TÉCNICO LISBOA

Multi-objective meta-heuristics applied to flexible job shop scheduling problems

Joana Inês Cunha Cabral

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. Susana Margarida da Silva Vieira
Eng. Miguel de Sousa Esteves Martins

Examination Committee

Chairperson: Prof. Duarte Pedro Mata de Oliveira Valério
Supervisor: Prof. Susana Margarida da Silva Vieira
Members of the Committee: Prof. José Firmino Aguilar Madeira
Prof. Luís Manuel Fernandes Mendonça

January 2021

Acknowledgments

For this part, I could ramble on about all the wonderful people that helped and guided me through this journey. But I prefer to keep it simple, and only make a special acknowledgment to someone I can no longer acknowledge in person. To Idalisa, for she was the first professor that truly inspired me to never stop learning and always be the best version of myself. For she was the first woman I looked at and thought, I want to be like her when I grow up. Well, her path is over now, and I might not have been able to show her just how much her guidance changed and moulded me, but for that I will always be thankful.

Resumo

Com um tremendo aumento do poder computacional observado nas últimas duas décadas e as melhorias na recolha e análise de dados, as indústrias voltaram-se para a digitalização. Na Indústria 4.0, o escalonamento através de funções multi-objectivo é agora uma oportunidade de fornecer soluções que ponderam vários objectivos conflituosos e proporcionam uma tomada de decisão optimizada. Para resolver este problema, foi implementado um algoritmo de Colónia de Abelhas Artificiais Multi-Objectivo. O algoritmo proposto combina um método de codificação que apenas gera soluções viáveis com diferentes técnicas para gerar uma população inicial e para gerar novas soluções. Para medir a qualidade das soluções e o desempenho global do algoritmo, foram utilizadas duas métricas, o Hipervolume e a Distância Média Ideal. Para obter o conjunto ideal de parâmetros que maximiza o desempenho do algoritmo ABC, foi utilizado um algoritmo de optimização Bayesiano. Além disso, este algoritmo foi validado e o seu desempenho avaliado e testado para o Problema de Programação Flexível de Job Shop utilizando dois tipos diferentes de conjuntos de dados de referência, com um total de 15 instâncias avaliadas. A abordagem proposta mostra um bom desempenho, competitivo com o estado da arte.

Palavras-chave: Escalonamento de Produção Flexível, Multi-objetivo, Algoritmo de Colónia de Abelhas Artificiais, Afinação de Parâmetros, Optimização Bayesiana

Abstract

With a tremendous increase in computational power observed in the past two decades and the improvements in data collection and analysis, industries turned to digitalization. Within Industry 4.0, scheduling using multi-objective functions is now an opportunity to provide solutions that address several conflicting objectives and provide optimized decision-making. To address this problem a Multi-Objective Artificial Bee Colony algorithm was implemented. The proposed algorithm combines a coding method that only generates feasible solutions with different techniques to generate an initial population and to generate new solutions. To measure the quality of solutions and the overall performance of the algorithm, two metrics were used, the Hypervolume and the Mean Ideal Distance. To obtain the ideal set of parameters that maximizes the ABC algorithm performance, a Bayesian optimization algorithm was used. Moreover, this algorithm was validated and its performance evaluated and tested for the Flexible Job Shop Scheduling Problem using two different types of benchmark datasets, with a total of 15 instances evaluated. The proposed approach shows a good performance, similar to that of the state of the art.

Keywords: Flexible Job Shop Scheduling, Multi-objective, Artificial Bee Colony Algorithm, Parameter Tuning, Bayesian Optimization

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Throughout the last few decades, the world has experienced an exponential growth of data gathering, storage and treatment. This growth has been correlated with the, just as well exponential increase, in computational power. A continuous exploration and effort to maximize efficiency in all sorts of processes and industries, has taken humanity from the steam engine powered factories, that define the first industrial revolution, in the late 19^{th} century, passing through incremental technological progress, such as mass production and automated systems applied to industries, to the digital era of today, defined by a new stage of systems with new capabilities to integrate and interact with information, cyber-physical systems, the Internet of Things (IoT) and cloud computing, [1, 2].

A transformation in industry has started, from classical automation units to fully integrated automated facilities, that are able to communicate with each other. These new facilities enable an increase not only in overall efficiency, but also speed in which processes are handled, overall quality and productivity. Thus creating industries with processes that are much more efficient and with greater flexibility to respond to the ever changing market needs, enabling companies to produce faster, with better quality and highly customized products at lower overall costs, [3, 4].

Firstly formalised in Germany, the term Industry 4.0 describes the fourth industrial revolution, marked by fast changing technologies like the ones already stated above. All of these instruments described contribute to new promising paths for industries and companies. Smart factories, are key drivers in improving manufacturing operations and a motivation of manufacturing intelligence [5]. Through digitalization, optimization, production customization, centralized automation and adaptation, industries can improve their operational efficiency and productivity.

Additionally, Industry 4.0 introduces a transformation from machine-based to evidence-based decision making manufacturing systems, where product-machine interaction is possible without human involvement, making an impact on time and costs, by reducing planning time, achieving better production control, lower energy consumption, improving quality and more. Smart factories exchange large amounts of data and information in real time between the different systems within a factory, allowing the different systems to perform better-informed decisions, in order to improve production status, such as throughput rates, energy consumption, inventory management and more. [6–8]

Scheduling for production management has always been an important problem in operations research, and within the scope of Industry 4.0 it is equally important, since it is a tool to improve the overall factory efficiency and productivity. From the collection of data from customers demands and factory state, scheduling can affect planning and optimize decisions, according to the objectives of the factory.

1.1 Scheduling for Production Management

In the context of production management, one of the most important steps of planning and operating manufacturing systems is scheduling. It is a common problem present in most industries, such as steel tubes production [9], pharmaceutical [10], crude oil [11], agricultural [12], textile [13] and rail transportation of hazardous materials [14]. The scheduling problem can be defined as the allocation of resources over time, that suits a given performance indicator, and takes part in manufacturing and production systems from industry 4.0 or traditional industry.

In industry 4.0, an intelligent scheduling system can enable jobs to be scheduled based on AI techniques and problem solvers. When referring to an intelligent manufacturing system, a generic framework categorizes the research topics into smart design, smart machines, smart monitoring, smart control, and, finally, smart scheduling. This last topic mainly includes advanced models and algorithms to draw on the data captured by sensors in order to achieve real-time reliable scheduling and execution [1].

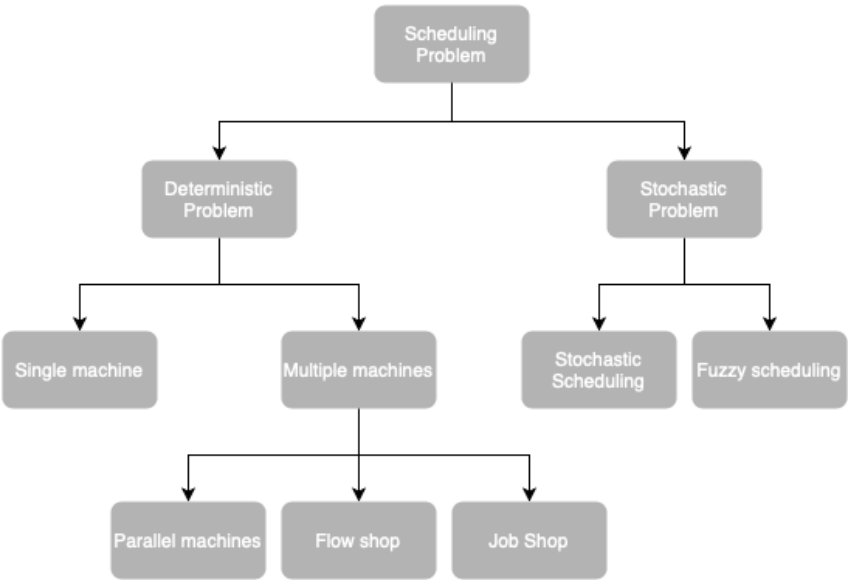
Scheduling focuses on finding the optimal or near-optimal schedule, which mainly is the job sequence and the operation-to-machine assignment, while subject to constraints such as time. This can be either a simple task or a very complicated one, depending on the process domain, the constraints involved in the process and its performance indicators [15]. A schedule optimizes one or more objectives that determine the quality of that schedule. These objectives, also described as performance indicators, can be the minimization of completion time, the reduction of machine usage, the minimization of job tardiness, minimizing costs and many others.

Focusing on static environment scheduling, the nature of the different problems found can be divided into two categories [16]:

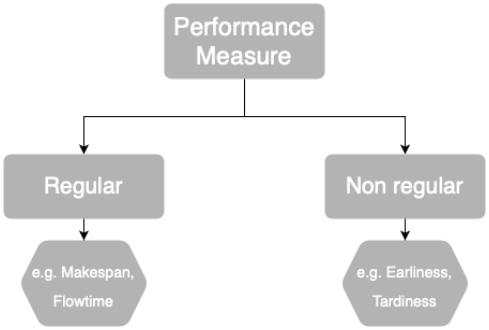
- Deterministic - processing times and other parameters are set and known *a priori*;
- Stochastic - parameters and processing times are modelled as random variables that follow a probability distribution.

Among the deterministic problems, a further classification is made, distinguishing single from multiple machine problems. The difference between one another focuses on the increased complexity of the problem, as shown in Figure 1.1(a). When only one machine processes the whole set of finite jobs, the problem is to find the sequence in which to process the jobs. Multiple machines directly relate to more complex arrangements, these configurations, by ascending order of complexity are parallel machine, flow shop and job shop.

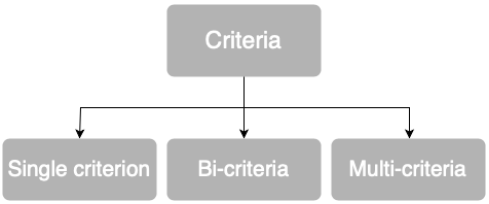
- Parallel machine problems [17] - studies the construction of schedules for parallel machines processing a set of one operation jobs. The main objective is to guarantee that the execution of all



(a) Classification of scheduling problems.



(b) Classification of performance meaures involved.



(c) Classification of criteria.

Figure 1.1: Classification of the scheduling problems.

jobs is done in an equitable amount of time. This theory marked the first steps for the theory of multiple machines, according to [16].

- Flow shop problems [18] - each job is defined by stages, and each step is performed on a machine, each job has to pass through each machine and the machines need to be arranged in a certain manner, this describes the pure flow shop. Another type of flow shop is the permutation flow shop [19].
- Job shop problems [20] - in this problem, the operations of each job can be assigned to a machine for processing, meaning that a job can befall on a machine more than once. The classical job shop problems has a set of identical machines, and each operation can only be performed in a specific machine. The flexible job shop problem as a set of multi-purpose machines, where each operation can be performed on more than one machine.

As for the performance measures, classification divides them into two groups, regular and non-regular, as it is shown in Figure 1.1(b). Further explanation on this matter will be given in the following sections.

As for the classification shown in Figure 1.1(c), it refers to the segmentation of different works by their

use of the criteria. It divides into single criterion, where only one performance measure is analysed, bi-criteria, which studies the effect of two performance measures and finally, multi-criteria, examining the effect of three or more performance measures in scheduling.

Multiple Machines Scheduling

Taking a further look into multiple machines scheduling problems, a more detailed classification arises [21], dividing these problems into four segments, open shop, flow shop, job shop and flexible job shop. The next paragraph provides a more detailed description of each segmented problem in order of increased complexity. A more graphical representation is presented in Figure 1.2.

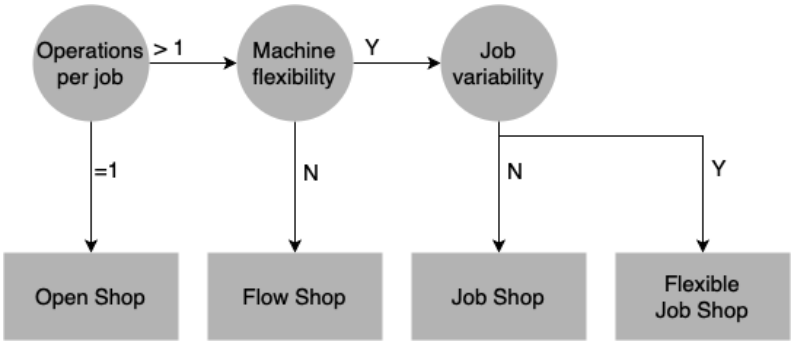


Figure 1.2: Classification of the shop scheduling problems.

The open shop scheduling problem handles the assignment of single operation jobs to machines. The flow shop scheduling problem also considers jobs with more than one operation, imposing job variability, but since each job has to pass by each machine, it has no machine flexibility. The job shop scheduling problem additionally states that the order of machines can differ from job to job. This increased complexity translates into imposing machine flexibility.

Finally, the most complex problem, that in addition to the previous job shop problem establishes job variability, the flexible job shop problem. It states that each operation of each job can be executed by a different set of machines.

In this work, the shop problem chosen is the Flexible Job Shop Problem, FJSP, the following section will enhance the complexity and interest around it.

Flexible Job Shop Problem

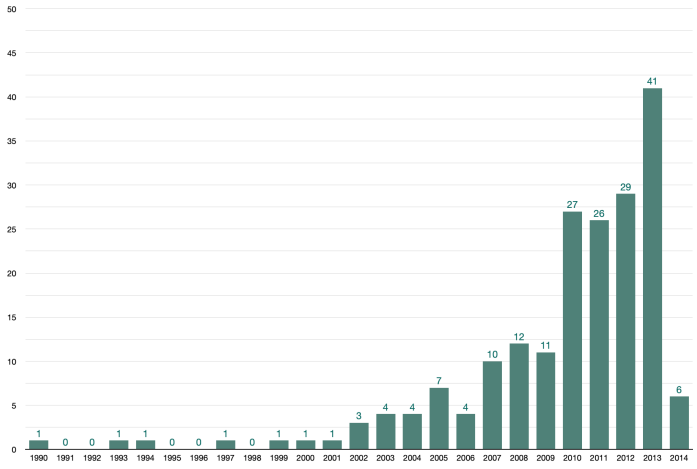
In the previous pages, the complexity of the FJSSP was detailed. It arises naturally from the way in which the problem itself is stated.

A set of operations belonging to a job can be performed by a chosen machine, each machine can only process an operation at a time repeatedly making each machine incessantly available. All this with the main target of finding the sequence that minimizes or maximizes a performance measure.

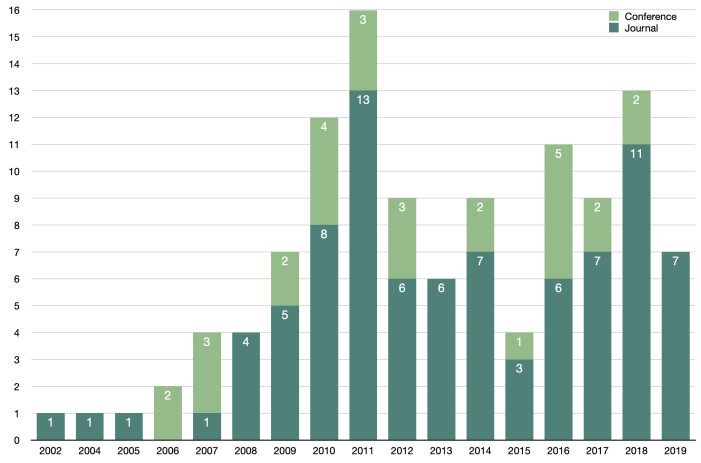
The FJSP is an NP-hard problem, with an additional resolution dimension. Apart from having to find the sequence of jobs it also needs to discover the machine allocation. Further details on the complexity of this problem are shown in [22].

