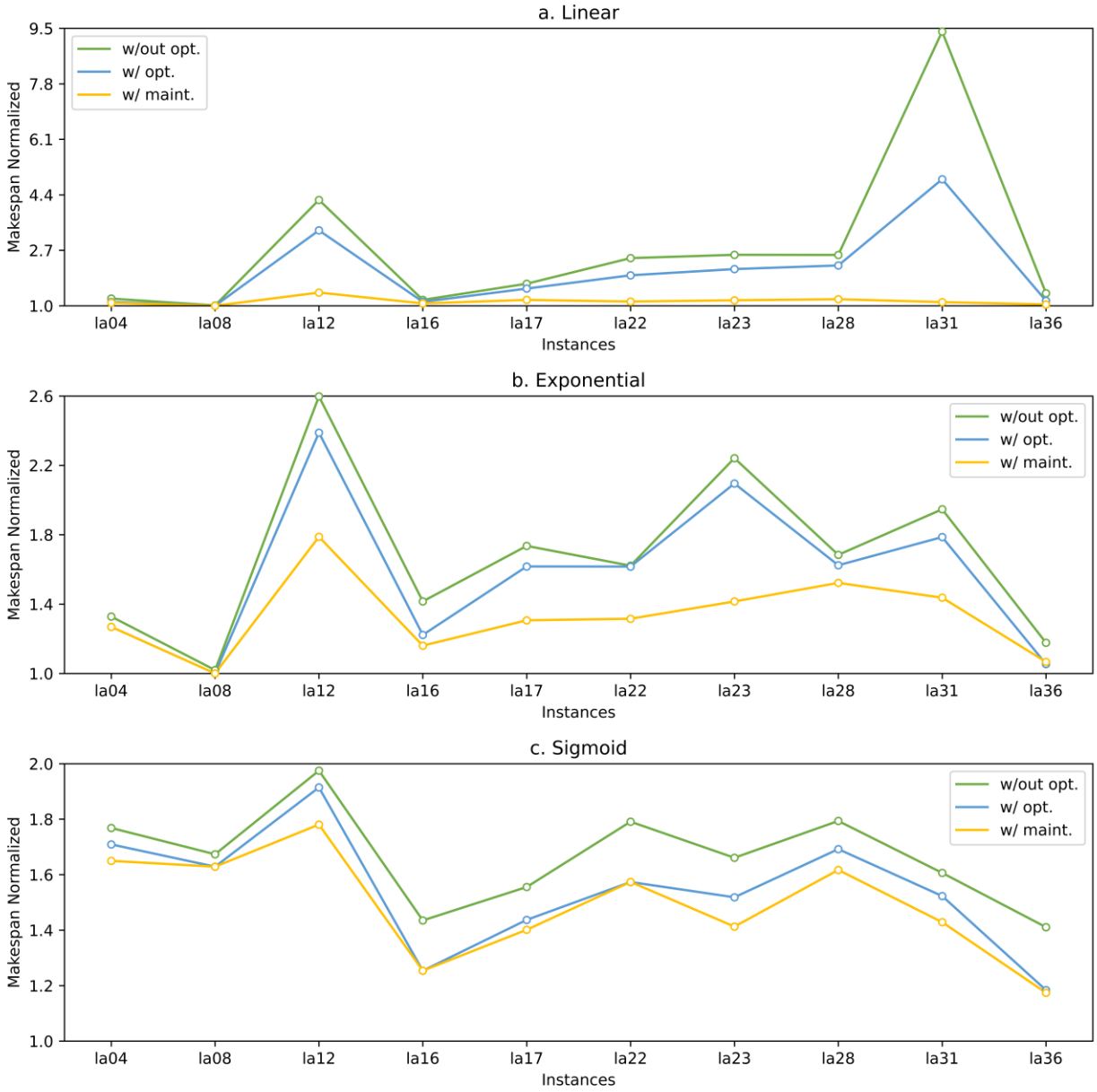

Figure 4.4. Makespan normalized for each instance with and without optimization and with maintenance.

Figure 4.5 and Figure 4.6 depict the Gantt charts for linear deterioration of instance la31 without and with optimization, respectively. It is worth noting that, contrary to the non-optimized scenario where operations on machines 9 and 10 (machines with deterioration) start at time 0, such is not the case in the optimized scenario for machine 10. This suggests that despite the "lost" time during which the machine is not operating, it subsequently enables an enormous reduction in makespan. In addition, there is also a rearrangement in the order of operations on machine 9 and 10, which likely contributed to the decrease in operation time.

The instance la31 improvement for linear deterioration was already significant; however, with maintenance, this instance exhibits a remarkable improvement of 746 % compared to the case

without optimization, as clearly depicted in Figure 4.4. This value reinforces the importance of considering maintenance in optimizing scheduling problems rather than solely performing maintenance after the end of production, as commonly practiced in many industries. As can be easily observed in Figure 4.7 (Gantt chart for this instance with maintenance), maintenance has dramatically reduced the operation time, making it almost imperceptible which machines are experiencing deterioration. The blank spaces on machines 9 and 10 correspond to maintenance periods, except for the blank space between the operation of job 30 and job 5, on machine 9.

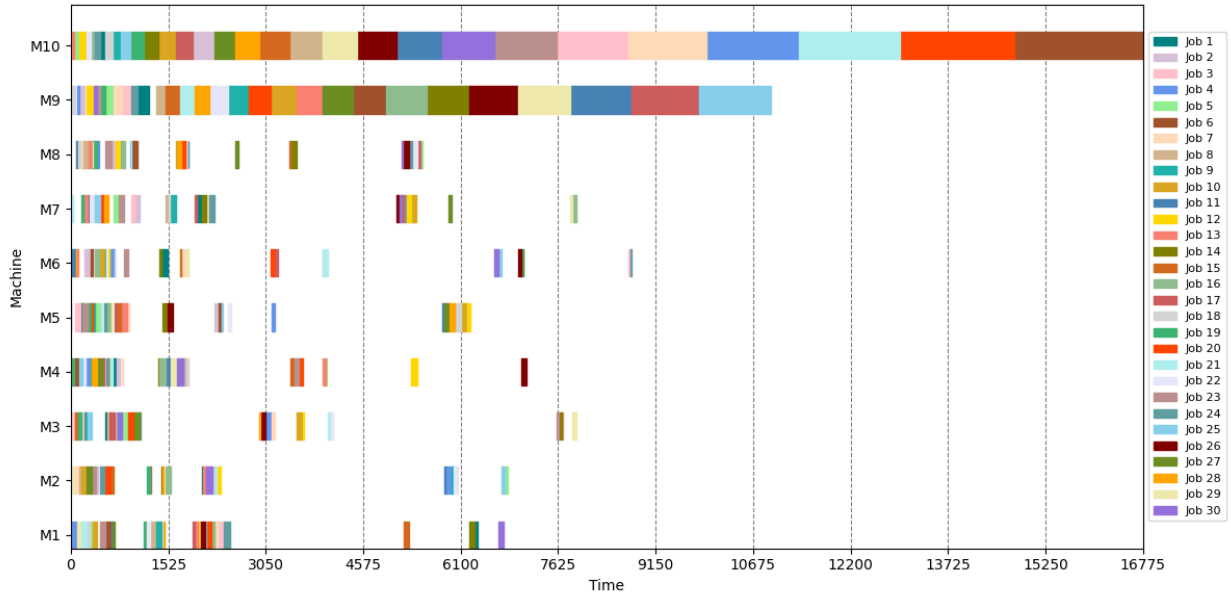

Figure 4.5. Instance la31 schedule considering linear machine deterioration without optimization.

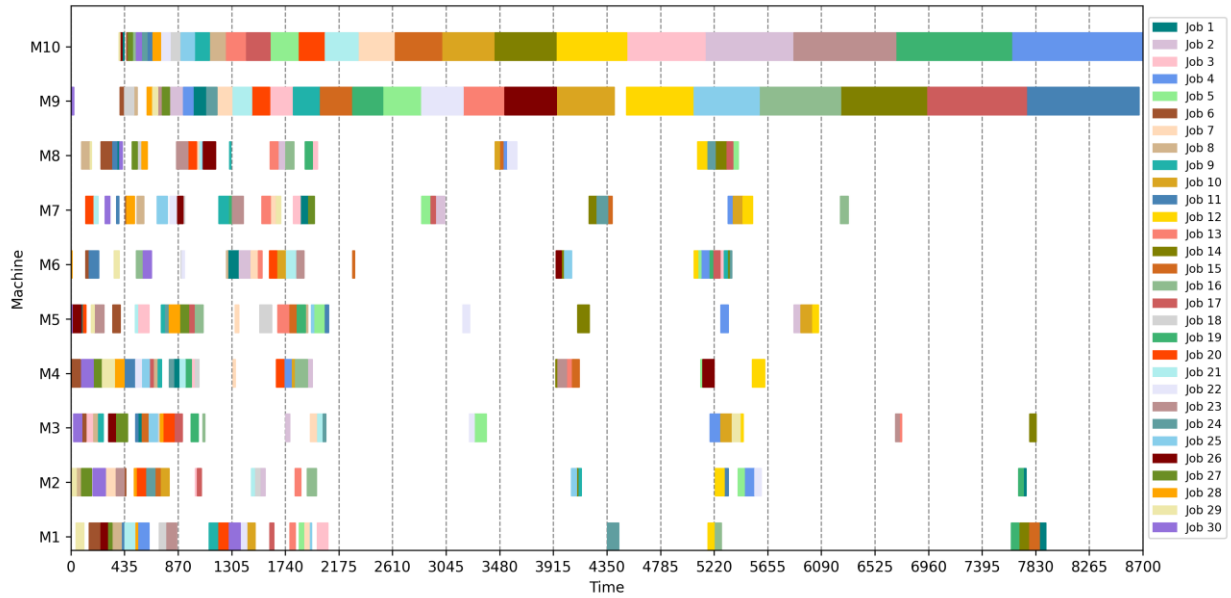

Figure 4.6. Instance la31 schedule considering linear machine deterioration with optimization.