Additionally, since FJSP was firstly introduced, it spiked the curiosity of researchers, in a review article from 2015 [23], it can be seen that between 1990 and 2014, a total of 404 publications that addressed the Flexible Job Shop Scheduling Problem, FJSSP, were found, and categorized into 410 different applications. Figure 1.3(a), shows the distributions of papers addressing the FJSSP by year, according to the survey mentioned.

In [24] from 2020, out of 218 research papers analysed, 117 articles from 2002 to 2019, addressed the multi-objective optimization of FJSSP with an heuristic approach. It also shows the yearly trend of publications, showing a rise in the last decade, and finding an average of 9 publications per year. Figure 1.3(b) shows this previously described yearly trend. Both figures allow for the conclusion that the interest on both FJSSP and the particularized problem for multi-objective has been increasing throughout the years.



(a) Yearly distribution of papers addressing the FJSSP, since it was first introduced until 2014, according to [23].



(b) Yearly distribution of papers addressing the multi-objective FJSSP, from 2002 until 2019, according to [24].

Figure 1.3: Yearly distribution of papers, tackling the FJSSP (a), and tackling specifically the multi-objective FJSSP (b).

Evaluation Criteria of Flexible Job Shop Scheduling Problems

As already stated, the criteria used to evaluated the performance can be divided into regular and non-regular. An objective function is considered regular if it is an non-decreasing function of job completion times. The main objective is to minimize that measure [25], thus these regular criteria intends to finish an activity as early as possible. Some examples of regular criteria are the makespan (C_{max}), total workload of machines (W_T), critical workload (W_M), mean flow time ($\overline{C}$), and others.

As for non regular criteria, this type of performance measures are described by a not monotonic function of the job completion times, for example, finishing a job early in a just-in-time environment would mean that there is an excess of work-in-progress, (WIP).

Taking into account the analysis done by [23] on this matter, out of 197 research papers evaluated, six objective functions make up for 75% of the criteria used. These six objective functions, are makespan, total workload of machines, critical workload (workload of the most loaded machine), mean tardiness, total tardiness and production costs. 47.21% of those papers use single objective optimization, roughly 4% use bi-criteria optimization, and 23.35% used multi-objective optimization, more specifically, use three regular objective functions, the makespan, total workload of machines and workload of the most loaded machine.

For the purpose of this work, these objective functions will be used, makespan, total and critical workload. Their conflicting nature makes them appropriate to use for multi-objective optimization as they bring significant complexity to the problem.

1.1.1 Scheduling Techniques

Commonly, two approaches are considered for solving the multi-criteria FJSSP [20], the first one consists on a deterministic approach mainly composed by exact methods, the second one consists on heuristic or meta-heuristic approaches.

First it is important to differentiate the use of multi-objective optimization as opposed to single objective. From Figure 1.3(b) it is easy to see that this problem has awakened the interest of researchers. Among the many reasons behind that lies the approximation to real-world problems, and simultaneously the increased complexity when compared to single objective optimization.

Multi-objective optimization addresses the problem of conflicting criteria, which is described as competing objectives, meaning that the difficulty of the problem drastically increases. When dealing with multiple criteria the goal shifts to finding a solution that presents the best trade-off between these conflicting objectives.

The first approach to handle Multi-Objective Flexible Job Shop Scheduling Problem, MOFJSSP, relies on mathematical models, typically used in smaller size problems due to the need of high computational power. The second approach has been widely studied within the scientific community due to its capability to generate good solutions for NP-hard and combinatorial problems, such as the FJSSP. Within the heuristic approaches two types exist: hierarchical and integrated [26]. Hierarchical approaches intend to reduce complexity through decoupling the sequencing and routing sub-problems, by solving them

consecutively. On the contrary, integrated approaches do not differentiate between the sequencing and routing sub-problems.

Figure 1.4 refers to the distribution of the different techniques used to solve the multi-objective FJSSP, according to [24]. It shows the five different methods found to evaluate multi-objective functions, and that Pareto optimality is used as ranking and selection method on 56.9% of the papers analysed. The figure shows the arrangement of methods used sorted by technique [24]. The genetic algorithm, that belongs to the evolutionary algorithms category, is separated in this configuration due to the number of papers evaluated that utilized this specific algorithm. It also shows that algorithms such as Artificial Bee Colony and other swarm intelligence heuristics favour Pareto optimality method, as well as Evolutionary Algorithms and Genetic Algorithms, whereas Ant Colony Optimization algorithms use the weighted sum method.

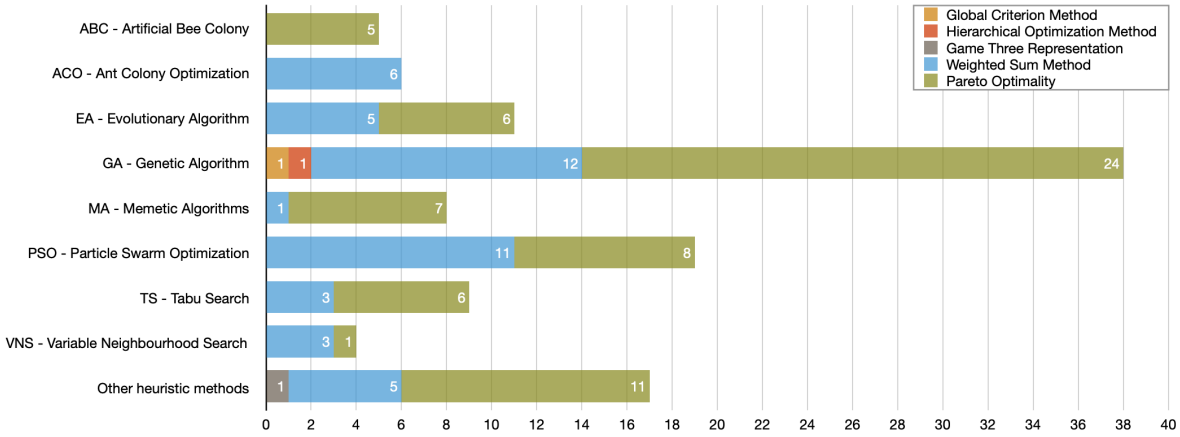


Figure 1.4: Different techniques applied to solve the MOFJSSP.

Deterministic Techniques

The deterministic approaches, composed of mathematical models of the multi-objective FJSSP are established through multi-criteria Mixed Integer Linear Programming (MILP), and are typically extensions of previously published single criterion models.

For instance, in [9] it is contemplated the production of seamless steel tubes, with several machines available in parallel that bring flexibility to the scheduling problem. The problem in question provides two quests for scheduling: first determine the allocation of jobs for each machine and second the sequence of jobs at each stage. The authors developed a non-linear mixed integer linear programming model with the total idle time of machines and the waiting time of jobs as the performance measuring criteria, and then proposed a genetic algorithm.

A review on several mathematical models designed for the FJSSP from [27], evaluates these models with respect to the decision variables that represent the operations execution sequence of the different machines. The authors in [28] present a multi-objective model to minimize makespan and total tardiness.

Mathematical models for MOFJSSP and solutions using exact method are mainly used for small and medium sized problems, due to their ability to provide fairly high quality solutions, which can be used as

a comparison cornerstone for the development of heuristic methods.

The masters thesis [29] provides a detailed analysis showing the inefficiency of these models for larger sized problems.

Heuristics and Meta-heuristics Techniques

Solving production scheduling problems with the aid of heuristics and meta-heuristics has become common over the past few decades. Heuristics are methods that require few to medium computational power, that find sensibly good quality solutions, but have a major setback which is not guaranteeing to find globally optimal solutions. The difference between heuristics and meta-heuristics is that the latter are high level heuristics that choose and guide heuristics, as local search, in order to escape local minimums, but despite these efforts, can not fully guarantee the global optimum is reached.

Heuristics are methods supported by the idea of exploring in neighbourhoods, such methods begin from an initial solution and iterate attempting to find better solutions within the neighbourhood of previously formed solutions. These search techniques stop after a given stopping criterion is reached, usually after enough iterations and the solution is ”stuck”.

Meta-heuristics, for instance, genetic algorithms and tabu search, extend local search by escaping local optimum solutions. These meta-heuristics, escape local optimum by creating better starting points for the local search in an astute manner, instead of just generating random solutions, or by exploring worse solutions. This will increment the space and the variety of the search for the following iteration.

In [30] a Pareto optimality-based hybrid algorithm is presented to solve the MOFJSSP. This hybrid technique merges the knowledge illustration skills of fuzzy logic with the adaptive abilities of evolutionary algorithms. First it generates an initial population with an approach by localization and then uses a genetic algorithm to find better solutions.

Another example can be found in [31], where a hybrid multi-objective evolutionary approach is used to solve the MOFJSSP, based on the well known NSGA-II. It uses a modified crowding distance measure to maintain diversity within the population, and obtain better results if there are fewer non-dominated solutions found. It also uses local search based on critical path to improve the convergence ability of the algorithm.

Other examples of meta-heuristics used to solve the MOFJSSP are Genetic Algorithms (GA) [32, 33], Particle Swarm Optimization (PSO) [34], Memetic Algorithms (MA) [35], Artificial Bee Colony (ABC) [36, 37], Variable Neighbourhood Search (VNS) [38], Tabu Search (TS) [39] and Ant Colony Optimization (ACO) [40]. Meta-heuristics have been widely used to solve production scheduling problems, not only in the flexible job shop problem, but also for areas such as open shops, flow shops, classical job shops, batch processing and others.

Moreover, many of the approaches used to solve multi-objective optimization problems can be classified into three different categories [41]. The first category addresses the multi-objective optimization problem by aggregating the objective functions used into a weighted sum. The second category is a non-Pareto approach were each objective is handled separately. The third category is based on finding Pareto optimality. Although there are some examples of algorithms using the first category to solve

the multi-objective optimization problem, like [42–45], the majority of approaches use Pareto optimality to handle the multi-objective optimization problem, as shown by figure 1.4. Some examples of papers addressing MOFJSSP with a Pareto optimality approach are [46],[47] and [48].

1.1.2 Artificial Bee Colony Algorithm

Artificial Bee Colony algorithms integrates the Swarm Intelligence branch of algorithms that is summarily defined by algorithms that model the collective behaviour of a decentralized population of interacting, self-organized swarms [49]. Swarm meta-heuristic were first introduced in the early 1990s, with the advancements on computational power, the computational cost was greatly reduced and researchers rotate towards drafting their ideas from nature to solve computer aided problems [50]. Efficient meta-heuristic procedures have been developed based on honey bees intelligent behaviour, in particular, they are motivated by mutual cooperation and self-organization to guide the bees in the allocation of tasks and search for new food sources.

Swarm is the term used for describing the aggregation of a population of natural creatures, like fishes, birds, insects, through collective behaviour. Such behaviour was extensively studied by researchers, for methods such as Ant Colony Optimization [51] or Fish School Search [52]. Swarm Intelligence is gaining importance in research areas such as computer science, biology, engineering and operations research. Evolutionary algorithms are search algorithms that resemble natural evolution and the Swarm Intelligence branch is no different, with publications in the fields of renewable energy [53], medical image processing [54], robotics [55], smart structures [56], to name a few.

Similarly to evolutionary algorithms, swarm intelligence methods also start with an initial population of organisms, the initial solutions, and from that point perform mutations and recombinations, and select the fittest organisms to subsist each generation, improving the solutions. The difference is that in swarm procedures, the unsupervised isolated organisms perform changes based on the perception of their neighbourhood, showing a stochastic behaviour. As for the idea behind ABC, it relies on the self-organized division of tasks presented in swarm intelligence that can be observed in honey bee colonies.

The Artificial Bee Colony algorithm, ABC, is a meta-heuristic approach first introduced in 2005 by Karaboga [57]. Karaboga then evaluated the performance of the proposed algorithm by comparing with GA, PSO and PS-EA (Particle Swarm inspired by an evolutionary algorithm) in [49]. In this study it showed that the proposed algorithm when compared to the GA, PSO and PS-EA methods, is able to escape local minimum and behaved efficiently for optimization problems with multi-variable and multi-modal functions. This method is inspired by the behaviour of honey-bee swarms and has three different types of bees: the scout bees that fly randomly and without guidance within the search domain, then the employed bees that use the neighbourhood space to select a random solution in order to perturb, and finally the onlooker bees that utilize fitness functions to select the probabilistic solution by exploiting the neighbourhood space. The bees act based on greedy selection. This means that when they find a new location with higher nectar amount, they forget the older location and memorize the new one.

This algorithm has an extensive number of applications, on diverse fields such as Renewable Energy

Systems [58, 59], in Nuclear Energy Systems [60], in medical Image Segmentation [61] and in Robotics [55].

In [58] it is presented an ABC to solve power loss minimization, voltage stability index and emissions for photovoltaic, wind and fuel cell systems. The proposed multi-objective achieved results with considerably good quality and better diversity when compared to NSGA-II and MOPSO procedures.

Within the renewable energy scope, [59] proposed an ABC algorithm to accurately identify the parameters of solar cells. When comparing the technique with other evolutionary algorithms, it showed a better search capacity to face multi-modal objective functions, and the experimental results show the robustness and accuracy of the method applied.

In the field of nuclear energy [60] proposes an ABC algorithm to find the best fuel lattice design of a pressurised water nuclear reactor that maximises the operating time. In this research it was also applied a GA and a PSO for the same problem and the results obtained verified that the ABC outperformed both. It also shows that the artificial bee colony algorithm used offers more flexibility and less parameters for controls.

In [61] it is presented a novel automated approach for blood vessel segmentation with a clustering approach. Initially fuzzy c-means (FCM) is optimized using an ABC to localize the retinal vessels and then using a local search for boosting the cluster centres.

In the field of robotics, [55] presents a co-evolution global-best-leading ABC algorithm to handle the optimization problems with dimension-independent and dimension-dependent parts. It also address the robot path planning problem to test the effectiveness of the proposed algorithm. The proposed algorithm finds higher quality and more robust results than the comparison algorithms.

1.2 Objectives

The main objective of this thesis is to develop tools for solving the Multi-Objective Flexible Job Shop Problem, given this, it can be divided into three sub-objectives:

- Implement a Multi-Objective Artificial Bee Colony algorithm using a Pareto optimality approach;
- Implement metrics that evaluate the convergence and diversity of non-dominated solutions for tuning the parameters used in the algorithm;
- Test the overall performance of the algorithm developed using two different benchmark datasets.

1.3 Contribution

The contributions of this thesis focus on the development and implementation of a method to produce schedules with multi-objective optimization.

This work follows the plan of creating a schedule with considerable high quality that is capable of reaching high levels of performance through the minimization of the makespan, total workload of ma-

chines and maximum workload. To achieve these results an Artificial Bee Colony algorithm was implemented,with a coding method that finds feasible solutions every time. Different strategies were used to generate an initial population and to apply mutation and crossover procedures. To improve quality of the solution a local search operator based on the critical path was applied.

Furthermore, to deal with the different parameters needed to employ the implemented algorithms, a Bayesian Optimization technique was used. Lastly, the performance of the proposed algorithm is evaluated using two benchmark datasets and comparing the results with several state-of-art algorithms.

1.4 Outline

From this point forward, this work is organized as follows.

Chapter 2 gives a further explanation and formally formulates the Flexible Job Shop Scheduling Problem, for a multi-objective approach.

Chapter 3 details the implementation of the multi-objective Artificial Bee Colony with a Non-Dominated Sorting Algorithm, and describes the process behind parameter tuning using Bayesian Optimization.

Chapter 4 discusses the results gathered by the proposed algorithm, as well as the performance evaluated through benchmarks.

Chapter 5 summarizes the results obtained and gives some perspectives on the future work based on the insights retrieved from this work.

Chapter 2

Flexible Job Shop Problem

This chapter will describe the Flexible Job Shop problematic and provide a description and formulation of the respective problem.

2.1 Flexible Job Shop Scheduling Problem

The Flexible Job Shop Scheduling Problem (FJSSP) is the generalization of the classical Job Shop Scheduling Problem (JSSP). This means that the main objective of the classical static JSSP is to find the schedule that processes n jobs on m machines following a given objective function.

To generalize the JSSP, FJSSP has an additional condition that imposes job variability and also creates the need for an operation sequence solution, not only the machine assignment. FJSSP allows operations to be executed by a set of machines instead of a single machine, providing a more realistic approach to the scheduling problems.

The FJSSP is stated as follows:

- A set of jobs, independent from each other $J = \{J_1, ..., J_n\}$;
- A set of independent machines, $M = \{M_1, ..., M_m\}$;
- A sequence of operations from each job J_i, $O_{ik} = \{O_{i1}, O_{i2}, ..., O_{ih}\}$;
- Each operation O_{ik}, must be processed by a subset of machines such that $M_{ik} \subset M$. If $M_{ik} = M$ $\forall i, k$ then the problem becomes a flexible job shop problem;
- Each machine can execute exactly one operation at a time;
- Each job and each operation must be processed exactly one time;
- Each job has a specific linear precedence structure with the sequence of machines it visits;
- If the precedent operation is still being processed, the remaining operations wait until the processing is completed;

- The processing time PT_{ikm} is given by the time it takes for operation O_{ik} of job J_i to be performed in machine M_m, without interruption.

- Machines must always be available at the start of the scheduling horizon;

- The start time of every operation O_{ik} on machine m is defined as ST_{ikm};

- The finishing time of each operation O_{ik} on machine m is defined as FT_{ikm}

The following table illustrates the problem formulation, the rows correspond to the operations and the columns to the machines, the entries correspond to the respective processing time (∞ corresponds to an operation that cannot be performed in that machine).

	M_1	M_2	M_3	M_4
O_{11}	4	7	6	5
O_{12}	2	6	∞	5
O_{21}	4	5	7	∞
O_{22}	5	∞	6	3
O_{23}	∞	5	4	7
O_{31}	5	3	∞	6
O_{32}	∞	∞	4	∞
O_{41}	2	4	5	∞
O_{42}	∞	4	2	∞
O_{43}	5	4	6	3

Table 2.1: Processing time of each operation O_{ji} in each machine M_i

As already stated, the FJSSP has an additional problem when compared to the classical JSSP, where the assignment of an operation is not fixed to a specific machine *a priori*, but instead it is assigned a set of machines able to perform that operation. This increases the complexity of the problem, since it has both a sequencing and a machine assignment problem.

The FJSSP can now be divided into two sub-problems, a routing problem that selects the machine to process each operation, and the sequencing problem on all the selected machines to obtain a feasible schedule. Additionally, some constraints related to this, are:

- The technique constraint which requires the operation to only be processed after all precedent operations have been processed, this will keep operations from overlapping and keep the machine available for other operations;

- The resource constraint which requires that one machine handles one and only one operation at a time;

- The precedence constraint for operations of the same job.

2.2 Objective Functions

The FJSSP intends to obtain a feasible schedule while quantifying performance measures. In manufacturing, the performance of a system relies on numerous factors, for example, cost, job manufacturing makespan, resources utilization, employee allocation, machine utilization, and many others.

In order to establish this performance quantification, some objectives are taken into account, in order to determine how the schedule must be created. This section will present some of the most important objectives to consider when scheduling a FJS.

Makespan

The makespan (C_{max}), represented in (2.1), is the maximum elapsed time between the beginning of each job and its ending task, this is the maximum completion time of all jobs in the production system. The intention behind using the makespan as an objective of the FJSSP is to produce feasible schedules where all jobs are completed as fast as possible.

$$C_{max} = \max\left(C_1, ..., C_j, ..., C_n\right), (j \in J, k \in M, i \in O_{ji}) \tag{2.1}$$

Another objective, that is correlated with the makespan, is the Weighted Completion Time, given by equation (2.2). With an increased parameter, the weight w_j, that corresponds to the importance of each job.

$$WCT = \sum_{j=1}^{n} \left(w_j C_j\right), (j \in J) \tag{2.2}$$

Workload

Machine workload can be divided into two objectives, the Total Machine Workload (TMW) given by equation (2.3), and the Maximum Machine Workload (MMW) given by equation (2.4). The first one (TMW) refers to the processing times of each operation, of each job, and its minimization will result in a more efficient allocation of the operations in its respective machines. The second one (MMW) considers the balance of workload between all machines and intends to avoid allocating too much work to a single machine.

$$W_t = \sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{i=1}^{n_l} \left(p_{jik} x_{jik}\right), (j \in J, k \in M, i \in O_{ji}) \tag{2.3}$$

$$W_m = \max_{1 \leq k \leq m} \left(\sum_{j=1}^{n} \sum_{i=1}^{n_l} \left(p_{jik} x_{jik}\right) \right), (j \in J, k \in K, i \in O_{ji}) \tag{2.4}$$

Tardiness and Lateness

Some objectives related to the due dates of jobs are the tardiness and the lateness objectives. To minimize the lateness, L_j, of a schedule is the same as minimizing the maximum lateness, L_{max} of that

schedule, and is defined in equations (2.5) and (2.6), where d_j corresponds to the due date.

$$L_j = C_j - d_j, (j \in J) \tag{2.5}$$

$$L_{max} = \max(L_i, ..., L_n) \tag{2.6}$$

Tardiness handles the number of tardy jobs as an objective, will qualify how tardy a job is, instead of measuring lateness and is given by equations (2.7) and (2.8).

$$T_j = \max\{0, c_j - d_j\}, (j \in J) \tag{2.7}$$

$$\sum_{j=1)}^{n} T_j, (j \in J) \tag{2.8}$$

Besides the objective functions described above, there are many others, such as Manufacturing Cost, Earliness, Unity penalty, and many others. The choice of an objective function depends on the practical uses of the considered scheduling problem.

For this case, three objective functions were taken into consideration, the Makespan, the Total Machine Workload and the Maximal Machine Workload. These objectives were chosen not only by their significance as to measure the efficiency of a schedule, but they are also in conflicting with each other.

Another motivation to choose these three specific objectives is the fact that they have been widely documented within the literature, which will help for the sake of results analysis.

2.3 Mathematical Formulation

The Multi-Objective Flexible Job Shop Scheduling Problem (MOFJSSP) determines the machine allocation for each operation and the sequence in which those operations will be performed in a way that optimizes multiple objectives. The mathematical formulation is presented below, based on [62].

But first it is important to give some definitions, equation (2.9) is used to describe a multi-objective problem without losing generality. Where x corresponds to the decision vector in space and y represents the objective vector with l objectives.

$$\min y = f(x) = (f_1(x), ..., f_l(x)), (x \in \Omega, y \in R^l, l > 1) \tag{2.9}$$

So, the multi-objective flexible job shop scheduling problem can be mathematically formulated as follows:

$$
\begin{aligned}
\text{Minimize} \quad & C_{max} = \max\left(C_1, ..., C_j, ..., C_n\right), (j \in J, k \in M, i \in O_{ji}) \\
& W_t = \sum_{k=1}^{m} \sum_{j=1}^{n} \sum_{i=1}^{n_l} \left(p_{jik} x_{jik}\right) \\
& W_m = \max_{1 \leq k \leq m} \left(\sum_{j=1}^{n} \sum_{i=1}^{n_l} \left(p_{jik} x_{jik}\right) \right), (j \in J, k \in K, i \in O_{ji})
\end{aligned} \tag{2.10}
$$

$$
\text{subject to} \quad ST_{jik} + \sum_{k} PT_{jik} \times X_{jik} \leq ST_{j,i+1,k}, (j \in J, k \in K, i \in O_{ji})) \tag{2.11}
$$

$$
ST^r_{jik} + \sum_{ji} PT_{jik} \times X_{jik} \leq ST^r_{lpk}, (j \in J, k \in K, i \in O_{ji}, p \in J, l \in O_{ji}) \tag{2.12}
$$

$$
ST_{jik} \leq ST^k_{lpk} + (1 - X_{jik}) \times BN, (j \in J, k \in K, i \in O_{ji}) \tag{2.13}
$$

$$
ST^k_{lpk} + (1 - X_{jik}) \times BN \geq ST_{jik}, (j \in J, k \in K, i \in O_{ji}) \tag{2.14}
$$

$$
\sum_{ij} X_{jik} = 1 \, (j \in J, k \in K, i \in O_{ji}) \tag{2.15}
$$

$$
\sum_{ij} X_{jik} \leq 1 \, (j \in J, k \in K, i \in O_{ji}) \tag{2.16}
$$

$$
X_{jik} \leq Y_j ik \, (j \in J, k \in K, i \in O_{ji}) \tag{2.17}
$$

$$
X_{jik} = \begin{cases} 1, \text{if operation } O_{ji} \text{ is selected to be performed on machine } k \\ 0, \text{otherwise} \end{cases} \tag{2.18}
$$

$$
Y_{jik} = \begin{cases} 1, \text{if operation } O_{ji} \text{ can be performed on machine } k \\ 0, \text{otherwise} \end{cases} \tag{2.19}
$$

Equations (2.11) to (2.19) ensure that all the previously described constraints in section 2.1 are respected. Equation (2.11) ensures that an operation belonging to the same job can only start once the previous one is finished, this constraint guarantees the technological sequence for each job. Equation (2.12) ensures that each machine will process one operation at a time, it represents a disjunctive constraint on the machine sequencing problem. Equations (2.13) and (2.14) guarantee the application of the link constraints, ensuring that the starting time of an operation is the same as the starting time of its machine. Equation (2.15) ensures that every operation is allocated to one machine from its candidate set, equation (2.16) ensures that only one operation is allocated to a specific machine at any given time and equation (2.17) guarantees that operations will only be allocated to machines that are available for them. Finally, equation (2.18) guarantees that an operation is allocated to a machine and equation (2.19) defines if the machine is able to process that operation.

In order to define the Makespan equation (2.1) is used, for the Total Machine Workload (2.3) and for the Maximal Machine Workload (2.4).

2.4 Representation

2.4.1 Disjunctive Graph representation

In order to represent the scheduling problem it will be used a disjunctive graph, as defined in equation (2.20).

$$G = (V, Con \cup Dis) \tag{2.20}$$

In the graph, operations are denoted by nodes $(i/j) \in V$, with V being the set of nodes that represent the operations, this set will include two dummy nodes, the starting node (1) and the finishing node $(N+1)$. The precedence constraints of each operation in each job is assured by the conjunctive arcs Con.

A conjunctive arc $(i,j) \rightarrow (k,j)$ denotes that operation O_{ji}, precedes operation O_{ki} belonging to that same job, these arcs are defined for any two operations within the same job. Also, these directed arcs ensure that the technological constraint is fulfilled. Adding to this premise, the starting node is connected to the first operation of each job, and the ending node is also directly connected to the last operation of each job, as it can be observed on figure 2.1(a).

The disjunctive arcs represent a sequence of jobs done on the same machine, at first these arcs exist in both directions, but in order to make a feasible schedule that fulfils both the machine assignment and operation sequence, only one direction of those arcs will be selected. Given the example on figure 2.1(a), at first, the disjunctive arcs represented by dashed lines would have either directions, but for each of those arcs, only one direction would be able to deliver a feasible schedule. The final graph will turn all the disjunctive arcs that allow to have a feasible schedule into directed arcs, this will be considered as the set Dis', ans thus obtaining a new acyclic graph $G' = (V, Con \cup Dis')$.

Each arc will have a weight, that translates into the length of that arc, which is the processing time of the operation from which that arc originated (example: the arc $(i,j) \rightarrow (k,j)$ will have a weight PT_{jik}). Respectively, the starting node and the ending node will have no processing time, so the weight of the conjunctive arcs that connects them to their respective operations will have no weight. The time to complete the longest path of the graph is the makespan, and the new directed graph will be given by equation (2.21), with S is an acyclic complete selection of the disjunctive edges turn into directed edges.

$$G(S) = (V, Con \cup S) \tag{2.21}$$

The final graph will deliver a feasible schedule corresponding to a sub-graph S which, contains all the conjunctive arcs, for each pair of disjunctive arcs, S contains only one of them and finally S is acyclic, meaning that it does not contain any directed cycle.

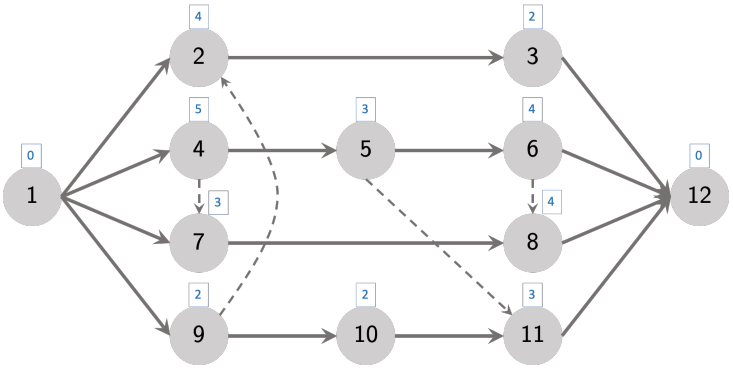
To further explanation, the graph in figure 2.1(a) presents the graph from the problem formulated in table 2.1 turned into a feasible schedule. Node 1 and Node 12 correspond respectively to the starting and the finishing nodes. Node 2 corresponds to operation $O_{1,1}$, Node 3 to operation $O_{1,2}$, Node 4 to operation $O_{2,1}$, and so on. The numbers above each node, in blue, represent the processing time of that operation. The full arrows describe conjunctive arcs, and the dashed arrows correspond to the disjunctive arcs.

2.4.2 Gantt Chart representation

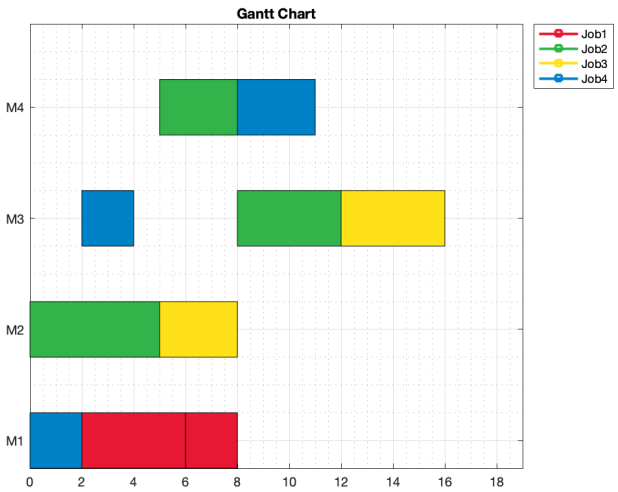
A Gantt chart is a commonly used bar chart that gives a visual representation of a feasible solution. It represents the allocation of the operations of each job and its given sequence. It gives also a representation of this allocation and also the starting time of each operation and its respective completion time and the idle time on each machine.

The schedule that minimizes both the makespan and workload will be a more compact schedule, where the last operation to be processed will be completed at a lower time instance, and well balanced in terms of operations in each machine, such that the minimum processing time is chosen, but also ensures that the operations are well distributed within the machines.