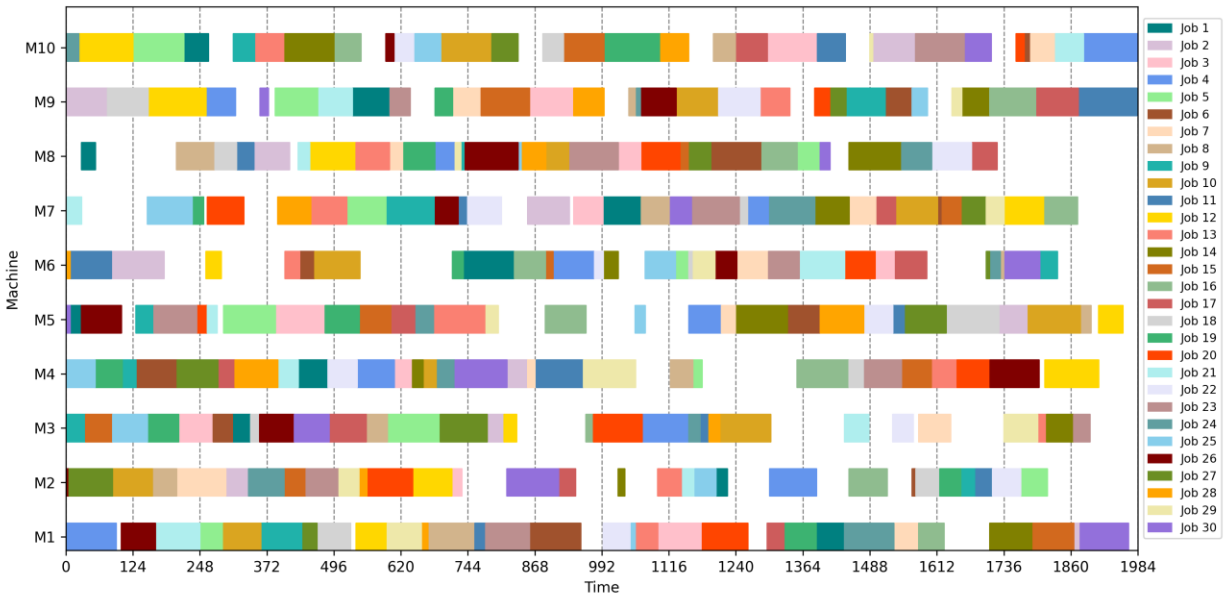

Figure 4.7. Instance la31 schedule considering linear machine deterioration with optimization and maintenance.

For machine deterioration, the standard deviation and computational time values are low compared to the results obtained in previous sections, with the computational time reaching 0 for instance la12 in the case of the sigmoid function (the makespan value was obtained from the initial population for seven out of the 10 runs and in the other three it required two generations).

In order to better analyse the behaviour, the Gantt charts of three solutions of instance la12 were drawn, see Figure 4.8, Figure 4.9, and Figure 4.10. Although these solutions are different and have different corresponding chromosomes, the makespan is the same. Despite the algorithm generating other different sequences, they consistently result in the same makespan, hence this is not due to excessive convergence; it is likely that the makespan found is optimal.

These three sequences share a common factor. The first two operations performed by the deteriorating machine (machine 2) correspond to jobs 6 and 13, respectively. The same pattern was observed in the remaining seven found solutions. Nevertheless, upon examining Figure 4.11, which represents the non-optimized schedule, it can be observed that the deteriorating machine does not start with these jobs. This suggests that, for this particular instance, initiating with jobs 6 and 13 could correspond to the optimal solution, regardless of the remaining sequence.

Despite these results, a modification to the alleles of instance la12 chromosome, was introduced to assess whether such a change would impact the makespan. Specifically, alleles with values lower than 0.5 were multiplied by 2, while alleles with values greater than 0.5 were divided by 2. Despite this abrupt alteration, the obtained results remained unchanged, and jobs

6 and 13 remained as the first ones to be processed by machine 2, suggesting that the obtained value may represent the global minimum.

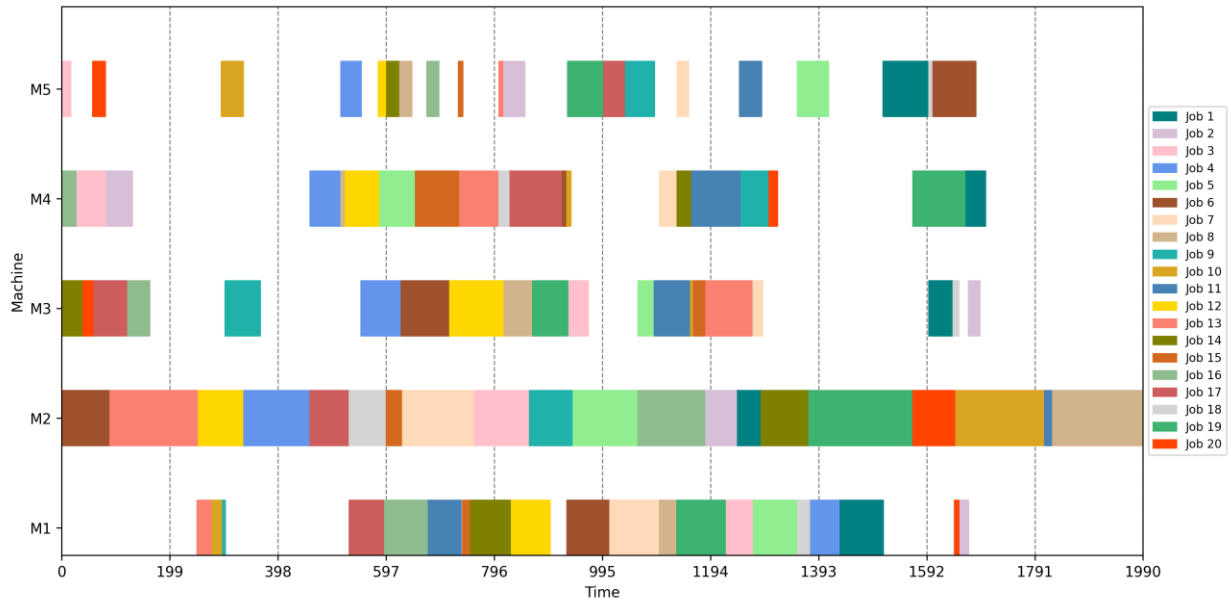

Figure 4.8 Instance la12 first schedule considering sigmoid machine deterioration with optimization.

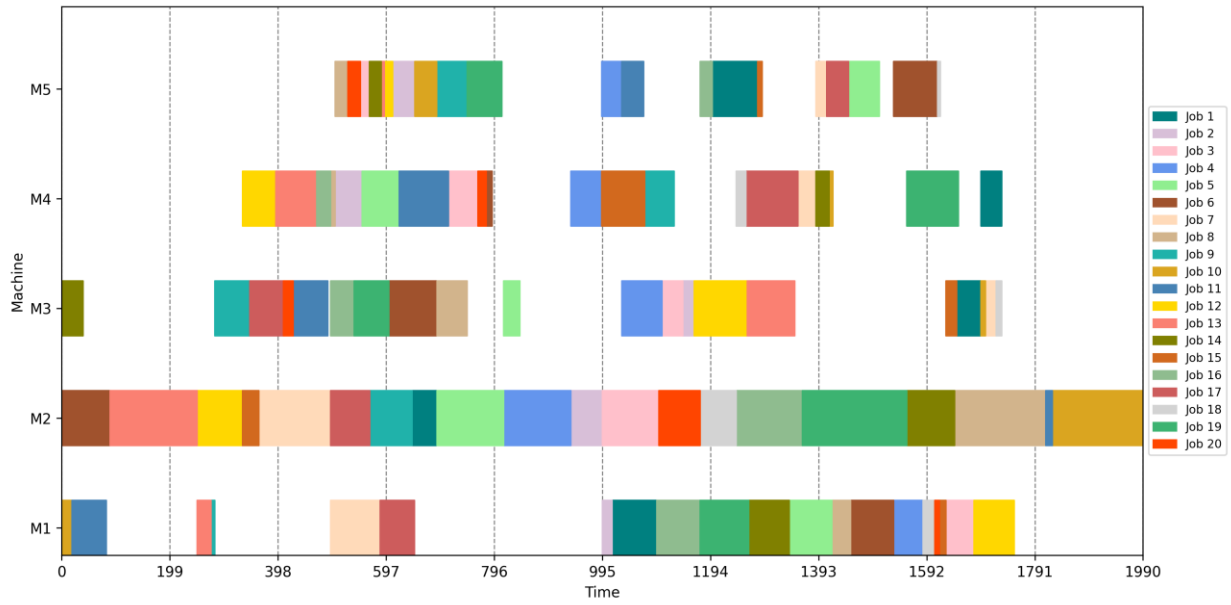

Figure 4.9. Instance la12 second schedule considering sigmoid machine deterioration with optimization.

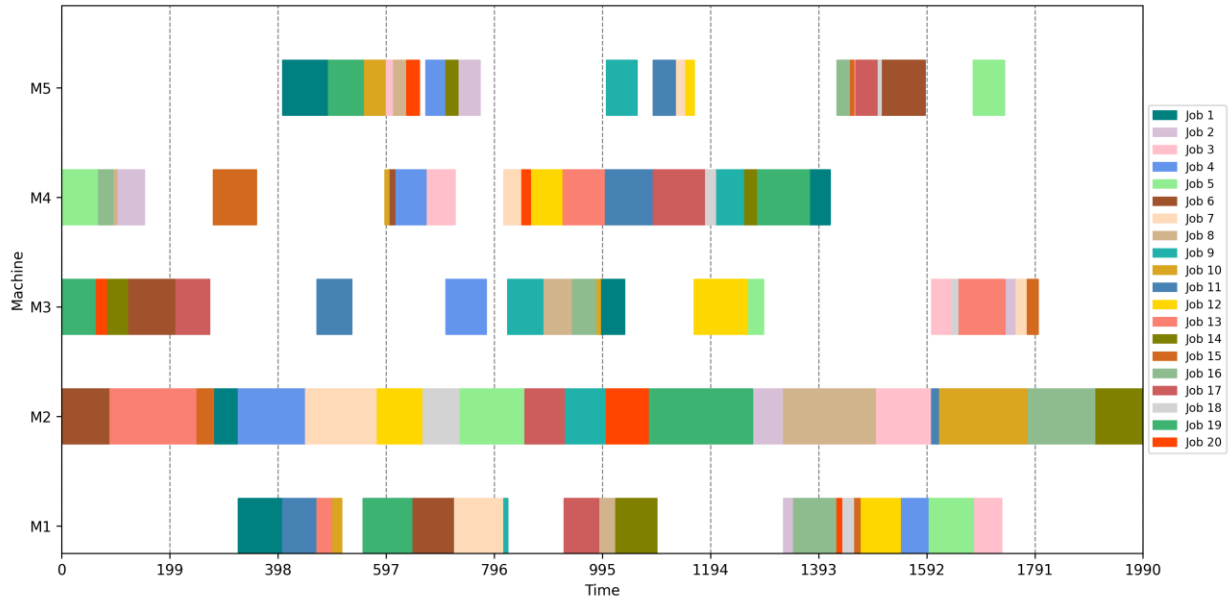

Figure 4.10. Instance la12 third schedule considering sigmoid machine deterioration with optimization.

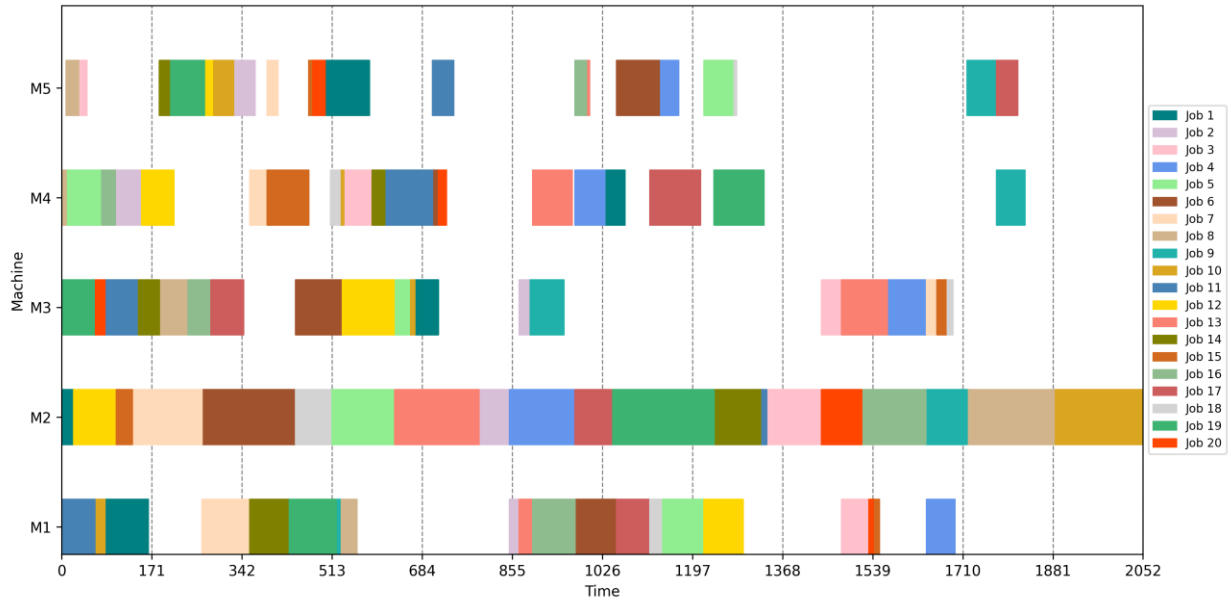

Figure 4.11. Instance la12 schedule considering sigmoid machine deterioration without optimization.

Regarding maintenance, once again the solutions could be improved. The solution for instance la12 (with sigmoidal deterioration) is represented in Figure 4.12. The blank spaces of machine 2 correspond to maintenance time. In this solution, the first operation of machine 2 remained as operation 1 of job 6; however, the subsequent sequence is different, which resulted in a 7.5 % reduction of the makespan. This finding confirms the importance of maintenance in optimizing scheduling problems, highlighting its role in improving operational efficiency.

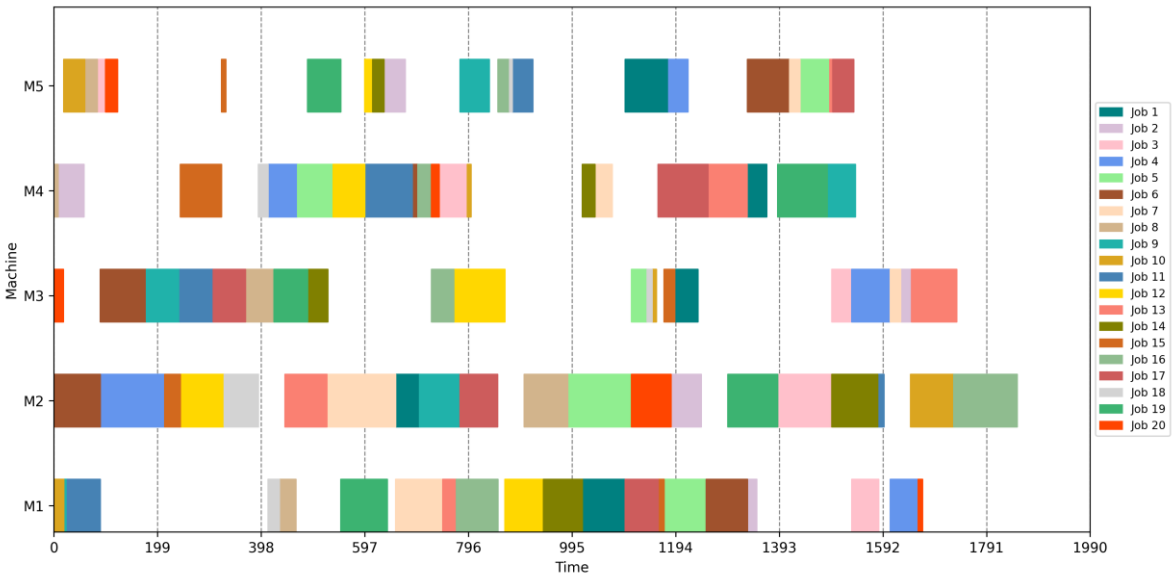

Figure 4.12. Instance la12 schedule considering sigmoid machine deterioration with optimization and maintenance.

The case in which maintenance demonstrated the greatest impact was in the linear case, exhibiting a removal rate of approximately 60 % of the deterioration effects, as easily observed in Figure 4.4. For the other cases, the removal percentage was not as high; the mitigation of the deterioration effects is around 19 % and 35 % for the sigmoid and exponential functions, respectively.

In conclusion, the results prove that optimizing machine deterioration can improve production time across all instances. However, incorporating maintenance into scheduling further reduces production time, which is the ultimate objective. Future research should focus on enhancing the integration of maintenance and exploring alternative decoders to improve the algorithm's efficiency and the quality of the obtained solutions.

5 Conclusion

The conclusions drawn from this dissertation highlight the effectiveness of the proposed BRKGA in solving the JSP. By considering instances from the literature and comparing the results with those obtained by other algorithms, it was clearly shown that the BRKGA performs well in these instances. Implementing the multi-population approach further enhances the algorithm's performance, providing better solutions with smaller optimality gaps and standard deviations. The latter characteristics is very important in practice as it ensures robustness, *i.e.*, that any found solution is never to far from the best found one; hence ensuring its reliability.

Moreover, considering deterioration effects in the scheduling process has proven crucial for achieving realistic and efficient solutions. The optimization approach, which accounts for job and machine deterioration, outperforms the approach that considers deterioration after solving the problem. It leads to significant improvements in the makespan, almost eliminating the negative impact of deterioration on productivity.

For the linear, exponential, and sigmoid job deterioration cases, the optimization approach significantly improves the makespan, reducing it by an average of 4.7 % for linear deterioration, 38.1 % for exponential deterioration, and 17.7 % for sigmoid deterioration. Similarly, the optimization approach effectively reduces the makespan by an average of 22.9 % for linear machine deterioration. These results highlight the importance of addressing deterioration in scheduling problems and emphasize the necessity of realistic modelling by incorporating deterioration effects. Doing so leads to more efficient job assignments and shorter makespans compared to scenarios where deterioration is not considered.

The importance of considering deterioration in industrial processes cannot be overstated. Uncontrolled schedules that do not account for deterioration can result in a significant loss of productivity and increased operating costs as well as worsening customer relations and increased waste disposal. By incorporating deterioration into the scheduling algorithm, jobs can be assigned more efficiently, resulting in shorter makespans compared to scenarios where deterioration is not considered.

Furthermore, maintenance should also be considered in the scheduling optimization process for machines subject to deterioration. The experiments have shown that by doing so, the makespan can be improved, highlighting the importance of scheduling periodic maintenance instead of random or end-of-production maintenance without optimization.

These findings contribute to the body of knowledge in the field of JSP and provide valuable insights for industrial applications. Realistic scheduling that considers deterioration effects can

lead to improved productivity and cost-efficiency as well as improve customer relations, and reduce waste.

In conclusion, this dissertation has demonstrated the effectiveness of the BRKGA and its multi-population variant in solving the JSP. The inclusion of deterioration effects in the optimization process has shown significant improvements in the makespan, emphasizing the importance of considering deterioration in real-world production processes. The findings contribute to the field of scheduling algorithms and provide valuable insights for industries seeking more efficient and robust scheduling solutions.

6 Assessment of the work done

6.1 Objectives Achieved

The main goal of this work was to study the effect of deterioration in a generic job-shop problem and create awareness for the topic within the industry.

In fact, this dissertation has successfully demonstrated that the inclusion of deterioration effects in the optimization process promotes significant improvements in the makespan, emphasizing the importance of considering deterioration in real-world production processes. The findings contribute to the field of scheduling algorithms and provide valuable managerial insights.

6.2 Final Assessment

This dissertation has successfully achieved its goals of investigating and improving scheduling optimization in the presence of deterioration. The results obtained indicate that the implemented approaches outperformed the non-optimized scenarios, demonstrating the significance of considering deterioration effects in scheduling decisions.

However, it is important to note that while the dissertation's objectives were fully accomplished, there is still room for further improvement in predicting and managing machine deterioration. One potential avenue for enhancing the results lies in exploring maintenance and alternative algorithm decoders. Future research should also focus on investigating more complex scenarios, aiming to enhance the understanding and application of deterioration effects in real-life production environments.

7 References

Abedi, Mehdi, Raymond Chiong, Nasimul Noman, and Rui Zhang. 2020. "A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines." *Expert Systems with Applications* 157: 113348. https://doi.org/10.1016/j.eswa.2020.113348.

Azzouz, Ameni, Abir Chaabani, Meriem Ennigrou, and Lamjed Ben Said. 2020. "Handling Sequence-dependent Setup Time Flexible Job Shop Problem with Learning and Deterioration Considerations using Evolutionary Bi-level Optimization." *Applied Artificial Intelligence* 34 (6): 433-455. https://doi.org/10.1080/08839514.2020.1723871.

Azzouz, Ameni, Meriem Ennigrou, and Lamjed Ben Said. 2017. "A self-adaptive hybrid algorithm for solving flexible job-shop problem with sequence dependent setup time." *Procedia Computer Science* 112: 457-466. https://doi.org/10.1016/j.procs.2017.08.023.

Bachman, Aleksander, and Adam Janiak. 2002. "Minimizing the total weighted completion time of deteriorating jobs." *Information Processing Letters* 81 (2): 81-84. https://doi.org/10.1016/S0020-0190(01)00196-X.

Bean, James C. 1994. "Genetic algorithms and random keys for sequencing and optimization." *ORSA journal on computing* 6 (2): 154-160. https://doi.org/10.1287/ijoc.6.2.154.

Bouazza, W., Y. Sallez, and B. Beldjilali. 2017. "A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect." *IFAC-PapersOnLine* 50 (1): 15890-15895. https://doi.org/10.1016/j.ifacol.2017.08.2354.