The main advantage in using a Gantt Chart is that the information that contains is easily visualised and interpreted. To corroborate this premise figure 2.1(b) easily gives information about the the number of jobs, operations and machines, the starting and completion time of each operation, it also deliver information about the machine allocation, operation sequence and idle times on each machine. Also by observing table 2.1 it can also be concluded that the schedule presented on this Gantt chart is not an optimal one.



(a) Disjunctive Graph



(b) Gantt Chart

Figure 2.1: Representation of a Disjunctive Graph and a Gantt Chart.

Chapter 3

Multi-Objective Artificial Bee Colony for Flexible Job Shop Scheduling

The first section of this chapter will provide a thorough description of the algorithm used to implement the Multi-Objective Flexible Job Shop Scheduling Problem, inspired by works [32, 33, 35–37]. The next section describes the process of tuning the parameters used in the algorithm, to obtain better solutions, using Bayesian Optimization.

3.1 Artificial Bee Colony with Non-Dominated Sorting Algorithm

In order to solve the MFJSSP described in section 2.3, it was implemented an Artificial Bee Colony with Non-Dominated Sorting Algorithm. The ABC algorithm was initially introduced by [57], and later enhanced in [49], essentially inspired by the intelligent foraging behaviour of a honeybee swarm, it contains three foraging bees, the employed bees, the onlooker bees and the scout bee.

A first step is taken by generating random food sources and associating the employed bees to those food sources. Then these bees will explore their neighbourhood in search for new food sources, and choose to keep exploring or drop the newly found food source, based on the amount of nectar it provides. Once the employed bees complete their process and enrich the hive with neighbouring food sources with considerably good amounts of nectar, this information will be shared with the onlooker bees.

And by now the employed bees have established a so called dancing area, which is the area from the hive where the information of food sources and their nectar amounts is traded. The onlooker bees will choose which food sources to explore based on the quality of that food source, determined by the nectar it provides, resulting in food sources with higher quality will captivate more of these onlooker bees. These bees will increase the dancing area by exploiting the chosen food sources with the same objective of finding higher quality food sources.

The exhausting food source from the employed bees, will then be associated with a scout bee that will then explore around that food source for a new and better one.

To translate this process, each food source represents a feasible solution, its nectar amount corre-

sponds to the quantitative values of each objective function. The employed and scout bees perform exploration, as the onlooker bees perform exploitation. This process will iterate until it meets a stopping criterion.

3.1.1 Procedure to implement a Pareto-based Artificial Bee Colony for the Multi-Objective Flexible Job Shop Problem

To properly define the algorithm implemented, first it is necessary to make some definitions. As previously stated in 2.3, a multi-objective problem is described as in 2.9, thus to define which solutions are better, and which are worse, it is necessary to apply some widely known notations and methods to classify them.

Starting with the **Pareto dominance**, straightforwardly, if a solution S_1 dominates another solution S_2 ($S_1 \prec S_2$, which denotes that S_2 is dominated by S_1), then 3.1 and 3.2 must always be true.

$$\forall i \in \{1, 2, 3, ..., l\} : f_i(S_1) \leq f_i(S_2) \tag{3.1}$$

$$\exists i \in \{1, 2, 3, ..., l\} : f_i(S_1) < f_i(S_2) \tag{3.2}$$

Equation 3.1 ensures that for all objective functions, their respective values in S_1 are less or equal than the values of S_2. Equation 3.2 ensures that at least one of the values of S_1 from an objective function is smaller than that of S_2.

With this definition of Pareto dominance it is now possible to define a **Pareto non-dominated solution**, which is a solution that is not dominated by any solution within the set of solutions found, it is considered an optimal solution and will be included in the **Pareto front.**

The Pareto front is the selection of the non-dominated solutions found as well as their respective image from the objective space. In this work it will also be used a structure to register the non-dominated solutions, which will be updated constantly through each iteration of the algorithm, as shown in figure 3.1.

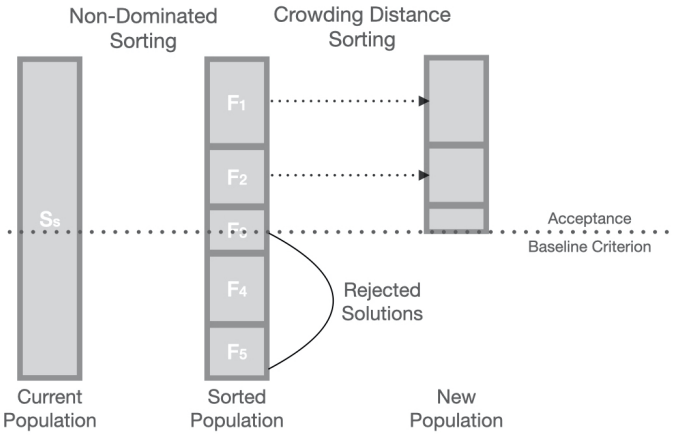


Figure 3.1: Exemplification of the Non-dominated sorting procedure.

Algorithm 1 Pseudo-code of the Non-Dominated Sorting Algorithm

1: **for** each solution i from the Population **do** ▷ Create the first Pareto Front
2: Allocate a set SD_i, to memorize solutions which are dominated by i
3: Define nd_i as the number of solutions that dominate i, initially, as zero
4: **for** each solution i from the Population **do**
5: **if** Solution j is dominated by i **then**
6: Add that solution to SD_i;
7: **if** Solution i is dominated by j **then**
8: Increment nd_i by 1;
9: **if** $nd_i = 0$ **then**
10: Store solutions i in the first Pareto front
11: **Set** k **as 1** ▷ Create other Pareto Fronts until all solutions are sorted
12: **if** k_{th} Pareto Front is **not** empty **then**
13: $k_{th} + 1$ Pareto Front is empty
14: **for** each solution $i \in k_{th}$ Pareto Front **do**
15: **for** each solution $j \in SD_i$ **do**
16: Decrement nd_j by 1
17: **if** $nd_j = 0$ **then**
18: Store solution j in the $k_{th} + 1$ Pareto Front
19: Increment k by 1 and **go to** *line 12*

As the number of solutions is rather diverse, and will vary from iteration to iteration, it is necessary to fractionate the solutions following a dominance level and to sort them by that same level and an additional method, the crowding distance [13]. The process of sorting and fractionating the solutions according to a dominance level d_i, will for now on be defined as **Non-dominated sorting**. The first level will accommodate the non-dominated solutions from the current set, the second level will uphold the solutions that are dominated by the ones from the first level, this process repeats until there are no more solutions to allocate.

In each level, solutions will have different objective values, so the crowding distance measures how close or far an individual solution is from its neighbour. **Crowding distance** for each solution is calculated after the non-dominated sorting is complete, so that for each level, the solutions are sorted in an ascent order. For the first solution S_1 and the last solution S_{last} the value of the crowding distance is not calculated, it will be set as infinity. As for the remaining solutions, the crowding distance cd_i will be calculated according to equation 3.3.

$$cd_i = \begin{cases} \infty \quad \text{if} \quad i = 1 \quad \text{or} \quad i = last \\ \sum_{l=1}^{L} \frac{f_l(s_{i+1}) - f_l(s_{i-1})}{f_l^{max} - f_l^{min}} \quad \text{otherwise} \end{cases} \tag{3.3}$$

Where cd_i is the crowding distance of solution s_i, $f_l(s_i)$ represents the value of the l_{th} objective function for solution s_i, f_l^{max} and f_l^{min} represent, respectively, the maximum and minimum values of that l_{th} objective function.

With this stated, it is easy to conclude that solutions with higher crowding distance will contribute to more diverse group of solutions, thus as a final though a solution S_1 belonging to the same level d_i as the solution S_2 , is superior than the latter one, if its crowding distance is higher ($d_1 = d_2$ and $cd_1 > cd_2$).

The methods described in this section will be used in the algorithm described in the following section,

as a way to organize solutions and to keep higher number of diversely allocated solutions.

3.1.2 Algorithm Steps and Procedure

The implemented algorithm will start its iterative process by linking the employed bees to a food source, that was generated randomly, when initializing the parameters and food sources. In the next step these employed bees will move to a new food source within the neighbourhood of their previous food source, always rating them by the amount of nectar. When these bees complete their process, they share their information with the onlooker bees, who will then move from food source to food source, always evaluating the amount of nectar in each source.

A general procedure of the ABC steps can be given by the following statements:

- Initialization step;
- Repeat
  - Employed Bees Step;
  - Onlooker Bees Step;
  - Scout Bees Step;
  - Update Solutions Set;
- Until it reaches maximum generation.

In the initialization step, the parameters are set, a population is initialized, creating the initial food sources with their respective nectar amounts, then the best ones are selected to continue.

The next step, the employed bees step, these bees will probe to new food sources with the soul objective of acquiring more nectar to their neighbourhood, when a new food source is found, an evaluation is made between the amount of nectar from the old and the new food sources, and the best one remains. Finally, these bees share their information with the next foraging bees, by dancing within their colony.

The next foraging bees start a new step in the algorithm, the onlooker bees step, these foragers will now select the food sources to explore through a tournament selecting one food source per bee. After this selection is done, the onlookers start exploring the neighbourhood of their selected food source, similarly to the employed bees, but with some added depth to their foraging.

The final step is the scout bee step where a random food source is chosen, and enhanced by these bee exploring around its neighbourhood and then if this bee finds a new food source with a similar or better fitness value, it will replace the food source with the worst fitness value from within the current set of food sources by the newly found one.

These moves will be performed until they satisfy the termination criterion, which in this case, is when it reaches the maximum number of generations.

The algorithm described can be further visualized in figure 3.2, which gives a general framework of the process, and the pseudo-code for the implemented method is described in algorithm 2, that presents a more detailed schema of the steps described above. A further explanation of the different steps is provided from sections 3.1.3 to 3.1.12.

Algorithm 2 Pseudo-code of the ABC Algorithm

Initialize parameters:
  Initial Population (PS),
  Maximum Generation ($Genmax$),
  Probability of Tournament (P_{as}),
  Probability of Crossover (P_c),
  Probabilities of rules for generating the Machine Assignment vector (P_{m1} and P_{m2})
  and Probabilities of rules for generating the Operation Sequence vector (P_{o1} and P_{o2})
Initialize population from the procedures to generate machine assignment an operation sequence vectors that compose each solution, and calculate the value of each objective function;
Apply Non-Dominated Sorting to the population;
repeat
  Employed Bee Step:
    Apply exploitation search;
    Update Population, if possible;
  Onlooker Bee Step:
    Apply exploitation search generating population P_t;
  if $Randomnumber < P_{as}$ **then**
      Apply tournament selection to select solution s' from the solutions found in previous step;
  else
      Select solution s' randomly from the solutions found in previous step;
    Apply crossover operators:
  if $Randomnumber < P_c$ **then**
      Two-point crossover for machine assignment;
  else
      Uniform crossover for machine assignment;
    Apply crossover operator for operation sequence;
    Apply local search based on critical path generating population Q_t;
    Apply recombination and selection to P_t and Q_t and update global population with the best Pareto front if possible;
  Scout Bee Step:
    Select, randomly, a solution from population;
    Apply exploitation search, generating a new solution;
    Apply local search to the newly generated solution;
    Replace the worst solution in population (found through crowding distance) by the newly generated solution, if possible;
until iteration = *Genmax*

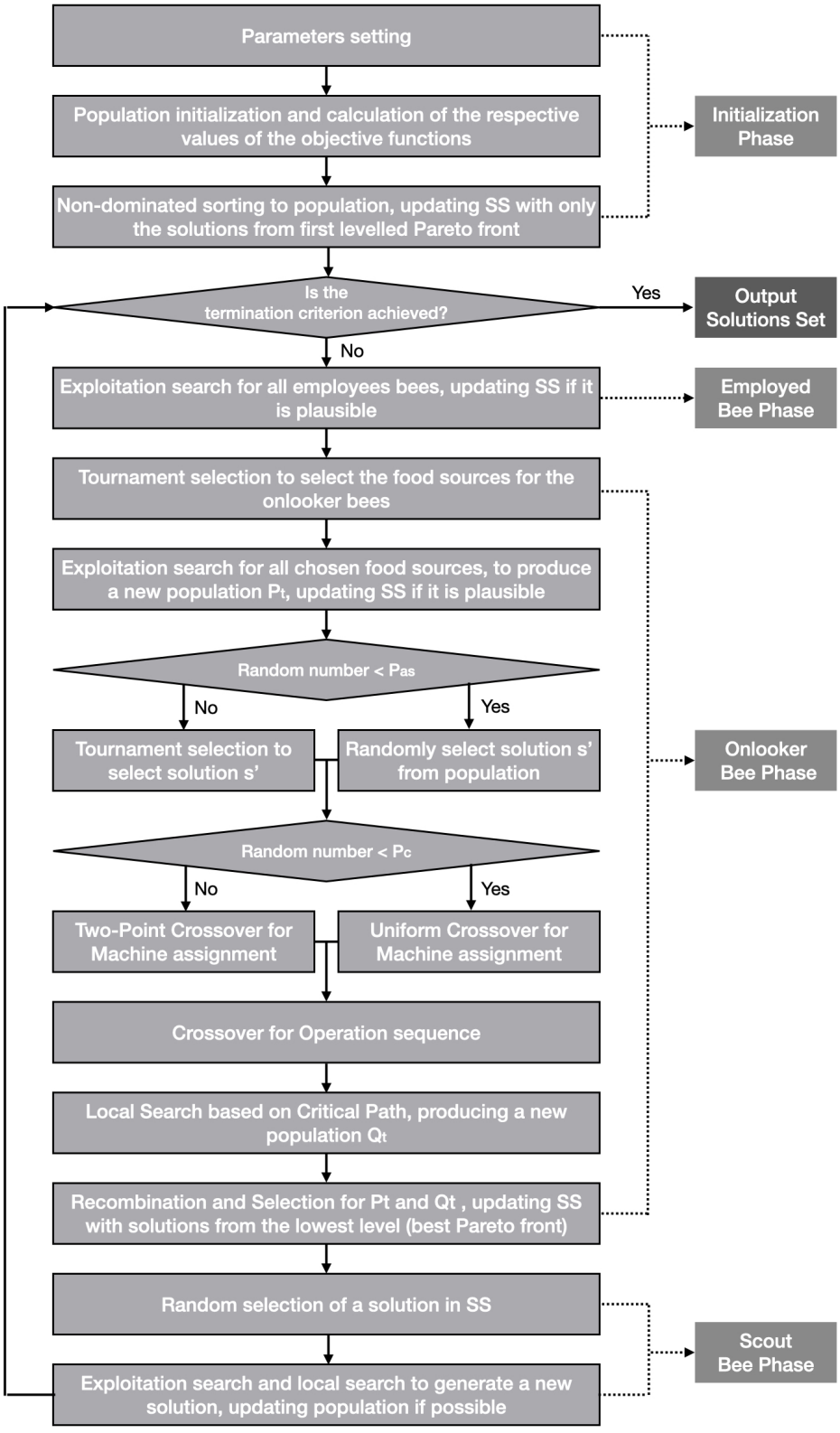


Figure 3.2: Structure of the ABC algorithm implemented.

3.1.3 Solution Representation and Encoding

In the previous section food sources are a representation of solutions, these solutions will be represented by two vectors, a **machine assignment vector** and an **operation sequence vector**, both vectors will represent a feasible solutions. The first one, machine assignment vector, allocates each operation from each job to a machine within its machine set, the second one, operation sequence vector, conceals the sequence of operations for each job.