Boussaïd, Ilhem, Julien Lepagnot, and Patrick Siarry. 2013. "A survey on optimization metaheuristics." *Information Sciences* 237: 82-117. https://doi.org/10.1016/j.ins.2013.02.041.

Browne, Sid, and Uri Yechiali. 1990. "Scheduling Deteriorating Jobs on a Single Processor." *Operations Research* 38 (3): 495-498. https://doi.org/10.1287/opre.38.3.495.

Brucker, Peter, and Sigrid Knust. 2012. "Scheduling Models." In *Complex Scheduling*, edited by Peter Brucker and Sigrid Knust, 1-28. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bustos-Tellez, C. A., J. S. Tenjo-García, and J. C. Figueroa-García. 2018. "Solving job-shop scheduling problems with fuzzy processing times and fuzzy due dates." Communications in Computer and Information Science.

Çaliş, Banu, and Serol Bulkan. 2015. "A research survey: review of AI solution strategies of job shop scheduling problem." *Journal of Intelligent Manufacturing* 26 (5): 961-973. https://doi.org/10.1007/s10845-013-0837-8.

Chaudhry, I. A. 2012. "Job shop scheduling problem with alternative machines using genetic algorithms." *Journal of Central South University of Technology (English Edition)* 19 (5): 1322-1333. https://doi.org/10.1007/s11771-012-1145-8.

Chen, Bo, Chris N. Potts, and Gerhard J. Woeginger. 1998. "A Review of Machine Scheduling: Complexity, Algorithms and Approximability." In *Handbook of Combinatorial Optimization: Volume1-3*, edited by Ding-Zhu Du and Panos M. Pardalos, 1493-1641. Boston, MA: Springer US.

Cunha, V., L. Pessoa, M. Vellasco, R. Tanscheit, and M. A. Pacheco. 2018. "A Biased Random-Key Genetic Algorithm for the Rescue Unit Allocation and Scheduling Problem." 2018 IEEE Congress on Evolutionary Computation (CEC), 8-13 July 2018.

Dimopoulos, C., and A. M. S. Zalzala. 2000. "Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons." *IEEE*

Transactions on Evolutionary Computation 4 (2): 93-113. https://doi.org/10.1109/4235.850651.

Fontes, Dalila B. M. M., and José Fernando Gonçalves. 2013. "A multi-population hybrid biased random key genetic algorithm for hop-constrained trees in nonlinear cost flow networks." *Optimization Letters* 7 (6): 1303-1324. https://doi.org/10.1007/s11590-012-0505-5.

Fontes, Dalila B. M. M., and S. Mahdi Homayouni. 2023a. "A bi-objective multi-population biased random key genetic algorithm for joint scheduling quay cranes and speed adjustable vehicles in container terminals." *Flexible Services and Manufacturing Journal* 35 (1): 241-268. https://doi.org/10.1007/s10696-022-09467-6.

Fontes, Dalila B. M. M., S. Mahdi Homayouni, and José F. Gonçalves. 2023b. "A hybrid particle swarm optimization and simulated annealing algorithm for the job shop scheduling problem with transport resources." *European Journal of Operational Research* 306 (3): 1140-1157. https://doi.org/10.1016/j.ejor.2022.09.006.

Fontes, Dalila B. M. M., S. Mahdi Homayouni, and Mauricio G. C. Resende. 2022. "Job-shop scheduling-joint consideration of production, transport, and storage/retrieval systems." *Journal of Combinatorial Optimization* 44 (2): 1284-1322. https://doi.org/10.1007/s10878-022-00885-8.

Fontes, Dalila B. M. M., Seyed Mahdi Homayouni, and João Chaves Fernandes. 2023c. "Energy-efficient job shop scheduling problem with transport resources considering speed adjustable resources." *International Journal of Production Research*: 1-24. https://doi.org/10.1080/00207543.2023.2175172.

Gao, J., X. Zhu, K. Bai, and R. Zhang. 2022. "New controllable processing time scheduling with subcontracting strategy for no-wait job shop problem." *International Journal of Production Research* 60 (7): 2254-2274. https://doi.org/10.1080/00207543.2021.1886368.

García-Martínez, Carlos, Francisco J. Rodriguez, and Manuel Lozano. 2018. "Genetic Algorithms." In *Handbook of Heuristics*, edited by Rafael Martí, Panos M. Pardalos and Mauricio G. C. Resende, 431-464. Cham: Springer International Publishing.

Gonçalves, José Fernando, Jorge José de Magalhães Mendes, and Maurício G. C. Resende. 2005. "A hybrid genetic algorithm for the job shop scheduling problem." *European Journal of Operational Research* 167 (1): 77-95. https://doi.org/10.1016/j.ejor.2004.03.012.

Gonçalves, José Fernando, and Mauricio G. C. Resende. 2010. "Biased random-key genetic algorithms for combinatorial optimization." *Journal of Heuristics* 17 (5): 487-525. https://doi.org/10.1007/s10732-010-9143-1.

Gonçalves, José Fernando, and Mauricio G. C. Resende. 2014. "An extended Akers graphical method with a biased random-key genetic algorithm for job-shop scheduling." *International Transactions in Operational Research* 21 (2): 215-246. https://doi.org/10.1111/itor.12044.

Gonçalves, José Fernando, and Mauricio G. C. Resende. 2018a. "Biased Random-Key Genetic Programming." In *Handbook of Heuristics*, edited by Rafael Martí, Panos M. Pardalos and Mauricio G. C. Resende, 23-37. Cham: Springer International Publishing.

Gonçalves, José Fernando, and Mauricio G. C. Resende. 2018b. "Random-Key Genetic Algorithms." In *Handbook of Heuristics*, edited by Rafael Martí, Panos M. Pardalos and Mauricio G. C. Resende, 703-715. Cham: Springer International Publishing.

Gupta, Jatinder N. D., and Sushil K. Gupta. 1988. "Single facility scheduling with nonlinear processing times." *Computers & Industrial Engineering* 14 (4): 387-393. https://doi.org/10.1016/0360-8352(88)90041-1.

Homayouni, S. M., D. B. M. M. Fontes, and F. A. C. C. Fontes. 2022. "A Multi-Population BRKGA for Energy-Efficient Job Shop Scheduling with Speed Adjustable Machines." Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).

Homayouni, S. Mahdi, Dalila B. M. M. Fontes, and José F. Gonçalves. 2023. "A multistart biased random key genetic algorithm for the flexible job shop scheduling problem with transportation." *International Transactions in Operational Research* 30 (2): 688-716. https://doi.org/10.1111/itor.12878.

Jiang, Tao, Shaojun Lu, Mingyu Ren, Hao Cheng, and Xinbao Liu. 2023. "Modified benders decomposition and metaheuristics for multi-machine parallel-batch scheduling and resource allocation under deterioration effect." *Computers & Industrial Engineering* 176: 108977. https://doi.org/10.1016/j.cie.2023.108977.

Jiang, Tianhua, Huiqi Zhu, Lu Liu, and Qingtao Gong. 2022. "Energy-conscious flexible job shop scheduling problem considering transportation time and deterioration effect simultaneously." *Sustainable Computing: Informatics and Systems* 35: 100680. https://doi.org/10.1016/j.suscom.2022.100680.

Kong, Min, Xinbao Liu, Jun Pei, Hao Cheng, and Panos M. Pardalos. 2020a. "A BRKGA-DE algorithm for parallel-batching scheduling with deterioration and learning effects on parallel machines under preventive maintenance consideration." *Annals of Mathematics and Artificial Intelligence* 88 (1): 237-267. https://doi.org/10.1007/s10472-018-9602-1.

Kong, Min, Jun Pei, Jin Xu, Xinbao Liu, Xiaoyu Yu, and Panos M. Pardalos. 2020b. "A robust optimization approach for integrated steel production and batch delivery scheduling with uncertain rolling times and deterioration effect." *International Journal of Production Research* 58 (17): 5132-5154. https://doi.org/10.1080/00207543.2019.1693659.

Korovessi, Ekaterini, and Andreas A Linninger. 2006. *Batch processes*. CRC/Taylor & Francis Boca Raton, FL.

Kunnathur, Anand S., and Sushil K. Gupta. 1990. "Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem." *European Journal of Operational Research* 47 (1): 56-64. https://doi.org/10.1016/0377-2217(90)90089-T.

Lawrence, Stephen. 1984. *Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement)*.

Li, J., and H. Guo. 2021. "A Hybrid Whale Optimization Algorithm for Plane Block Parallel Blocking Flowline Scheduling Optimization With Deterioration Effect in Lean Shipbuilding." *IEEE Access* 9: 131893-131905. https://doi.org/10.1109/ACCESS.2021.3112742.

Li, Yufeng, Yan He, Yulin Wang, Fei Tao, and John W. Sutherland. 2020. "An optimization method for energy-conscious production in flexible machining job shops with dynamic job arrivals and machine breakdowns." *Journal of Cleaner Production* 254: 120009. https://doi.org/10.1016/j.jclepro.2020.120009.

Lin, S. W., and K. C. Ying. 2021. "Minimising makespan in job-shops with deterministic machine availability constraints." *International Journal of Production Research* 59 (14): 4403-4415. https://doi.org/10.1080/00207543.2020.1764125.

Lopes, Mauro Cardoso, Carlos Eduardo de Andrade, Thiago Alves de Queiroz, Mauricio G. C. Resende, and Flávio Keidi Miyazawa. 2016. "Heuristics for a hub location-routing problem." *Networks* 68 (1): 54-90. https://doi.org/10.1002/net.21685.

Luo, Qiang, Qianwang Deng, Guiliang Gong, Xin Guo, and Xiahui Liu. 2022. "A distributed flexible job shop scheduling problem considering worker arrangement using an improved

memetic algorithm." *Expert Systems with Applications* 207: 117984. https://doi.org/10.1016/j.eswa.2022.117984.

Ma, Chunfeng, Min Kong, Jun Pei, and Panos M. Pardalos. 2018. "BRKGA-VNS for Parallel-Batching Scheduling on a Single Machine with Step-Deteriorating Jobs and Release Times." Machine Learning, Optimization, and Big Data, Cham, 2018.

Mencía, Raúl, María R. Sierra, Carlos Mencía, and Ramiro Varela. 2015. "Memetic algorithms for the job shop scheduling problem with operators." *Applied Soft Computing* 34: 94-105. https://doi.org/10.1016/j.asoc.2015.05.004.

Mokeddem, D., and A. Khellaf. 2009. "Optimal solutions of multiproduct batch chemical process using multiobjective genetic algorithm with expert decision system." *Journal of Automated Methods and Management in Chemistry* 2009. https://doi.org/10.1155/2009/927426.

Moon, Ilkyeong, Sanghyup Lee, and Hyerim Bae. 2008. "Genetic algorithms for job shop scheduling problems with alternative routings." *International Journal of Production Research* 46 (10): 2695-2705. https://doi.org/10.1080/00207540701244820.