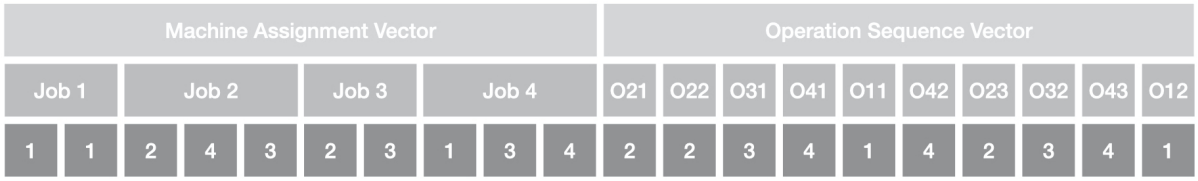


Figure 3.3: Illustration of a coded solution.

As an example, the processing time table 2.1, were there is four jobs, four machines, and a solution is represented in figure 2.1, that graph comes from the decoding of the machine assignment vector and the operation sequence vector that can be observed in figure 3.3. The decoding technique will be further explained in section 3.1.4.

The machine assignment vector [1 1 2 4 3 2 3 1 3 4] depicts that operation one, from job one O_{11} will be processed on machine 1, operation 2, from job one O_{12} will be processed on machine 1, operation one, from job two O_{21} will be processed on machine 2, and this same logic will apply to the remaining operations, so that each integer from the vector will correspond to a machine assigned uninterruptedly for each operation.

The second part of the solution, the operation sequence vector [2 2 3 4 1 4 2 3 4 1] works a little differently, the vector corresponds to the following sequence ($O_{21} \rightarrow O_{22} \rightarrow O_{31} \rightarrow O_{41} \rightarrow O_{11} \rightarrow O_{42} \rightarrow O_{23} \rightarrow O_{32} \rightarrow O_{43} \rightarrow O_{12}$), in words, the operations from each job correspond to their job number, and the i_{th} happening of a job denotes the Ith operation of that same job.

3.1.4 Solution Decoding

To transform the machine assignment vector and the operation sequence vector that constitute the encoded representation of a feasible solution, into a schedule is the action of decoding the solution. To represent the schedule it will be used a Gantt chart, like the one in figure 2.1(b) and it is necessary to establish at what time each operation starts and ends, and also the idle time between operations in the same machine.

As already stated in section 2.4 the precedence constraints must allows uphold, this means that the second operation of a job, can only start after the first one has finished, and if that operation can not start immediately after, due to its machine being currently processing another operation, from another job, it will create idle time on that machine. Considering the makespan objective, a left-shift scheme may be applied in order to shift operations as much to the left as possible, thus compacting the schedule.

3.1.5 Population Initialization

As previously stated, a solution needs two components the machine assignment vector and the operation sequence vector, and so an initial population needs to guarantee diversity and quality of its solutions, another important point being the ability to avoid falling into local optima, and so it was used an hybrid way of generating an initial population set with $P = PS \times m \times n$ solutions, where PS is a predefined parameter related to the size of the initial number of solutions.

To generate initial machine assignment vectors, three rules were used: a **random rule**, a **local minimum processing time rule**, and a **global minimum processing time rule**.

1. **Random rule**: in order to introduce diversity to the vector it generates, it selects a random machine within the machine candidate set for each operation and allocates it to its respective position in the machine assignment vector.

2. **Global minimum processing time rule**: this rule intends to consider the global workload of the solution. Its process is described by:

   (a) Find the minimum processing time of the entire processing time table;

   (b) Fix the minimum value found on the machine assignment vector;

   (c) In the column where it was found the global minimum processing time, add its value to every other entry from that column;

   (d) Go back to point (a) and without considering the fixed rows and repeat the process until all operations are allocated to a machine.

3. **Local minimum processing time rule**: similar to the second rule, the main difference is that it selects the minimum processing time for each row. The steps for this rule are explained as follows:

   (a) From the first operation, represented by the first row, and for each row of the processing time table, find the minimum processing time of the row;

   (b) Fix the minimum value found on the machine assignment vector;

   (c) Add the selected processing time to every other entry from the same column;

   (d) Go back to point (a) until all rows of the processing time table are handled.

The operation sequence initialization process presents three rules, similarly to this first one described, only with different rules. The three rules applied were: a **random rule**, a **most time remaining rule**, and a **most number of operations remaining rule**.

1. **Random rule**: it generates a randomly the sequence of operations on each machine.

2. **Most time remaining rule**: it sequences the jobs in a non-increasing remaining time order, so that the job with the most remaining time will be selected first. It pursues the following method:

   (a) Operations are pre-allocated to the machines, so sum up the processing time of all operations per each job, to calculate the remaining work to be concluded for each job;

(b) Sort the jobs by the amount of work to be done and select the one with the higher value;

(c) Fix the first operation from the selected job on the operation sequence vector;

(d) Subtract the processing time of that fixed operation to the amount of work to be concluded from the selected job;

(e) Go back to point (b) and repeat until all operations are sequenced.

3. **Most number of operations remaining rule**: it selects first the jobs with the most unprocessed operations remaining.

(a) Operations are pre-allocated to the machines, so sum up the number of operations per job, to calculate the remaining operations to be concluded for each job;

(b) Sort the jobs by the amount of operations to be done and select the one with the higher value;

(c) Fix the first operation from the selected job on the operation sequence vector;

(d) Subtract one to the amount of operations to be concluded from the selected job;

(e) Go back to point (b) and repeat until all operations are sequenced.

3.1.6 Exploitation Search in Employed Bee Step

As it was mentioned in section 3.1.2, in order to find new food sources within the neighbourhood, the employed bees perform exploitation search, which consists on examining a limited, yet promising region within the search space. Since a food source consists of two parts, the machine assignment vector and the operation sequence vector, two different exploitation procedure will be implemented for each part of the solution.

Exploitation Search for Machine Assignment

For each solution S_i from the employed bee step the exploitation search for the machine assignment proceeds as described in the next list:

1. Generate an integer I from 1 to O, randomly, where O is the whole number of operations;

2. Select randomly I positions from the machine assignment vector of the solution S_i;

3. For each chosen position, replace the current allocated machine from another within the candidate machine set in order to produce a new solution $S_{i,new}$, anticipating cases where the candidate set is only composed of one machine, in that case, no replacement will be done;

4. After generating $S_{i,new}$, evaluate it. If $S_{i,new}$ dominates S_i, then replace S_i with $S_{i,new}$, if they have different objective values and do not dominate one another, then keep both and update SS.

Exploitation Search for Operation Sequence

For each solution S_i from the employed bee step the exploitation search for the operation sequence proceeds as described in the next list:

1. Select two jobs J_1 and J_2, randomly. Assuming that the number of operations of J_1 is less than or equal to the one of J_2;
2. Fill J_1 within the positions of J_2 from left to right, and repeat the process for J_2 into the positions of J_1, in order to generate the new solution $S_{i,new}$;
3. After generating $S_{i,new}$, evaluate it, if $S_{i,new}$ dominates S_i, then replace S_i with $S_{i,new}$, if they have different objective values and do not dominate one another, then keep both and update SS.

To exemplify this procedure, figure 3.4, demonstrates it for jobs 3 and 4 from the example use in section 3.1.3.

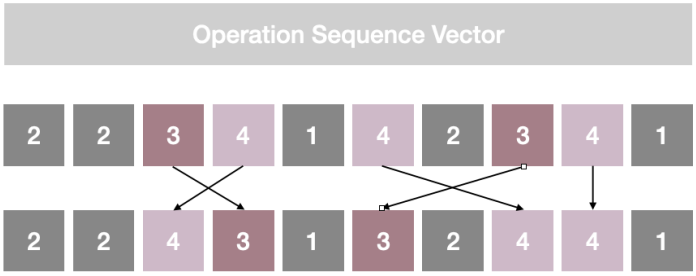


Figure 3.4: Representation of the exploitation search for operation sequence.

These two procedures were applied to the employed bee step to generate new promising food sources.

3.1.7 Tournament Selection for the Onlooker bee step

Due to the difficulty of calculating properly the fitness value that is associated with the probability of an onlooker bee choosing a food source, due to the multi-objective nature of the algorithm, it was implemented a tournament with size three. This method is described as follows:

1. Select three solutions randomly from the employed bee solutions;
2. Sort the solutions according to the Pareto level and crowding distance and choose the best solution to be the food source of the onlooker bee.

3.1.8 Exploitation search in Onlooker Bee Step

With all the onlooker bees having a selected food source from the tournament selection described in section 3.1.7, every onlooker bee will perform the techniques for machine assignment vector and operation sequence vector described in section 3.1.6, to generate new neighbouring solutions and the

better solutions found will update the population to establish a new population P_t, to be used by the crossover operators described in the next section.

3.1.9 Crossover Operators

The crossover operators were designed to enhance the exploring capabilities of the onlooker bees, or in other words, to share information in order to find more promising food sources, better solutions. Since the solution is composed by the machine assignment vector and the operation sequence vector, and similarly to section 3.1.6 it was designed different crossover operators for each of this parts.

Associated with this, there is the probability P_{as} were for each solution S_i from the current solution set it will be attributed another solution S' randomly if it has a probability P_{as} or higher, alternatively if it has a probability $(1-Pas)$, then it will attribute a solution by tournament selection with the same process as described in section 3.1.7.

Crossover Operators for Machine Assignment

Starting by first describing the crossover operators for machine assignment, two operators were used, associated with another probability P_c, were it is used the two-point crossover operator and with probability $(1 - P_c)$ an uniform crossover operator is used.

To describe the **two-point crossover** it is first necessary to introduce some comments on parent, child and partial chromosomes. The parent chromosomes will be identified as P_A and P_B, the child chromosomes as C_A and C_B, in order to have partial chromosomes two crossover points are needed which will be denoted as cp_1 and cp_2, and finally, the partial chromosomes between cp_1 and cp_2 from the parent chromosomes are A_p and B_p, respectively the partial chromosomes from the child chromosomes correspond to A_c and B_c. The two-point crossover can be described as follows:

1. Select P_A and P_B as parent chromosomes and create the empty C_A and C_B child chromosomes;
2. Select randomly, the two crossover cp_1 and cp_2, and associate the partial chromosomes A_p and B_p;
3. Attribute all genes to C_A and C_B except for the ones belonging to the partial chromosomes;
4. Perform the following steps:
   (a) If exists the same gene, the gene of A_p as compared to the gene of B_p, then exchange mutual position information and move to A_c and B_c, respectively;
   (b) Save the order information and move the remaining genes to the child chromosomes.

The **uniform crossover** procedure for solutions S_i and S' will be described in the following list.

1. Generate a binary string uniformly that composed of 0 and 1, with the same length of the machine assignment vector;

Figure 3.5: Example of the two-point crossover for machine assignment.

2. The new solution will assume the elements which are 1 from solution S' and the elements which are 0 from solution S_i.

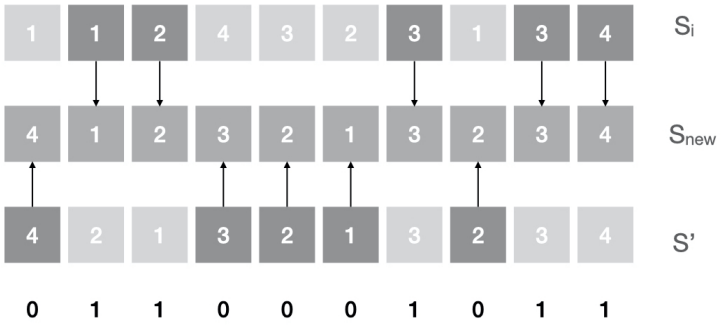


Figure 3.6: Example of the uniform crossover for machine assignment.

Crossover Operator for Operation Sequence

Finally, for the operation sequence, the crossover operator implemented is a modified precedence operation crossover operator (**MPOX**) according to the precedence preserving order-based crossover (**POX**), the method for solutions S_i and S' can be described as follows:

1. Generate a random subset of all jobs;

2. The new solutions will inherit the element of S_i at the position if that element belongs to the subset generated, if not, it inherits the elements of S' that do not belong to the subset. The element inheritance is performed from left to right.

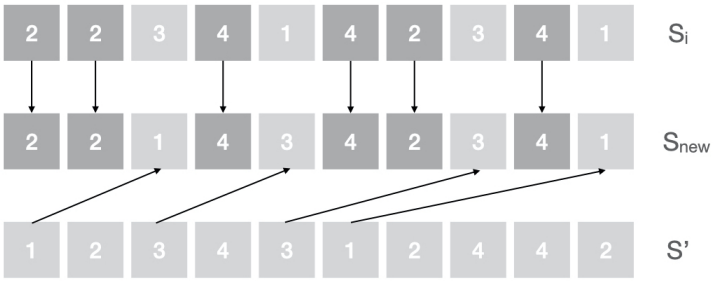


Figure 3.7: Illustration of the crossover for operation sequence.

3.1.10 Local Search Based on the Critical Path

A step towards reaching an optimal solution requires to promote the intensification of the exploration capacity of the algorithm, thus a local search method was implemented to the onlooker bees step. To describe the local search based on critical path it is first needed to remind the notations and the models to represent a feasible solution stated in section 2.4. Additionally to the previously stated notations and definitions of the graph model described in section 2.4.1, some further explanation will be given in the following lines.

Starting by describing the process to make a graph G, to better understand which operations belong to each job, there will be dummy nodes such as, node 0 and node $E = N + 1$, where N is total number of operations. These nodes will have no processing time, the starting time of node 0 will be 0, and the completion time of node E is the makespan of the schedule.

The processing time for each operation will be expressed in each respective node. The earliest starting time of operation O_{ij}, expressed by equation 3.4, will be designated as ES_{ij} and will be defined as the maximum time value between the earliest completion time in the operation that precedes operation j in the job i, represented as $PO^J_{ij} = O_{i,j-1}$ and the earliest completion time of the previously executed operation on the same machine k, represented by PO^M_{ij}.

$$ES_{ij} = \max\left(EC(PO^J_{ij}), EC(PO^M_{ij})\right) \tag{3.4}$$

The earliest completion time of operation O_{ij}, expressed in equation 3.5, is the sum between the earliest starting time of that same operation and its processing time on machine k.

$$EC_{ij} = ES_{ij} + PT_{ijk} \tag{3.5}$$

To compute the earliest starting and completion times, the process will perform iteratively from node 0 until node E, in other words, from left to right. Now, to compute the latest starting and completion times, the process will be done backwards, starting from node E, ending in node 0.

The latest starting time of operation O_{ij} without delaying the makespan, given by equation 3.6, designated as LS_{ij} is given by the subtraction between the latest completion time of that same operation and its processing time machine k.

$$LS_{ij} = LC_{ij} - PT_{ijk} \tag{3.6}$$

The latest completion time of operation O_{ij} without delaying the makespan , given by equation 3.7, designated as LC_{ij} is given by the minimum time value between the latest starting time of the operation that follows j within job i, designated as $FO^J_{ij} = O_{i,j+1}$, and the latest starting time of the operation that follows O_ij on machine k, designated as FO^M_{ij}.

$$LC_{ij} = \min\left(LC(FO^J_{ij}), LC(FO^M_{ij})\right) \tag{3.7}$$

It is also important to note that for the dummy nodes these values will remained unaltered. Also, the

makespan will be given by the latest completion time of the ending node E.

The total slack of each operation translates into the time amount as to which an operation can be delayed without delaying the makespan, intuitively, if an operation has zero slack, it means that if that operation delays, then the makespan will delay as well, thus making it a critical operation. The total slack of an operation is given by subtracting the earliest starting time to the latest starting time, as expressed in equation 3.8.

$$TS_{ij} = LS_{ij} - ES_{ij} \tag{3.8}$$

Once a critical path is found, the critical operations will be moved towards minimizing the objective functions. Referencing the theorem in [35], in the schedule represented by graph G, if a new schedule represented by graph G' obtained by moving an operation $O_{ij} \notin \chi(G)$ in G, then $C_{max}(G) \leq C_{max}(G')$. This means that minimizing the makespan, and consequently total and maximum workload can only be achieved through moving the critical operations exclusively.

An operation can only be moved and inserted into another position if it satisfies the following inequality:

$$\max\left(EC(PO^{M}_{(G_i^-,v)}), EC(PO^{J}_{(G_i^-,co_i)})\right) + PT_{co_i,k} \leq \min\left(LS(FO^{M}_{(G_i^-,v)}), EC(FO^{J}_{(G_i^-,co_i)}\right) \tag{3.9}$$

Where v is the operation in G^- that will succeed operation co_i in the newly appointed position. This relates to another theorem from [35] which states that a schedule G' is obtained by inserting co_i into a position located before operation v on machine M_k in G_i^- under the circumstance that satisfies 3.9. If $C_{max}(G') = C_{max}(G)$ then co_i is not the critical operation in G'. This theorem avoids falling into a cyclic behaviour by appointing co_i as a non-critical operations in the next iteration. Now, to examine which positions will be available to insert co_i into, before v, there is the need of Γ, which is the set of positions before all the operations of δ_k excluding Ψ_k and after all the operations from Ψ_k excluding δ_k. This translates that in G_i^-, the schedule obtained by inserting the critical operation co_i into γ in Γ_k is always feasible and that exists a position in Γ_k that minimizes the makespan. This means that only the positions in this set are checked when trying to move the critical operation on machine M_k, as it can be seen from algorithm 3.

Now, $\psi_k(G)$ consists of the possible machines in which each critical operation can be performed sorted to minimize total and maximum workload. Where δ_t and δ_m are used as metrics to consider respectively the total and maximum workload, defined by equation 3.10:

$$\begin{cases} \delta_t(co_i \mapsto M_k) = PT_{co_i,k} - PT(co_i, k \text{ from } co_i \in G) \\ \delta_m(co_i \mapsto M_k) = W_k(G) + PT_{co_i,k} \end{cases} \tag{3.10}$$

This condition ensures that the attempts to minimize the makespan, also guarantee that it will minimize the total and maximum workload objectives, by sorting this ψ_k according to an ascent order of δ_t

and δ_m.

The procedure explained above is detailed in algorithm 4, and after it generates a new solution, this procedure evaluates its perform with respect to the old solution, and similarly to what has been described so far, if the solution dominates the old one it will update the population, and if none dominates one another, both will be kept.

Algorithm 3 Pseudo-code of the inserting operations procedure in local search.

```
Get the set of Γ_k on machine M_k
for each position γ in Γ_k do
    if γ satisfies equality 3.9 then
        Insert co_i into γ in G_i^- return true
return false
```

Algorithm 4 Pseudo-code of the moving operations procedure in local search.

```
Get the set of nc critical operations CO = {co_1, co_2, ..., co_nc}
Exclude them from graph G, generating nc new graphs G^-
Sort ψ(G) according to δ_t and δ_m
for each operation co_i executed on machine M_k from the candidate machine set do
    if G_i^- = ∅ then
        Store a copy of G^- to G* and exclude co_i from G*
        Assume G* as G_i^-
    if Inserting an operation on machine is true then
        Assume G_i^- as the new G' return G'
return ∅
```

3.1.11 Recombination and Selection

A final method in the onlooker bee step that will decide the population to proceed to the scout bee step is the recombination and selection approach.

1. Starts by combining populations P_t and Q_t together, and sorting them through the non-dominated sorting procedure;

2. Selects the solutions from the best front to update the SS, and if the number of solutions from that front exceeds the initial population size P, the solutions from that front will be sorted in a descending order of crowding distance and only the ones with a higher number will update SS.

3.1.12 Scout Bee Step

The scout bee step is used to randomly generate a new food source that replaces the worst solution found. Global exploration will be applied to the solution to intensify the population diversity and improve results, this step is described next.

1. Select a random solution from SS;

2. Perform exploitation search for the machine assignment on the selected solution, generating a new temporary solution S_{temp};

3. Perform local search based on the critical path to the new temporary solution S_{temp} and generate S_{scout};

4. Evaluate and compare the new solution generated with the worst solution from the SS and replace it with S_{scout} if it is better.

3.2 Parameter Tuning

As it was mentioned in section 3.1.2, shown by the general overview in figure 3.2 and by the algorithm 2, the algorithm implemented to solve the MOFJSSP needs to initialize promptly a set of key parameters.

This set of 8 parameters is composed by the parameter related to the maximum generation Gen_{max} the parameter related to the initial population size PS, the probability of the machine assignment to be executed by rule 1, and respectively the rule 3, the probability of the operation sequence to be executed by rule 1, and respectively the rule 3, the probability for solutions to participate in the tournament selection and finally the probability to perform the two-point crossover.

Naturally, such a number of parameters to be set can have different methodologies to tune and set these parameters. And since these parameters will affect the performance of the algorithm implemented, it was implemented a Bayesian optimization algorithm for tuning the parameters set.

3.2.1 Bayesian Optimization

The Bayesian Optimization (BO) algorithm was chosen for being a state-of-art solution in circumstances with a finite set of parameters, that will directly reflect on the performance of an algorithm whose choice of parameters can not be done intuitively, that is a computationally expensive system and through other methods such as a grid search would consume a tremendous amount of time and could leave out of testing lots of parameter set combinations.

The main advantage of choosing Bayesian optimization is that the iterative process of retrieving the best combination of parameters is done through an unbiased utility function, that will maximize the algorithms performance when choosing an new set of parameters to test, resulting in a lot of computational time saved and reducing the number of tests to be done. The Bayesian optimization, as shown in [63], starts by creating a surrogate model, SM, of the system $f(\mathbf{x})$, that is subjected to an optimization of the parameters combinations, described as $\mathbf{x}$, that have been tested, and their respective outputs $y = f(\mathbf{x})$. Equation 3.11 describes the optimization problem fittingly.

$$\mathbf{x}^* = \underset{\mathbf{x} \in \Re^p}{\arg\min} f(\mathbf{x}) \tag{3.11}$$

Where p is the number of parameters that minimizes the objective function. The surrogate model is created through an unbiased stochastic procedure, defined as the Gaussian process.

The Gaussian process is defined by two guidelines, a mean function $m(\mathbf{x})$ and a kernel function $k(\mathbf{x}, \mathbf{x}')$. The kernel function, gives information about the correlation between the objective function

values of the two points considered $(\mathbf{x}, \mathbf{x}')$, and considering the various options to choose, the one that best suits for Bayesian algorithms favours to be the Mattern 5/2 Kernel function, according to [64], described in equation 3.12

$$k_{M5/2} = \theta_0^2 \left(1 + \sqrt{5} \times r(\mathbf{x}, \mathbf{x}') + \frac{5}{3} \times r(\mathbf{x}, \mathbf{x}')^2\right) + e^{-\sqrt{5} \times r(\mathbf{x}, \mathbf{x}')} + \theta_n \tag{3.12}$$

where,

$$r(\mathbf{x}, \mathbf{x}')^2 = \sum_{p=1}^{n_p} \left[\left(\frac{x_p - x'_p}{\theta_p}\right)^2\right] \tag{3.13}$$

With θ_p as the ratio parameter between length and scale for each dimension p, the ratio that expresses the relevance of each parameter in the output, θ_0 and θ_n are respectively, the amplitude and noise parameters chosen.

After creating the surrogate model, it will be analysed by an acquisition function, to determine the best combination to test next. Amongst the various acquisition functions to choose from, according to [65], the one that shows better performance, is the expected-improvement acquisition function, μ_{EI}. The main advantage of this acquisition function is its ability to control the exploration versus exploitation ratio for new solutions, from using this ratio as an input parameter. To better describe this process, algorithm 5 gives a summarized explanation. The results from this parameter optimization were used to fuel the algorithm presented in section 3.1, and are further elaborated in section 4.1.

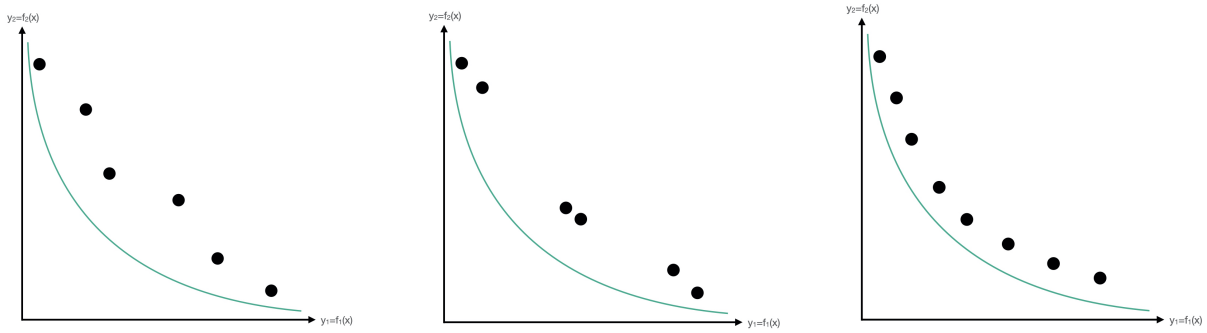
Algorithm 5 Pseudo-code of the Bayesian optimization.

Input: Maximum number of iterations
Output: Optimal parameter set
$\mathbf{x_1} \leftarrow rand(x)$ ▷ Choose a starting point, a random parameter set
$y_1 = f(\mathbf{x_1})$ ▷ Compute the output and store parameter input and output observation
$Obs = [\mathbf{x_1}; y_1]$
$SM \leftarrow Obs$ ▷ Create the surrogate model
$i = 2$
while $i \leq$ Maximum number of iterations **do**
    $\mathbf{x_i} = \arg\max a(x|SM)$ ▷ Maximize the acquisition function to choose the new parameter set
    $y_i = f(\mathbf{x_i})$
    $Obs = append(Obs, [\mathbf{x_i}; y_i])$ ▷ Compute the output and append the observation to Obs
    $SM \leftarrow Obs$ ▷ Update the surrogate model
return $\mathbf{x_i}$ that corresponds to the minimum y_i

3.2.2 Performance Metrics to Evaluate Solutions

Considering the implemented algorithm, the Pareto front of three objective functions, will be a 3D Pareto front, but, for the sake of simplicity, the explanation of the metrics used to evaluate, and to find the best set of parameters through the Bayesian optimization, will be done by a bi-objective generic Pareto front, like shown in figure 3.8. In figure 3.8(a) the results are displayed irregularly, where the closeness to the Pareto front is not constant, and some results appear more distant than others, this shows solutions that did not converge well. As for the middle one, figure 3.8(b) clearly shows an irregular, non-uniform distribution of solutions, that are actually closer to the Pareto front, when compared to the left figure, but

with solutions that are more spread out. Finally, in figure 3.8(c) shows an example that is not only closer to the Pareto front, but its solutions are evenly distributed throughout the objective space.



(a) Bad convergence (b) Bad diversity (c) Good convergence and diversity

Figure 3.8: Three examples of solutions for a bi-objective problem, where the Pareto front is represented by the green line whereas the the solutions are represented as black circles.

In conclusion, to completely analyse the performance of a multi-objective algorithm, it is not only necessary to analyse its convergence, but also the diversity within the solutions. There exists several indicators expressed in table 3.1, showing what they measure, if only convergence, only diversity, or both. To analyse the behaviour of the algorithm, and inspired by the metrics from [66], [67] and [68], the metrics chosen in this work were the Mean Ideal Distance and the Hypervolume.

Metric	What it measures
DM Spacing Spread	Diversity
Set Convergence Generational Distance Error Rotation	Convergence
Hypervolume Mean Ideal Distance G-metric	Convergence and Diversity

Table 3.1: Performance indicators of multi-objective algorithms.

Hypervolume

This metric can measure both convergence and diversity in the case of a bi-objective algorithm, it calculates the area created by the non-dominated front in the objective space, provided a reference point, where in the case of this work, for three objective functions, it calculates the volume. The metric result is given by summing up the rectangular areas/ parallelepiped volumes bounded by some reference point, as seen in [69]. As a result, a simple representation of the hypervolume metric is shown in equation 3.14, and from that it can easily be concluded that the computational complexity increases greatly with the increasing number of objective functions. If a result shows a high value of hypervolume, is consequently

a converged and well-spread set of solutions.

$$HV = \sum_{s_i \in SS} (V_{s_i}) \tag{3.14}$$

Where, V_{s_i} is the hypervolume of solution s_i with respect to the reference point V_R. To exemplify, in figure 3.9, two solutions are presented, the one that occupies more area is similarly the one that is closer to the Pareto front, as well as more well-spread, given this, solution A has a higher hypervolume value than solution B, thus A is a better solution set than B .

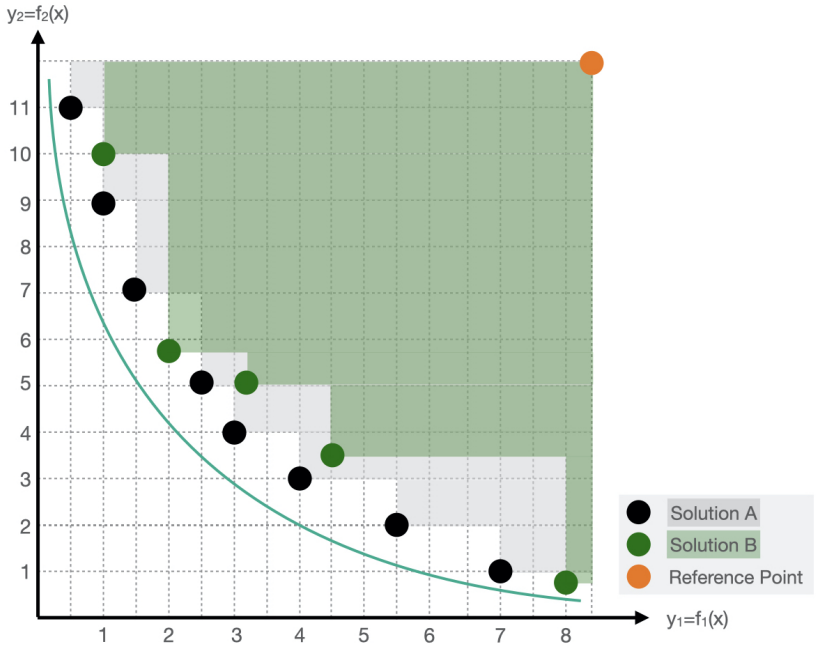


Figure 3.9: Exemplification of the hypervolume calculation for a bi-objective problem.

As it can be observed by the figure above, the reference point is a sensitive point for choosing the best solution set, thus needs a careful choice as shown by [69].

Mean Ideal Distance

Similarly to the hypervolume metric, so does the Mean Ideal Distance [67] evaluate a solutions set quality by measuring both convergence and diversity. This metric is given by the average distance between non-dominated solutions and the origin point, as shown by equation 3.15, where $f_l(s_i)$ is the value of the objective function l for solution s_i, $|SS|$ is the number of solutions from the population.

$$MID = \frac{\sum_{s_i \in SS} \left(\sqrt{\sum_{l \in L} \left(f_l(s_i)^2 \right)} \right)}{|SS|} \tag{3.15}$$

Chapter 4

Results and Discussion

In this section, results from the methods proposed in Chapter 3 are presented. First, the ABC algorithm from section 3.1 is tuned with the algorithm described in section 3.2 and tested with FJSSP benchmarks. Finally, the results obtained are presented and discussed.