Mosheiov, Gur. 2002. "Complexity analysis of job-shop scheduling with deteriorating jobs." *Discrete Applied Mathematics* 117 (1): 195-209. https://doi.org/10.1016/S0166-218X(00)00385-1.

Oliveira, José António, Luís M. S. Dias, and Guilherme A. B. Pereira. 2010. "Solving the Job Shop Problem with a random keys genetic algorithm with instance parameters." 2nd International Conference on Engineering Optimization, Lisbon.

Pei, Jun, Ya Zhou, Ping Yan, and Panos M. Pardalos. 2022. "A concise guide to scheduling with learning and deteriorating effects." *International Journal of Production Research*: 1-22. https://doi.org/10.1080/00207543.2022.2049911.

Pinedo, Michael L. 2012. *Scheduling: Theory, Algorithms, and Systems*. Edited by Michael L. Pinedo. Boston, MA: Springer US.

Renna, Paolo. 2015. "Deteriorating job scheduling problem in a job-shop manufacturing system by multi-agent system." *International Journal of Computer Integrated Manufacturing* 28 (9): 936-945. https://doi.org/10.1080/0951192X.2014.928747.

Rizzi, Mateus Matos, Antonio Augusto Chaves, Edson Luiz França Senne, and Luiz Antonio Nogueira Lorena. 2015. "Metaheurística híbrida aplicada ao problema de job shop." Simpósio Brasileiro de Pesquisa Operacional, Porto de Galinhas, Pernambuco, 08/2015. https://www.din.uem.br/~ademir/sbpo/sbpo2015/pdf/142838.pdf.

Shabtay, Dvir, and Baruch Mor. 2022. "Exact algorithms and approximation schemes for proportionate flow shop scheduling with step-deteriorating processing times." *Journal of Scheduling*. https://doi.org/10.1007/s10951-022-00766-2.

Tamssaouet, Karim, Stéphane Dauzère-Pérès, and Claude Yugma. 2018. "Metaheuristics for the job-shop scheduling problem with machine availability constraints." *Computers & Industrial Engineering* 125: 1-8. https://doi.org/10.1016/j.cie.2018.08.008.

Tao, Ning, Duan Xiaodong, An Lu, and Gou Tao. 2021. "Research on disruption management of urgent arrival in job shop with deteriorating effect." *Journal of Intelligent & Fuzzy Systems* 41: 1247-1259. https://doi.org/10.3233/JIFS-210166.

Teunou, E., and J. J. Fitzpatrick. 1999. "Effect of relative humidity and temperature on food powder flowability." *Journal of Food Engineering* 42 (2): 109-116. https://doi.org/10.1016/S0260-8774(99)00087-4.

Wang, Ting, Roberto Baldacci, Andrew Lim, and Qian Hu. 2018. "A branch-and-price algorithm for scheduling of deteriorating jobs and flexible periodic maintenance on a single

machine." *European Journal of Operational Research* 271 (3): 826-838. https://doi.org/10.1016/j.ejor.2018.05.050.

Won, Sujin, Hyerim Bae, and Riska Asriana Sutrisnowati. 2018. "Production scheduling in steel manufacturing with cutting and parallel operations." *ICIC express letters. Part B, Applications: an international journal of research and surveys* 9 (8): 869-877. https://doi.org/10.24507/icicelb.09.08.869.

Wu, Ouyang, Ala E. F. Bouaswaig, Stefan M. Schneider, Fernando Moreno Leira, Lars Imsland, and Matthias Roth. 2018. "Data-driven degradation model for batch processes: a case study on heat exchanger fouling." In *Computer Aided Chemical Engineering*, edited by Anton Friedl, Jiří J. Klemeš, Stefan Radl, Petar S. Varbanov and Thomas Wallek, 139-144. Elsevier.

Wu, Xiuli, Jing Li, Xianli Shen, and Ning Zhao. 2020. "NSGA-III for solving dynamic flexible job shop scheduling problem considering deterioration effect." *IET Collaborative Intelligent Manufacturing* 2 (1): 22-33. https://doi.org/10.1049/iet-cim.2019.0056.

Wu, Xiuli, Xianli Shen, and Congbo Li. 2019. "The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously." *Computers & Industrial Engineering* 135: 1004-1024. https://doi.org/10.1016/j.cie.2019.06.048.

Xiong, Hegen, Shuangyuan Shi, Danni Ren, and Jinjin Hu. 2022. "A survey of job shop scheduling problem: The types and models." *Computers & Operations Research* 142: 105731. https://doi.org/10.1016/j.cor.2022.105731.

Zhang, Rui, and Cheng Wu. 2011. "An Artificial Bee Colony Algorithm for the Job Shop Scheduling Problem with Random Processing Times." *Entropy* 13 (9): 1708-1729. https://doi.org/10.3390/e13091708.

Annex A - Instance configurations and job deterioration rates

Table A.1. La04 instance settings and job deterioration rate.

la04	Operation				
Job	1	2	3	4	5
J1	M1 (12)	M3 (94)	M4 (92)	M5 (91; 0.151)	M2 (7)
J2	M2 (19)	M4 (11)	M5 (66)	M3 (21)	M1 (87)
J3	M2 (14)	M1 (75)	M4 (13; 0.093)	M5 (16)	M3 (20)
J4	M3 (95)	M5 (66; 0.198)	M1 (7)	M4 (7)	M2 (77; 0.085)
J5	M2 (45)	M4 (6)	M5 (89)	M1 (15)	M3 (34)
J6	M4 (77)	M3 (20)	M1 (76)	M5 (88)	M2 (53)
J7	M3 (74; 0.097)	M2 (88; 0.108)	M1 (52)	M4 (27)	M5 (9;0.016)
J8	M2 (88)	M4 (69; 0.019)	M1 (62; 0.088)	M5 (98)	M3 (52)
J9	M3 (61; 0.143)	M5 (9)	M1 (62)	M2 (52; 0.195)	M4 (90; 0.193)
J10	M3 (54)	M5 (5)	M4 (59)	M2 (15)	M1 (88)

Table A.2. La08 instance settings and job deterioration rate.

la08	Operation				
Job	1	2	3	4	5
J1	M4 (92; 0.186)	M3 (94)	M1 (12)	M5 (91; 0.146)	M2 (7)
J2	M3 (21)	M2 (19)	M1 (87; 0.176)	M4 (11)	M5 (66)
J3	M2 (14)	M4 (13; 0.023)	M1 (75; 0.052)	M5 (16)	M3 (20)
J4	M3 (95)	M5 (66; 0.125)	M1 (7)	M2 (77)	M4 (7)
J5	M3 (34)	M5 (89)	M4 (6)	M2 (45)	M1 (15)
J6	M5 (88)	M4 (77; 0.044)	M3 (20)	M2 (53)	M1 (76)
J7	M5 (9)	M4 (27)	M1 (52)	M2 (88)	M3 (74; 0.105)
J8	M4 (69; 0.092)	M3 (52)	M1 (62)	M2 (88)	M5 (98)
J9	M4 (90; 0.183)	M1 (62)	M5 (9)	M3 (61)	M2 (52)
J10	M5 (5)	M3 (54)	M4 (59)	M1 (88)	M2 (15)
J11	M1 (41)	M2 (50)	M5 (78; 0.137)	M4 (53)	M3 (23)
J12	M1 (38)	M5 (72)	M3 (91; 0.146)	M4 (68)	M2 (71; 0.115)
J13	M1 (45; 0.086)	M4 (95; 0.011)	M5 (52)	M3 (25; 0.158)	M2 (6)
J14	M4 (30)	M2 (66)	M1 (23)	M5 (36)	M3 (17; 0.054)
J15	M3 (95)	M1 (71)	M4 (76; 0.136)	M2 (8)	M5 (88)

Table A.3. La12 instance settings and job deterioration rate.

la12	Operation				
Job	1	2	3	4	5
J1	M2 (23)	M1 (82)	M5 (84; 0.108)	M3 (45; 0.120)	M4 (38)
J2	M4 (50; 0.091)	M5 (41)	M2 (29)	M1 (18)	M3 (21)
J3	M5 (16)	M4 (54)	M2 (52)	M3 (38; 0.051)	M1 (52)
J4	M2 (62)	M4 (57)	M5 (37)	M3 (74)	M1 (54)
J5	M4 (68)	M2 (61)	M3 (30)	M1 (81)	M5 (57)
J6	M2 (89; 0.059)	M3 (89; 0.177)	M4 (11)	M1 (79)	M5 (81; 0.100)
J7	M2 (66)	M1 (91)	M4 (33)	M5 (20; 0.036)	M3 (20)
J8	M4 (8; 0.069)	M5 (24)	M3 (55)	M1 (32)	M2 (84)
J9	M1 (7)	M3 (64)	M2 (39)	M5 (56)	M4 (54)
J10	M1 (19; 0.152)	M5 (40)	M4 (7)	M3 (8; 0.056)	M2 (83)
J11	M1 (63)	M3 (64)	M4 (91)	M5 (40)	M2 (6)
J12	M2 (42; 0.190)	M4 (61)	M5 (15)	M3 (98)	M1 (74)
J13	M2 (80)	M1 (26)	M4 (75)	M5 (6)	M3 (87)
J14	M3 (39)	M5 (22; 0.005)	M1 (75)	M4 (24)	M2 (44)
J15	M2 (15)	M4 (79)	M5 (8)	M1 (12; 0.048)	M3 (20)
J16	M4 (26)	M3 (43; 0.134)	M1 (80)	M5 (22)	M2 (61)
J17	M3 (62)	M2 (36; 0.173)	M1 (63)	M4 (96; 0.041)	M5 (40)
J18	M2 (33)	M4 (18)	M1 (22; 0.103)	M5(5)	M3 (10)
J19	M3 (64; 0.056)	M5 (64; 0.11)	M1 (89; 0.096)	M2 (96; 0.196)	M4 (95; 0.197)
J20	M3 (18)	M5 (23)	M4 (15)	M2 (38; 0.196)	M1 (8; 0.170)

Table A.4. La16 instance settings and job deterioration rate.