The ABC as well as the Bayesian optimization algorithms were implemented using *Matlab 2018b*. The implemented algorithm was tested on the ten instances of the Brandimarte dataset from [70] and on the five Kacem instances from [71] and [30], in order to verify and validate its performance for solving the multi-objective flexible job shop scheduling problem.

The first dataset, the Brandimarte, is composed of instances that range from 10 to 20 jobs, 4 to 15 machines, with instances that contain jobs with 3 to 15 operations per job and processing times varying from 1 to 20, as it can be seen from table 4.1, with M_{ik} being the average maximum number of machines that an operation can be performed at.

Instance	Number of Jobs	Number of Machines	Number of Operations per Job	$\lvert M_{ik} \rvert$	Processing Time
mk01	10	6	5 to 7	3	1 to 7
mk02	10	6	5 to 7	6	1 to 7
mk03	15	8	10	5	1 to 20
mk04	15	8	3 to 10	3	1 to 10
mk05	15	4	3 to 10	2	5 to 10
mk06	10	15	15	5	1 to 10
mk07	20	5	5	5	1 to 20
mk08	20	10	5 to 15	2	5 to 20
mk09	20	15	10 to 15	5	5 to 20
mk10	20	15	10 to 15	5	5 to 20

Table 4.1: Attributes of the Brandimarte dataset instances.

The second dataset used to evaluate the quality of the algorithm was the Kacem dataset, with five instances, which range from 4 to 15 jobs, 5 to 10 machines, 2 to 4 operations per job, an M_{ik} that ranges from 5 to 10, and processing times that go from 1 time unit to 99. From table 4.2 it is possible to infer that each instance increases complexity from the previous, both by number of jobs and number of machines.

To obtain a better performance of the ABC, the parameters were tuned using Bayesian optimization as described in section 3.2. The next section describes the tuning process used. The rest of this chapter shows the results obtained for both datasets and provides a thorough discussion of these results.

Instance	Number of Jobs	Number of Machines	Number of Operations per Job	$\|M_{ik}\|$	Processing Time
kacem1 (4×5)	4	5	2 to 4	5	1 to 54
kacem2 (8×8)	8	8	3 to 4	6 to 8	1 to 12
kacem2 (10×7)	10	7	2 to 3	7	1 to 99
kacem3 (10×10)	10	10	3	10	1 to 20
kacem4 (15×10)	15	10	2 to 4	10	1 to 56

Table 4.2: Attributes of the Kacem dataset instances.

4.1 Parameter Tuning

As already mentioned above, the parameter tuning was done using the Bayesian optimization algorithm presented in Chapter 3. The algorithm was implemented with the aid of the *Matlab* function *bayesopt*.

It uses a specific parameter, the exploration versus exploitation ratio of the acquisition function, which was set to $\epsilon = 0.5$, to establish equal importance on exploration and exploitation. Also, the acquisition function used was the expected improvement, μ_{EI}, and the maximum number of iterations defined was 30. The search for the optimal set of parameters was done on two metrics, the Mean Ideal Distance and the Hypervolume. For each metric and for each benchmarked instance, the algorithm was executed 5 times to guarantee consistency within the obtain sets of parameters.

Figure 4.1 shows box plots for the different results obtained for benchmark instance mk01, to exemplify the process behind choosing the right set of parameters. It shows that between metrics, the difference within the parameters set is very little and the results from both measures are quite similar. The parameters were obtained using the median values from both metrics.

The parameters obtained from this process are presented in table 4.3 for the Brandimarte instances, and table 4.4 for the Kacem instances. Further detail and discussion of the different parameters used is presented in the following sections.

4.1.1 Generating the initial population

The ABC algorithm belongs to the Swarm Intelligence algorithms and needs an initial population of solutions to start the evolution of the colony. The initial solutions need to have relatively good quality and diversity, as they shape starting neighbourhood, the solution space that will be explored, and are a key factor to avoid the solution falling in local optima.

In the algorithm implemented, the solution is coded with two vectors, the machine assignment and the sequence of operations. The initial population of bees is generated with the aid of the methods

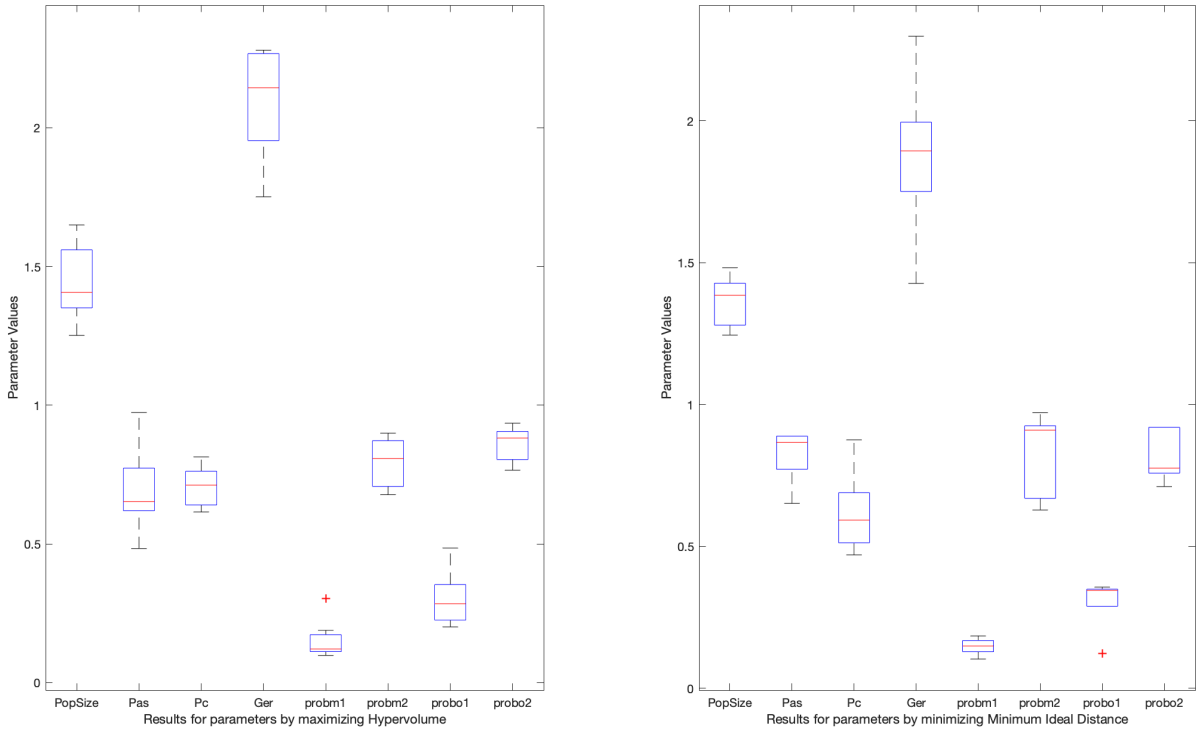


Figure 4.1: Results from several runs of the Bayesian Optimization for instance mk01.

Instance	Population Size	Generation	P_c	P_{as}	$p_{m,rule1}$	$p_{m,rule3}$	$p_{o,rule1}$	$p_{o,rule3}$
mk01	1.458	2.073	0.744	0.591	0.168	0.859	0.303	0.831
mk02	1.945	3.630	0.469	0.638	0.187	0.786	0.271	0.720
mk03	0.988	3.099	0.515	0.579	0.114	0.844	0.176	0.801
mk04	1.230	1.424	0.683	0.750	0.160	0.744	0.243	0.698
mk05	0.999	2.306	0.624	0.638	0.177	0.804	0.169	0.711
mk06	1.423	1.756	0.716	0.630	0.168	0.742	0.278	0.850
mk07	1.030	3.399	0.579	0.574	0.197	0.731	0.256	0.679
mk08	0.874	3.709	0.652	0.663	0.128	0.872	0.262	0.836
mk09	0.830	1.422	0.538	0.599	0.217	0.786	0.219	0.869
mk10	0.865	1.684	0.625	0.648	0.183	0.794	0.203	0.857

Table 4.3: Set of parameters for each instance of the Brandimarte dataset tested.

Instance	Population Size	Generation	P_c	P_{as}	$p_{m,rule1}$	$p_{m,rule3}$	$p_{o,rule1}$	$p_{o,rule3}$
kacem1 (4×5)	1.900	2.445	0.629	0.651	0.215	0.776	0.355	0.848
kacem2 (8×8)	1.365	1.555	0.600	0.598	0.149	0.824	0.201	0.767
kacem3 (10×7)	1.365	2.381	0.566	0.640	0.229	0.854	0.165	0.731
kacem4 (10×10)	1.547	1.730	0.595	0.597	0.185	0.789	0.157	0.812
kacem5 (15×10)	1.131	2.565	0.632	0.634	0.179	0.829	0.302	0.838

Table 4.4: Set of parameters for each instance of the Kacem dataset tested.

detailed in section 3.1.5. These methods are composed of three rules for both the machine assignment vector and the operation sequence vector.

The three rules that generate the initial machine assignment vectors give different solutions. Two

of these rules, the *local minimum processing time* and the *global minimum processing time* produce different results, but slightly similar, meaning that these two rules do not introduce much variability to the solutions they generate. A third one, the *random rule* is used for this exact purpose, to introduce variability to the population.

As for the operation sequence part of the population, also three rules are employed. The *random rule* selects jobs to be sequenced randomly and sets them on the first available entry. The *most time remaining rule*, appoints jobs giving priority to the ones that have a higher processing time to do, hence more work to be done. Lastly, the *most number of operations remaining rule* appoints jobs according to the operations yet to be done.

Identically to the global minimum processing time and the local minimum processing time rules, the most time remaining and most numbers of operations rules create good solutions with low variability. The random rule was also implemented to introduce more variability to the generated operation sequences.

These rules were implemented with a probability of occurrence associated to them. These probabilities were determined by the parameter tuning with Bayesian optimization in order to tune, along with the other parameters, the creation of an initial population.

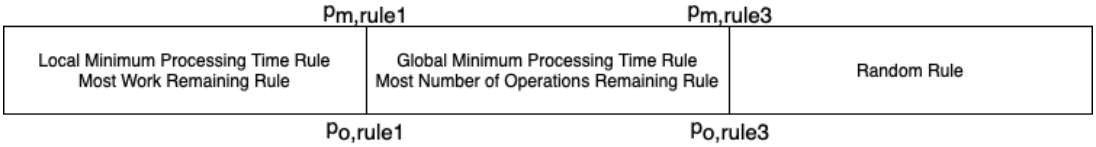


Figure 4.2: Distribution of the probability of occurrence associated with the rules to generate the initial population.

Figure 4.2 illustrates how the probability of occurrence for each rule was implemented. For each part of the solution there are two probabilities of occurrence associated. The rules were configured according to the following probabilities:

- A solution being generated using local minimum processing time rule: $p_{m,rule1}$;
- A solution being generated using global minimum processing time rule: $p_{m,rule3} - p_{m,rule1}$;
- A solution being generated using random assignment rule: $p_{m,rule3}$;
- A solution being generated using most work remaining rule: $p_{o,rule1}$;
- A solution being generated using most number of operations remaining rule: $p_{o,rule3} - p_{o,rule1}$;
- A solution being generated using random sequencing rule: $p_{o,rule3}$.

Tables 4.3 and 4.4 both show that the difference between the values for these probabilities of occurrence for each benchmark instance is significantly small. Figure 4.3 illustrates this, showing the box plot for these values from all benchmarks tested. This figure shows that these probabilities of occurrence do not vary significantly depending on the benchmark used, hence do not vary significantly when the static environment changes. Resulting from this observation, it may be implied that fixed values obtained from the tuning process can be used. Table 4.5 shows the standard deviations obtained for each of these

parameters, and since these values are substantially lower than the median value, the final parameters were set by the given median values presented.

Parameter	Standard Deviation	Median Value
$p_{m,rule1}$	0.0316	0.1790
$p_{m,rule3}$	0.0441	0.7940
$p_{o,rule1}$	0.0592	0.2430
$p_{o,rule3}$	0.0656	0.8120

Table 4.5: Standard deviation and median values for the parameters that generate the initial solutions.

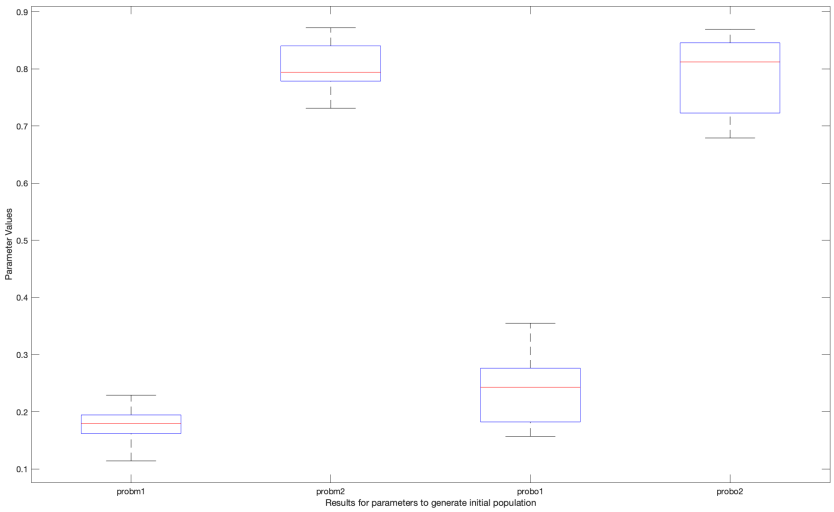


Figure 4.3: Distribution of the probability of occurrence associated with the rules to generate the initial population.

4.1.2 Population size

For evolutionary algorithms, the initial population size plays an important role in determining the performance of the algorithm. If the chosen initial population is small, the algorithm might converge too quickly and not provide a decent exploration of the search space, and if the initial population is too large the algorithm will take a lot of computational effort in order to converge.

Regarding this specific parameter, the unbiased Bayesian optimization algorithm provided the results for tuning the parameters, shown in tables 4.3 and 4.4 in the column that corresponds to Population Size. The initial population is calculated as a product of this parameter by the number of machines, m, and the number of jobs, n, $\mathbf{InitialPopulation} = \mathbf{PopulationSize} \times m \times n$, where the initial population is the initial number of solutions generated by the algorithm. The initial population size for each instance tested is shown in table 4.6.

Instance	Population Size	Initial Population	Generation	Maximum Generation
mk01	1.458	88	2.073	125
mk02	1.945	117	3.630	218
mk03	0.988	119	3.099	372
mk04	1.230	148	1.424	171
mk05	0.999	60	2.306	138
mk06	1.423	213	1.756	263
mk07	1.030	103	3.399	340
mk08	0.874	175	3.709	742
mk09	0.830	249	1.422	426
mk10	0.865	260	1.684	505
kacem1	1.900	38	2.445	49
kacem2	1.365	88	1.555	100
kacem3	1.365	96	2.381	167
kacem4	1.547	155	1.730	173
kacem5	1.131	170	2.565	285

Table 4.6: Values for the initial population of solutions.

4.1.3 Tournament selection parameter

The tournament selection finds the most suitable solution for another individual to perform the crossover operators, as opposed to this operation the algorithm will assign a random individual to apply the same operators. The details of this procedure are explained in section 3.1.7.

Similarly to the parameters used to generate initial solutions, the tournament selection also has a probability of occurrence. The values for this parameter were tuned using the Bayesian optimization. The considerations were to maximize the hypervolume metric as well as minimize the mean ideal distance metric, and the parameter in question is P_{as}.