la16	Operation									
Job	1	2	3	4	5	6	7	8	9	10
J1	M2 (21)	M7 (71; 0.090)	M10 (16; 0.083)	M9 (52)	M8 (26)	M3 (34)	M1 (53)	M5 (21)	M4 (55)	M6 (95)
J2	M5 (55)	M3 (31; 0.137)	M6 (98)	M10 (79; 0.113)	M1 (12)	M8 (66; 0.126)	M2 (42)	M9 (77)	M7 (77)	M4 (39; 0.014)
J3	M4 (34; 0.086)	M3 (64)	M9 (62)	M2 (19)	M5 (92)	M10 (79; 0.186)	M8(43)	M7(54)	M1 (83; 0.073)	M6(37)
J4	M2 (87)	M4 (69)	M3 (87)	M8 (38)	M9 (24)	M10 (83)	M7 (41)	M1 (93)	M6 (77; 0.052)	M5 (60)
J5	M3 (98)	M1 (44)	M6 (25; 0.156)	M7 (75)	M8 (43)	M2 (49)	M5 (96)	M10 (77)	M4 (17)	M9 (79; 0.053)
J6	M3 (35)	M4 (76)	M6 (28)	M10 (10)	M5 (61; 109)	M7 (9; 0.072)	M1 (95)	M9 (35)	M2 (7; 0.034)	M8 (95)
J7	M4 (16)	M3 (59)	M1 (46)	M2 (91)	M10 (43)	M9 (50)	M7 (52; 0.033)	M6 (59)	M5 (28; 0.067)	M8 (27)
J8	M2 (45)	M1 (87)	M4(41)	M5 (20)	M7 (54)	M10 (43; 0.037)	M9 (14)	M6 (9)	M3 (39)	M8(71)
J9	M5 (33)	M3 (37; 0.105)	M9 (66)	M6 (33)	M4 (26)	M8 (8; 0.066)	M2 (28; 0.120)	M7 (89; 0.009)	M10 (42; 0.135)	M1 (78)
J10	M9 (69)	M10 (81)	M3 (94; 0.047)	M5 (96)	M4 (27)	M1 (69; 0.196)	M8 (45)	M7 (78)	M2 (74)	M6 (84)

Table A.5. La17 instance settings and job deterioration rate.

la17	Operation									
Job	1	2	3	4	5	6	7	8	9	10
J1	M5 (18; 0.024)	M8 (21; 0.151)	M10 (41; 0.151)	M3 (45)	M4 (38)	M9 (50)	M6 (84)	M7 (29; 0.106)	M2 (23; 0.178)	M1 (82)
J2	M9 (57)	M6 (16; 0.139)	M2 (52)	M8 (74; 0.169)	M3 (38; 0.061)	M4 (54)	M7 (62)	M10 (37)	M5 (54)	M1 (52)
J3	M3 (30)	M5 (79; 0.189)	M4 (68)	M2 (61)	M9 (11; 0.143)	M7 (89)	M8 (89)	M1 (81)	M10(81; 0.018)	M6 (57)
J4	M1 (91)	M9 (8; 0.145)	M4 (33; 0.126)	M8 (55)	M6 (20)	M3 (20)	M5 (32)	M7 (84)	M2 (66; 0.172)	M10 (24)
J5	M10 (40)	M1 (7)	M5 (19)	M9 (7)	M7 (83)	M3 (64)	M6 (56)	M4 (54)	M8 (8)	M2 (39; 0.180)
J6	M4 (91)	M3 (64)	M6 (40; 0.132)	M1 (63)	M8 (98)	M5 (74)	M9 (61)	M2 (6)	M7 (42)	M10 (15; 0.090)
J7	M2 (80; 0.061)	M8 (39)	M9 (24)	M4 (75)	M5 (75)	M6 (6)	M7 (44; 0.056)	M1 (26; 0.043)	M3 (87)	M10 (22)
J8	M2 (15)	M8 (43)	M3 (20)	M1 (12)	M9 (26)	M7 (61)	M4 (79)	M10 (22)	M6 (8)	M5 (80)
J9	M3 (62; 0.166)	M4 (96)	M5 (22)	M10 (5)	M1 (63)	M7 (33; 0.158)	M8 (10)	M9 (18; 0.008)	M2 (36; 0.168)	M6 (40)
J10	M2 (96)	M1 (89)	M6 (64)	M4 (95)	M10 (23)	M8 (18)	M9 (15)	M3 (64)	M7 (38; 0.035)	M5 (8)

Table A.6. La22 instance settings and job deterioration rate.

la22	Operation									
Job	1	2	3	4	5	6	7	8	9	10
J1	M10 (66; 0.108)	M6 (91)	M5 (87)	M3 (94)	M8 (21; 0.083)	M4 (92; 0.017)	M2 (7)	M1 (12)	M9 (11)	M7 (19)
J2	M4 (13)	M3 (20; 0.169)	M5 (7)	M2 (14)	M10 (66; 0.152)	M1 (75; 0.162)	M7 (77)	M6 (16; 0.175)	M8 (95)	M9 (7)
J3	M9 (77; 0.076)	M8 (20)	M3 (34; 0.137)	M1 (15)	M10 (88)	M6 (89; 0.083)	M7 (53)	M4 (6)	M2 (45)	M5 (76; 0.106)
J4	M4 (27)	M3 (74)	M7 (88)	M5 (62)	M8 (52)	M9 (69; 0.168)	M6 (9)	M10 (98)	M1 (52)	M2 (88)
J5	M5 (88)	M7 (15; 0.088)	M2 (52)	M3 (61)	M8 (54)	M1 (62)	M9 (59)	M6 (9)	M4 (90; 0.141)	M10 (5)
J6	M7 (71)	M1 (41; 0.046)	M5 (38)	M4 (53)	M8 (91)	M9 (68)	M2 (50)	M6 (78; 0.029)	M3 (23; 0.197)	M10 (72)
J7	M4 (95)	M10 (36)	M7 (66)	M6 (52)	M1 (45; 0.050)	M9 (30)	M5 (23)	M3 (25)	M8 (17)	M2 (6; 0.056)
J8	M5 (65; 0.111)	M2 (8; 0.117)	M9 (85; 0.199)	M1 (71)	M8 (65)	M7 (28; 0.122)	M6 (88)	M4 (76; 0.120)	M10 (27)	M3 (95)
J9	M10 (37)	M2 (37)	M5 (28)	M4 (51)	M9 (86)	M3 (9)	M7 (55)	M1 (73)	M8 (51)	M6 (90)
J10	M4 (39)	M3 (15)	M7 (83; 0.039)	M10 (44)	M8 (53)	M1 (16)	M5 (46; 0.149)	M6 (24)	M2 (25)	M9 (82)
J11	M2 (72; 0.151)	M5 (48)	M1 (87)	M3 (66)	M10 (5)	M7 (54)	M8 (39; 0.037)	M9 (35)	M6 (95; 0.145)	M4 (60)
J12	M2 (46)	M4 (20)	M1 (97)	M3 (21)	M10 (46)	M8 (37; 0.023)	M9 (19)	M5 (59)	M7 (34)	M6 (55)
J13	M6 (23; 0.118)	M4 (25)	M7 (78)	M2 (24; 0.195)	M1 (28)	M8 (83)	M9 (28; 0.184)	M10 (5)	M3 (73)	M5 (45; 0.148)
J14	M2 (37)	M1 (53; 0.064)	M8 (87)	M5 (38)	M4 (71)	M6 (29)	M10 (12)	M9 (33)	M7 (55)	M3 (12)
J15	M5 (90)	M9 (17)	M3 (49)	M4 (83)	M2 (40)	M7 (23; 0.117)	M8 (65; 0.126)	M10 (27)	M6 (7)	M1 (48)

Table A.7. La23 instance settings and job deterioration rate.

la23	Operation									
Job	1	2	3	4	5	6	7	8	9	10
J1	M8 (84)	M6 (58; 0.048)	M9 (77)	M3 (44)	M5 (97)	M7 (89)	M4 (5; 0.012)	M2 (58)	M10 (96)	M1 (9; 0.128)
J2	M7 (21)	M2 (87; 0.049)	M5 (15)	M6 (39)	M3 (81)	M4 (85; 0.024)	M8 (31)	M9 (57)	M10 (73)	M1 (77)
J3	M1 (40)	M6 (71)	M9 (34)	M10 (82)	M4 (70)	M7 (22)	M5 (10)	M8 (80)	M3 (48)	M2 (49)
J4	M6 (75)	M3 (17)	M4 (7)	M7 (72)	M5 (11)	M8 (62)	M9 (47)	M10 (35; 0.081)	M2 (91)	M1 (55)
J5	M10 (20)	M5 (12)	M7 (71)	M8 (67)	M1 (64)	M3 (94)	M9 (15)	M6 (50)	M4 (75; 0.073)	M2 (90)
J6	M7 (93)	M6 (93)	M2 (57)	M8 (70; 0.036)	M9 (77)	M5 (58; 0.127)	M1 (52)	M3 (29)	M10 (7; 0.027)	M4 (68)
J7	M8 (56)	M1 (95; 0.038)	M9 (48)	M5 (26)	M3 (82)	M2 (63)	M10 (36)	M4 (27)	M7 (87)	M6 (6; 0.139)
J8	M4 (76)	M6 (15)	M10 (78)	M2 (8; 0.095)	M9 (41; 0.060)	M3 (36; 0.099)	M5 (30)	M7 (84)	M1 (36)	M8 (76)
J9	M1 (75)	M8 (13; 0.040)	M3 (81; 0.178)	M9 (29)	M5 (54)	M7 (82; 0.035)	M6 (88)	M2 (78)	M10 (40)	M4 (13)
J10	M3 (6)	M2 (26)	M8 (32)	M7 (64)	M5 (54)	M1 (52; 0.014)	M6 (82)	M4 (6)	M10 (88)	M9 (54; 0.148)
J11	M9 (62; 0.132)	M3 (67)	M6 (32)	M1 (62; 0.162)	M8 (69)	M4 (61)	M2(35; 0.047)	M5 (72)	M10 (5; 0.164)	M7 (93; 0.132)
J12	M3 (78)	M10 (90)	M1 (85; 0.179)	M2 (72)	M9 (64; 0.077)	M7 (63)	M4 (11)	M8 (82)	M6 (88; 0.117)	M5 (7)
J13	M5 (28)	M10 (11)	M8 (50)	M7 (88)	M1 (44)	M6 (31)	M3 (27)	M2 (66; 0.070)	M9 (49)	M4 (35; 0.124)
J14	M3 (14; 0.120)	M6 (39)	M7 (56)	M5 (62)	M4 (97; 0.076)	M10 (66)	M8 (69)	M 2(7; 0.186)	M9 (47)	M1 (76, 0.113)
J15	M2 (18)	M9 (93)	M8 (58)	M7 (47)	M4 (69; 0.081)	M10 (57)	M3 (41; 0.110)	M6 (53; 0.049)	M5 (79)	M1 (64)

Table A.8. La28 instance settings and job deterioration rate.

la28	Operation									
Job	1	2	3	4	5	6	7	8	9	10
J1	M9 (32; 0.137)	M2 (81)	M5 (55)	M8 (40)	M1 (6)	M6 (19)	M10 (81)	M4 (37)	M3 (40; 0.107)	M7 (9)
J2	M3 (70)	M4 (55)	M8 (21; 0.178)	M5 (64)	M2 (46)	M9 (25)	M10 (65)	M1 (77)	M6 (65)	M7 (15)
J3	M8 (84)	M5 (89)	M4 (24)	M2 (44)	M3 (85)	M9 (31)	M10 (29)	M7 (83; 0.157)	M6 (37)	M1 (40; 0.190)
J4	M5 (80)	M6 (59)	M1 (8)	M3 (30)	M7 (77)	M4 (38)	M2 (80; 0.131)	M8 (56)	M10 (41)	M9 (97; 0.058)
J5	M7 (40)	M3 (71)	M1 (91)	M8 (7; 0.197)	M10 (59; 0.046)	M9 (80)	M4 (50; 0.030)	M6 (56)	M2 (17)	M5 (88; 0.078)
J6	M8 (36)	M10 (10; 0.121)	M1 (45)	M7 (9; 0.016)	M5 (54)	M9 (96)	M3 (8; 0.129)	M6 (77; 0.118)	M2 (29)	M4 (58; 0.156)
J7	M7 (99; 0.128)	M9 (86)	M4 (92)	M1 (28; 0.056)	M2 (98)	M5 (70)	M6 (87)	M10 (96)	M3 (73)	M8 (27)
J8	M2 (95)	M4 (85)	M6 (56; 0.105)	M5 (52)	M1 (59)	M3 (41)	M7 (81)	M9 (39)	M10 (32)	M8 (92)
J9	M2 (7)	M8 (69; 0.164)	M5 (93)	M7 (27)	M6 (22)	M1 (88)	M9 (45)	M4 (60)	M10 (49)	M3 (12)
J10	M8 (33)	M3 (61)	M9 (44; 0.027)	M6 (26; 0.048)	M2 (84)	M7 (82; 0.029)	M4 (68)	M1 (21)	M10 (71)	M5 (99; 0.136)
J11	M9 (43)	M1 (72)	M5 (30; 0.056)	M6 (98)	M10 (75)	M2 (26)	M8 (8)	M7 (74; 0.134)	M4 (19)	M3 (38; 0.183)
J12	M7 (19)	M3 (67)	M9 (73)	M2 (85; 0.191)	M10 (26)	M5 (39)	M8 (9)	M1 (23)	M6 (13)	M4 (43)
J13	M9 (72)	M8 (46)	M6 (80; 0.073)	M4 (93)	M3 (61)	M5 (7; 0.032)	M10 (42)	M2 (50; 0.199)	M1 (55)	M7 (57)
J14	M5 (99)	M1 (91)	M10 (11)	M6 (68; 0.088)	M8 (43)	M4 (96; 0.066)	M3 (72)	M9 (11; 0.091)	M7 (60)	M2 (68)
J15	M10 (69)	M1 (43; 0.193)	M4 (12)	M9 (40)	M2 (70)	M7 (74)	M3 (34)	M6 (7)	M5 (30)	M8 (84; 0.162)
J16	M8 (99)	M4 (27)	M5 (59)	M6 (72)	M3 (9)	M7 (45)	M1 (49)	M10 (63; 0.102)	M2 (69)	M9 (60)
J17	M1 (75)	M4 (17)	M3 (91)	M8 (50; 0.185)	M9 (65; 0.020)	M6 (37)	M10 (98; 0.079)	M2 (90)	M7 (71)	M5 (8; 0.167)
J18	M10 (72; 0.142)	M2 (9)	M4 (31)	M7 (49; 0.180)	M3 (91)	M9 (62)	M8 (90)	M1 (72)	M6 (98)	M5 (38; 0.062)
J19	M5 (35)	M3 (63)	M6 (25)	M7 (35; 0.108)	M9 (21)	M8 (47)	M4 (52)	M2 (80)	M1 (39)	M10 (74)
J20	M3 (68)	M6 (24)	M10 (58; 0.147)	M9 (52; 0.181)	M1 (5; 0.041)	M7 (20)	M4 (50)	M8 (57; 0.078)	M2 (88; 0.089)	M5 (53)

Table A.9. La31 instance settings and job deterioration rate.

la31	Operation									
Job	1	2	3	4	5	6	7	8	9	10
J1	M5 (21)	M8 (26)	M10 (16)	M3 (34)	M4 (55)	M9 (52; 0.030)	M6 (95; 0.193)	M7 (71; 0.085)	M2 (21; 0.015)	M1 (53; 0.111)
J2	M9 (77; 0.128)	M6 (98; 0.013)	M2 (42)	M8 (66)	M3 (31; 0.159)	M4 (39)	M7 (77)	M10 (79)	M5 (55; 0.057)	M1 (12)
J3	M3 (64)	M5 (92; 0.133)	M4 (34)	M2 (19; 0.049)	M9 (62)	M7 (54)	M8 (43)	M1 (83)	M10 (79)	M6 (37)
J4	M1 (93)	M9 (24)	M4 (69)	M8 (38; 0.066)	M6 (77)	M3 (87)	M5 (60)	M7 (41)	M2 (87)	M10 (83)
J5	M10 (77)	M1 (44; 0.060)	M5 (96; 0.017)	M9 (79; 0.102)	M7 (75)	M3 (98; 0.053)	M6 (25; 0.133)	M4 (17)	M8 (43; 0.072)	M2 (49)
J6	M4 (76)	M3 (35)	M6 (28)	M1 (95)	M8 (95)	M5 (61)	M9 (35)	M2 (7)	M7 (9)	M10(10)
J7	M2 (91)	M8 (27; 0.071)	M9 (50; 0.022)	M4 (16)	M5 (28)	M6 (59)	M7 (52)	M1 (46)	M3 (59)	M10 (43)
J8	M2 (45)	M8 (71)	M3 (39)	M1 (87)	M9 (14)	M7 (54)	M4 (41)	M10 (43)	M6 (9)	M5 (20; 0.037)
J9	M3 (37)	M4 (26)	M5 (33)	M10 (42)	M1 (78)	M7 (89)	M8 (8; 0.007)	M9 (66)	M2 (28; 0.069)	M6 (33)
J10	M2 (74)	M1 (69)	M6 (84)	M4 (27)	M10(81; 0.014)	M8 (45)	M9 (69)	M3 (94)	M7 (78)	M5 (96)
J11	M6 (76)	M8 (32)	M7 (18)	M1 (20)	M4 (87)	M3 (17)	M10(25; 0.134)	M5 (24)	M2 (31; 0.028)	M9 (81)
J12	M10(97; 0.086)	M9 (90)	M6 (28)	M8 (86)	M1 (58)	M2 (72; 0.080)	M3 (23)	M7 (76; 0.035)	M4 (99)	M5 (45; 0.072)
J13	M10 (48)	M6 (27; 0.056)	M7 (67)	M8 (62)	M5 (98)	M1 (42)	M2 (46)	M9 (27; 0.197)	M4 (48)	M3 (17)
J14	M10 (80)	M4 (19)	M6 (28; 0.174)	M2 (12)	M5 (94)	M7 (63; 0.042)	M8 (98)	M9 (50)	M1 (80; 0.138)	M3 (50; 0.027)
J15	M3 (50)	M2 (41; 0.103)	M5 (61)	M9 (79; 0.061)	M6 (14)	M10 (72)	M8 (18; 0.053)	M4 (55)	M7 (37; 0.032)	M1 (75)
J16	M10 (22)	M6 (57; 0.118)	M5 (75)	M3 (14)	M8 (65; 0.035)	M4 (96; 0.150)	M2 (71)	M1 (47; 0.160)	M9 (79)	M7 (60)
J17	M4 (32)	M3 (69)	M5 (44)	M2 (31; 0.148)	M10 (51)	M1 (33)	M7 (34)	M6 (58)	M8 (47)	M9 (58)
J18	M9 (66)	M8 (40)	M3 (17)	M1 (62)	M10(38; 0.017)	M6 (8)	M7 (15; 0.033)	M4 (29)	M2 (44)	M5 (97)
J19	M4 (50; 0.045)	M3 (58)	M7 (21)	M5 (63)	M8 (57)	M9 (32; 0.051)	M6 (20)	M10(87; 0.094)	M1 (57)	M2 (39; 0.051)
J20	M5 (20; 0.084)	M7 (67)	M2 (85; 0.043)	M3 (90; 0.056)	M8 (70)	M1 (84)	M9 (30)	M6 (56)	M4 (61; 0.006)	M10 (15)
J21	M7 (29; 0.073)	M1 (82)	M5 (18; 0.158)	M4 (38)	M8 (21; 0.186)	M9 (50)	M2 (23)	M6 (84)	M3 (45)	M10 (41)
J22	M4 (54)	M10 (37)	M7 (62)	M6 (16)	M1 (52)	M9 (57)	M5 (54; 0.020)	M3 (38; 0.179)	M8 (74)	M2 (52)
J23	M5 (79)	M2 (61)	M9 (11)	M1 (81)	M8 (89; 0.135)	M7 (89)	M6 (57)	M4 (68)	M10(81; 0.080)	M3 (30)
J24	M10(24; 0.104)	M2 (66; 0.079)	M5 (32; 0.126)	M4 (33)	M9 (8)	M3 (20)	M7 (84; 0.104)	M1 (91)	M8 (55; 0.182)	M6 (20)
J25	M4 (54)	M3 (64)	M7 (83)	M10 (40)	M8 (8; 0.128)	M1 (7; 0.011)	M5 (19)	M6 (56)	M2 (39)	M9 (7)
J26	M2 (6)	M5 (74)	M1 (63)	M3 (64)	M10 15)	M7 (42)	M8 (98)	M9 (61)	M6 (40)	M4 (91)
J27	M2 (80)	M4 (75)	M1 (26)	M3 (87)	M10 (22)	M8 (39)	M9 (24)	M5 (75)	M7 (44; 0.079)	M6 (6; 0.171)
J28	M6 (8)	M4 (79)	M7 (61; 0.016)	M2 (15)	M1 (12; 0.015)	M8 (43)	M9 (26)	M10 (22)	M3 (20)	M5 (80)
J29	M2 (36)	M1 (63)	M8 (10)	M5 (22)	M4 (96)	M6 (40)	M10 (5)	M9 (18; 0.080)	M7 (33)	M3 (62)
J30	M5 (8)	M9 (15)	M3 (64)	M4 (95)	M2 (96; 0.134)	M7 (38)	M8 (18)	M10 (23)	M6 (64)	M1 (89)

Table A.10. La36 instance settings and job deterioration rate.

la36	Operation														
Job	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
J1	M5 (21; 0.160)	M4 (55; 0.177)	M7 (71; 0.144)	M15 (98)	M11 (12)	M3 (34)	M10 (16;0.190)	M2 (21)	M1 (53; 0.159)	M8 (26; 0.036)	M9 (52)	M6 (95)	M13 (31)	M12 (42)	M14 (39; 0.179)
J2	M12 (54)	M5 (83)	M2 (77; 0.043)	M8 (64)	M9 (34)	M15 (79)	M13 (43)	M1 (55)	M4 (77)	M7 (19)	M10(37; 0.065)	M6(79; 0.173)	M11 (92)	M14 (62)	M3 (66)
J3	M10 (83)	M6 (77)	M3 (87)	M8 (38)	M5 (60)	M13(98; 0.027)	M1 (93)	M14 (17)	M7 (41)	M11 (44; 0.192)	M4 (69)	M12 (49)	M9(24; 0.113)	M2 (87)	M15 (25)
J4	M6 (77)	M1 (96)	M10 (28)	M7 (7; 0.106)	M5 (95)	M14(35; 0.195)	M8 (35)	M9(76; 0.137)	M12 (9)	M13 (95)	M3 (43)	M2 (75)	M11 (61)	M15 (10)	M4 (79)
J5	M11 (87)	M5 (28; 0.016)	M9 (50)	M3 (59)	M1 (46)	M12(45; 0.194)	M15 (9; 0.146)	M10 (43)	M7 (52; 0.134)	M8 (27)	M2 (91)	M14 (41)	M4 (16)	M6 (59)	M13 (39)
J6	M1 (20)	M3 (71)	M5 (78)	M14 (66)	M4(14; 0.050)	M13 (8)	M15 (42)	M7(28; 0.104)	M2 (54; 0.042)	M10 (33)	M12(89; 0.188)	M9 (26)	M8(37; 0.035)	M11 (33)	M6 (43; 0.137)
J7	M9 (69; 0.032)	M5 (96)	M13 (17; 0.042)	M1 (69)	M8 (45)	M12 (31)	M7 (78)	M11(20; 0.115)	M4 (27)	M14 (87; 0.045)	M2 (74)	M6 (84)	M15 (76)	M3 (94)	M10 (81; 0.086)
J8	M5 (58)	M14 (90)	M12 (76)	M4 (81)	M8 (23)	M10 (28)	M2 (18)	M3 (32; 0.020)	M13 (86)	M9 (99)	M15 (97)	M1 (24)	M11 (45)	M7 (72)	M6 (25; 0.040)
J9	M6 (27; 0.195)	M2 (46; 0.154)	M7 (67)	M9 (27)	M14(19; 0.171)	M11 (80)	M3 (17)	M4 (48)	M8 (62)	M12 (12)	M15(28; 0.074)	M5 (98)	M1 (42)	M10 (48)	M13 (50; 0.123)
J10	M12 (37)	M6 (80)	M5 (75; 0.022)	M9(55; 0.133)	M8 (50; 0.042)	M1 (94)	M10 (14)	M7 (41)	M15 (72)	M4 (50)	M11 (61)	M14 (79)	M3 (98)	M13 (18)	M2 (63)
J11	M8 (65)	M4 (96)	M1 (47)	M5 (75)	M13 (69)	M15 (58)	M11 (33; 0.137)	M2 (71)	M10 (22)	M14 (32; 0.066)	M6 (57)	M9(79; 0.077)	M3(14; 0.107)	M12 (31)	M7 (60; 0.012)
J12	M2 (34; 0.041)	M3 (47)	M4 (58)	M6 (51)	M5 (62)	M7 (44)	M10 (8; 0.070)	M8 (17; 0.115)	M11 (97)	M9 (29)	M12(15; 0.157)	M14 (66)	M13 (40)	M1 (44)	M15 (38)
J13	M4 (50)	M8 (57)	M14 (61)	M6 (20)	M12 (85)	M13 (90)	M3 (58)	M5 (63)	M11 (84)	M2 (39)	M10 (87)	M7 (21)	M15 (56)	M9 (32)	M1 (57; 0.060)
J14	M10 (84; 0.028)	M8 (45)	M6 (15)	M15 (41)	M11 (18; 0.183)	M5 (82)	M12 (29)	M3 (70)	M2 (67)	M4 (30)	M14 (50; 0.149)	M7 (23)	M1 (20)	M13 (21)	M9 (38)
J15	M10 (37)	M11 (81; 0.095)	M12 (61)	M15 (57)	M9 (57)	M1 (52)	M8 (74)	M7 (62)	M13 (30)	M2 (52; 0.045)	M3 (38)	M14 (68)	M5 (54)	M4 (54)	M6 (16)

Annex B - Machine deterioration rates

Table B.1. Machine deterioration rates for all instances.

	Instances									
Machine	la04	la08	la12	la16	la17	la22	la23	la28	la31	la36
1	0.051	0	0	0	0	0.023	0	0	0	0
2	0	0	0.142	0	0	0	0	0	0	0
3	0	0.013	0	0	0	0	0	0.096	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0.168	0	0	0
6				0.094	0.151	0	0.146	0.062	0	0
7				0	0.129	0	0	0	0	0.027
8				0	0	0	0	0	0	0.089
9				0	0	0.129	0	0	0.116	0.055
10				0.015	0	0	0	0	0.136	0
11										0
12										0
13										0
14										0
15										0

Annex C – Deterioration functions

The functions employed to study deterioration were represented to understand each function's impact on the processing time of various operations. The reference processing times of these operations range from 5 to 99, thus cases were considered where this parameter takes on the values of 10, 50, and 100. Regarding the deterioration rate, extreme values of 0.005 and 0.200 were analyzed, along with an intermediate case of 0.103.

For job deterioration, the independent variable is the waiting time between two consecutive operations and for the machine deterioration, it is the previous total machine processing time. In Figure C.1, the functions describing deterioration for the different considered parameters are depicted. Regarding job deterioration, the impact on the processing time is represented by the left hand-side of the graphs, since usually small times values are observed. Note that since the makespan is minimised, the job waiting times tend to be small. On the other hand, total machine processing times are only small for the first operations they process.

Regarding small deterioration rates, *i.e.*, 0.005, the linear and the exponential functions do not appear to have a significant impact on the actually processing time; not even if the waiting/processing time is large. On the other hand, the sigmoid function, despite leading to a much larger processing times, never reaches double the imposed threshold value, *i.e.*, the double of the reference processing time.

For intermediate deterioration rates, *i.e.*, 0.103, the behavior of the various functions varies depending on the reference processing time: for low values, the exponential function has no significant impact (Figure C.1b). However, for intermediate and large reference processing times, both the exponential and sigmoid functions lead a rapid increase of the processing time. The most impactful being the exponential in the presence of a large reference processing time.

Finally, for large deterioration rates, *i.e.*, 0.200, the linear function has the largest impact in the presence of small reference processing times (the exponential having the smallest), the exponential has the largest impact in the presence of intermediate and larger reference processing times; the latter leading to extremely large processing times even in the presence of very small waiting/processing times.

Therefore, the intermediate case (Figure C.1e) is the one in which the three functions exhibit the most similar behaviours. This scenario is the most common, as the parameters of this case correspond to the average cases.

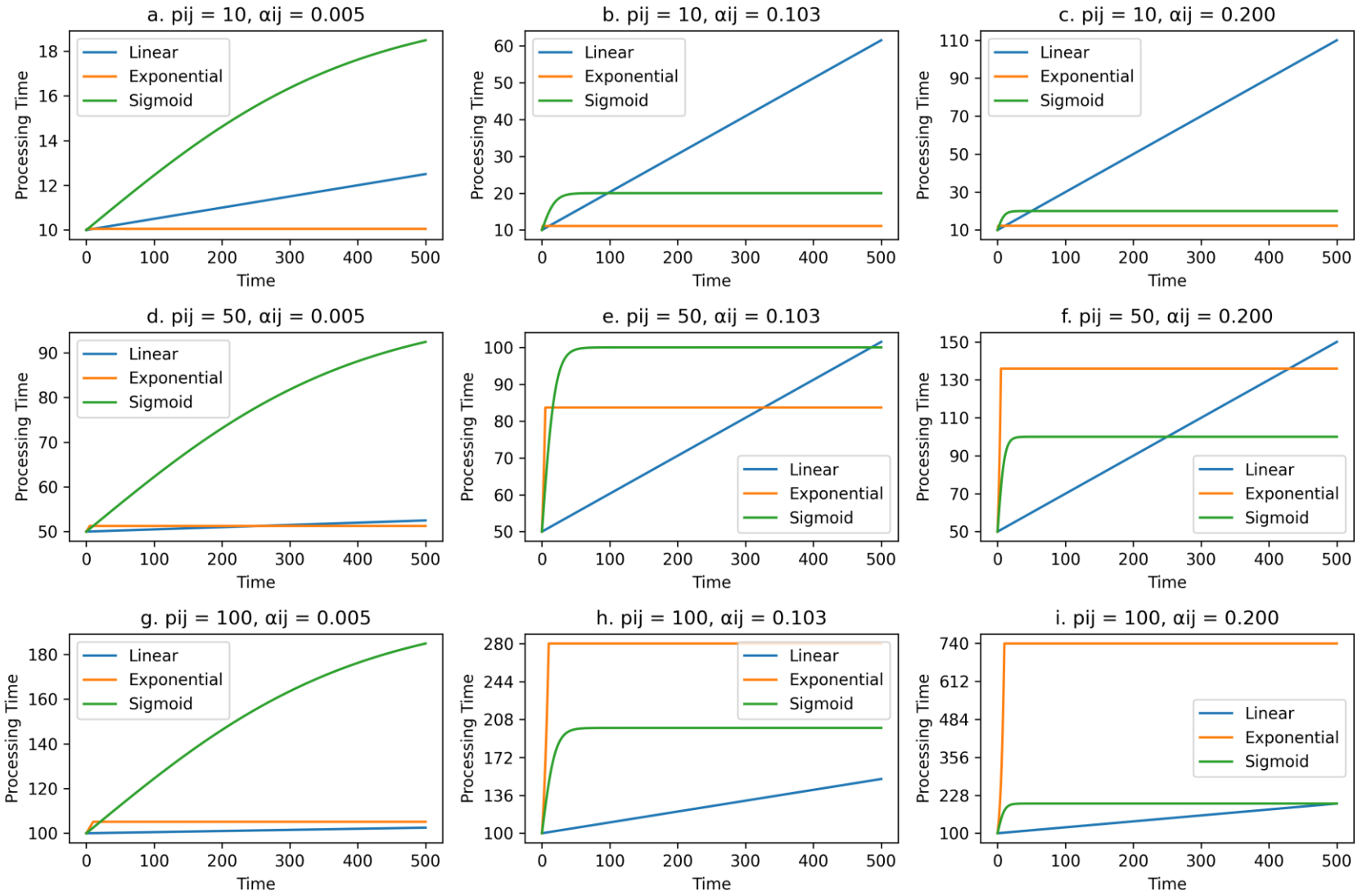

Figure C.1. Deterioration associated with time: impact on the actual processing time considering three different reference processing times (i.e., 10, 50, and 100-time units) and three different deterioration rates (i.e., 0.005, 0.103, and 0.200).