The probability of occurrence for tournament selection or random selection are expressed as follows:

- Probability of applying tournament selection to the population: P_{as}
- Probability of applying random selection to the population: $(1 - P_{as})$

By observing the column concerning P_{as} in tables 4.3 and 4.4, it is possible to derive a standard deviation for this particular parameter of 0.0438, which is less than 5%. This allows to conclude that, once tuned, this parameter should not change with the different instances used. And so, it is possible to set P_as as 0.634.

4.1.4 Crossover operators parameter

To improve global exploration around the multiple neighbourhoods, machine assignment and operation sequence crossover operators were implemented. For the operation sequence only a modified precedence operation crossover (MPOX) was implemented. For machine assignment two crossover operators were implemented, uniform crossover and two-point crossover. All crossover operators are fully

detailed in section 3.1.9.

For the operation sequence crossover operator, since only one was implemented, the probability of happening was set at 1. As for the machine assignment crossover operators, the probability of occurrence was tuned using the Bayesian optimization algorithm, with considerations as to maximize the hypervolume metric, as well as, minimize the mean ideal distance metric. The parameter which concerns the crossover operators is P_c. As for the crossover operators, the probability of occurrence for each is expressed as follows:

- Probability of applying uniform crossover to the machine assignment vector: $(1 - P_c)$
- Probability of applying two-point crossover to the machine assignment vector: P_c

As for the results obtained, seen in tables 4.3 and 4.4, this particular parameter appears to have a standard deviation of 0.0731. This is considerably smaller than the median value, which allows to conclude that once tuned, this parameter is not affected by the instance used. And so, it is possible to set P_c as 0.624.

4.1.5 Termination criterion

The termination criterion used was a maximum number of iterations, also defined as the maximum number of generations due to the evolutionary nature of the algorithm. Similarly to what happens with the Population Size parameter, also the maximum number of iterations plays an important role in the algorithms convergence and diversity. If the number of generations is assumed too low, then prosperous solutions might not be achieved.

Again, using the Bayesian optimization algorithm considering the metrics used, the values obtained for this parameter are presented in tables 4.3 and 4.4, with respect to the parameter Generation. Furthermore, the maximum number of generations depends on the number of machines, m, and the number of jobs, n, and is calculated as $Max_{generations} = Generation \times m \times n$. These values are presented in table 4.6, under the column Maximum Generation.

4.2 Results comparison

To evaluate the performance of the ABC algorithm implemented, two benchmarked datasets were used and the results obtained for each were compared with several other algorithms. The algorithms chosen to compare the proposed implementation with involve algorithms with multi-criteria approaches for the multi-objective optimization and Pareto optimality approaches. To verify the superiority of results from the latter one with respect to the first type of approaches. And also compare the proposed algorithm with other Pareto optimality approaches.

The results were obtained after 20 runs of each instance from each dataset, and by selecting the fittest solutions. The following sections provide results comparison for each dataset.

Figure 4.4 shows the non-dominated solutions obtained in all iterations. The more dispersed points represent the first solutions obtained by the algorithm, whereas the solutions highlighted show the best solutions found.

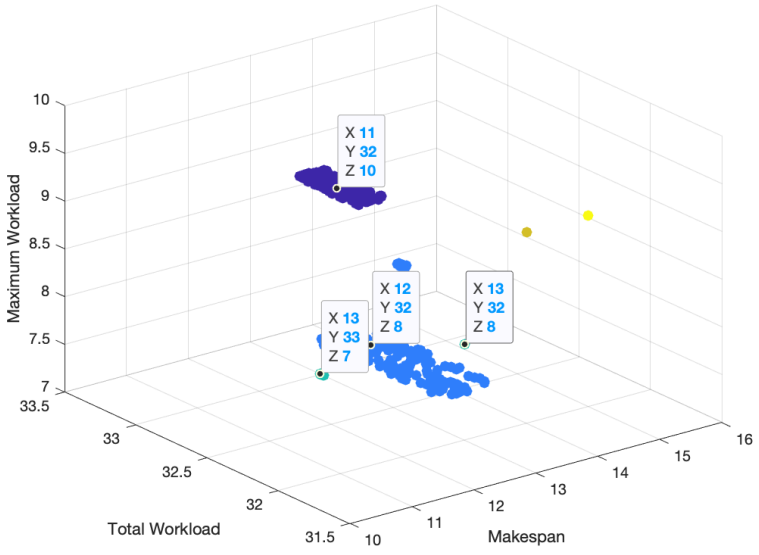
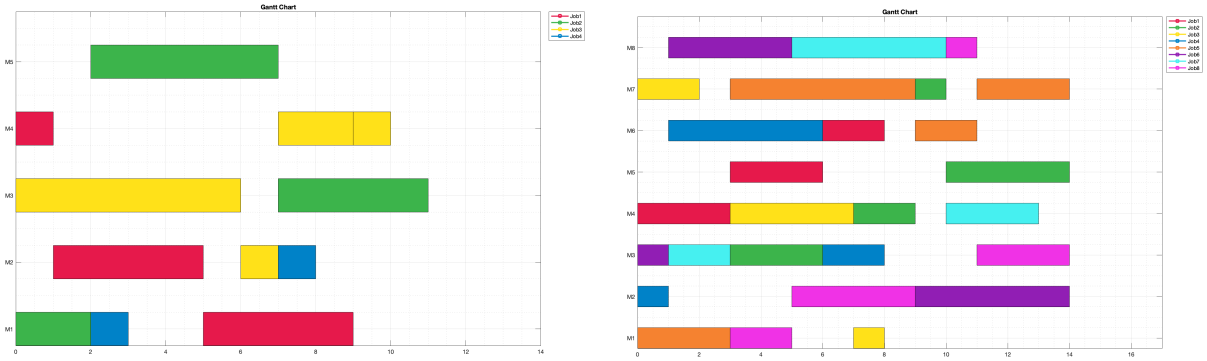


Figure 4.4: Swarm-chart of solutions obtained for instance kacem1.

4.2.1 Tests on Kacem dataset

The proposed algorithm was tested with the five instances from the Kacem dataset described by table 4.2, using the parameters presented in table 4.4. The results were compared with five already existing algorithms, namely Approach by Localization (AL+CGA) [71], Particle Swarm Optimization Combined with Simulated Annealing (PSO+SE) [44], Particle Swarm Optimization combined with Tabu Search (PSO+TS) [45], Discrete Artificial Bee Colony (P-DABC) [72] and Enhanced Pareto Artificial Bee Colony (EPABC) [36].



(a) A solution for kacem1 instance (4×5) with $C_{max} = 11$, $W_T = 32$ and $W_M = 10$.

(b) A solution for kacem2 instance (8×8) with $C_{max} = 14$, $W_T = 77$ and $W_M = 12$.

Figure 4.5: Gantt charts of solutions for two Kacem instances, 1 and 2 ((4×5) and (8×8)).

Amongst these algorithms, PSO+SE and PSO+TS, both use a fitness function that aggregates the objective functions into a single fitness function to be minimized, turning the problem into multi-criteria evaluation. AL+CGA proposes an approach that simultaneously minimizes the makespan and balances the workload of each machine. It is only on P-DABC and EPABC that a Pareto optimality approach is used.

Instance/	AL+CGA[71]		PSO+SA[44]		PSO+TS[45]		P-DABC[72]			EPABC [36]				Proposed MO-ABC			
Objective	s_1	s_2	s_1	s_2	s_1	s_2	s_1	s_2	s_3	s_1	s_2	s_3	s_4	s_1	s_2	s_3	s_4
kacem1																	
C_{max}	16				11		11	12	13	11	12	13	11	11	12	13	13
W_T	34				32		32	32	33	32	32	33	34	32	32	32	33
W_M	10				10		10	8	7	10	8	7	9	10	8	8	7
kacem2																	
C_{max}	15	16	15	16	14	15	14	15	16	14	15	16	16	14	15	16	16
W_T	79	75	75	73	77	75	77	75	73	77	75	73	77	77	75	73	75
W_M	13	13	12	13	12	12	12	12	13	12	12	13	11	12	12	13	12
kacem3																	
C_{max}							12	11	12	11	12	11		11	11	12	12
W_T							61	63	60	61	60	61		61	62	60	61
W_M							11	11	12	11	12	10		11	10	12	11
kacem4																	
C_{max}	7		7		7		8	7	8	8	7	8	7	8	7	8	
W_T	45		44		43		41	43	42	41	43	42	42	41	43	42	
W_M	5		6		6		7	5	5	7	5	5	6	7	5	5	
kacem5																	
C_{max}	23	24	12		11		12	11		11	11			11	12		
W_T	95	91	91		93		91	93		91	93			91	91		
W_M	11	11	11		11		11	11		11	10			11	11		

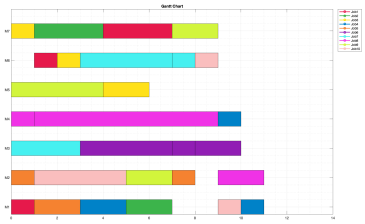
Table 4.7: Results comparison of the five Kacem instances.

Table 4.7 presents the results obtained by these five algorithms and the proposed algorithm. The proposed algorithm can obtain either more non-dominated solutions or better solutions when compared with the first four algorithms presented in table 4.7. As for the last two, the P-DABC and the EPABC, the results are quite similar either by number of non-dominated solutions or by the quality of solutions, since both these two algorithms already outperform the previous four mentioned. It is reasonable to say that the proposed algorithm has an overall good quality performance for these five Kacem instances.

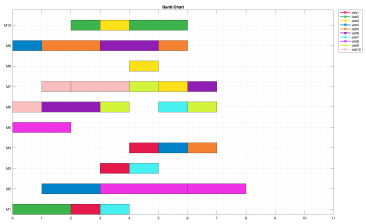
Figures 4.5 and 4.6 show the Gantt chart representations of some solutions of each of these instances.

4.2.2 Tests on Brandimarte dataset

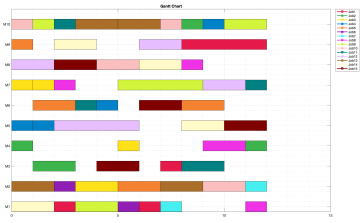
The second benchmarks dataset used to test the implemented algorithm is the Brandimarte dataset. As described by table 4.1, the instances of this dataset are generated through a uniform distribution between given limits. The proposed algorithm is using the parameters described in table 4.3. The results from five other algorithms found in the literature were used to compare the results of the implemented algorithm. These algorithms are a search algorithm developed by Xing [42], a Multi-Objective Genetic Algorithm (MOGA)[46], an Hybrid Tabu Search Algorithm (HTSA) [43], a Shuffle Frog-Leaping Algorithm



(a) A solution for kacem2 instance (10×7) with $C_{max} = 11$, $W_T = 62$ and $W_M = 10$.



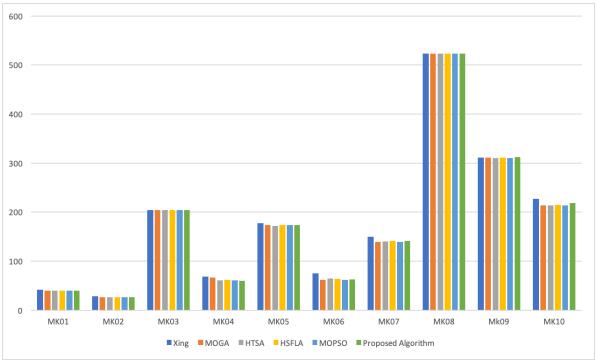
(b) A solution for kacem4 instance (10×10) with $C_{max} = 8$, $W_T = 41$ and $W_M = 7$.



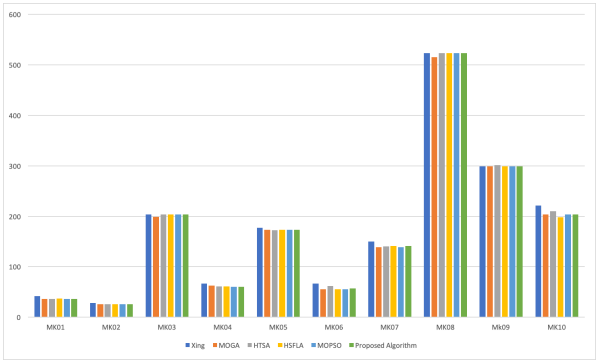
(c) A solution for kacem5 instance (15×10) with $C_{max} = 11$, $W_T = 91$ and $W_M = 11$.

Figure 4.6: Gantt charts of solutions for the Kacem instances 3, 4 and 5 (respectively (10×7), (10×10) and (15×10)).

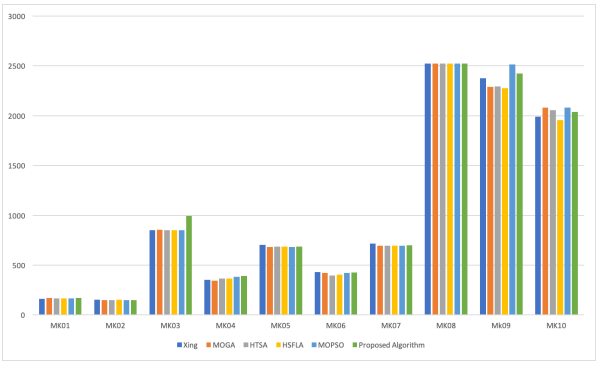
(HSFLA) [47], and a Particle Swarm Optimization (MOPSO) [48]. The best results are enhanced in bold in table 4.8.



(a) Distribution of results according to makespan, C_{max}.



(b) Distribution of results according to total workload, W_T.

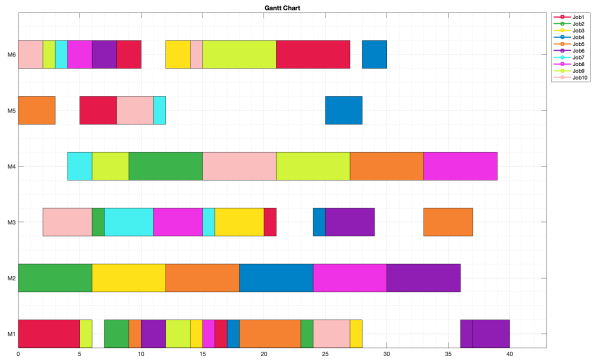


(c) Distribution of results according to maximum workload, W_M.

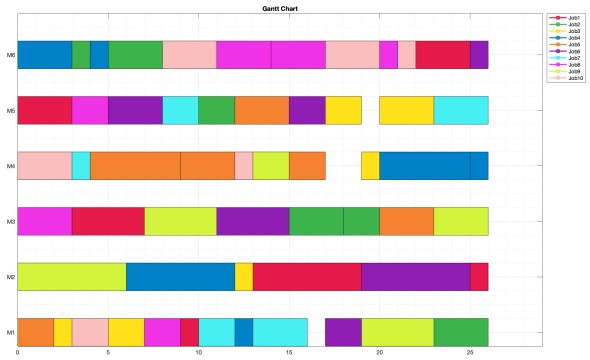
Figure 4.7: Distribution of the results for different algorithms.

On the algorithm presented by Xing, and on HTSA, the multi-objective problem turns into a mono-objective problem by evaluating the objective functions with a weighted sum. MOGA, HSFLA and MOPSO, all use a pareto optimality approach to evaluate non-dominated solutions.

Table 4.8 shows the results for non-dominated solutions with the minimum makespan obtained. Observing the solutions for *mk01* and comparing them the ones obtained with the proposed algorithm, it can be seen that the only algorithms that outperform it are the MOPSO, HTSA and MOGA, specifically in the Total Workload objective ,W_T. For *mk02* and *mk08*, the results are similar in terms of performance, to the ones obtained by MOGA, HTSA and MOPSO.



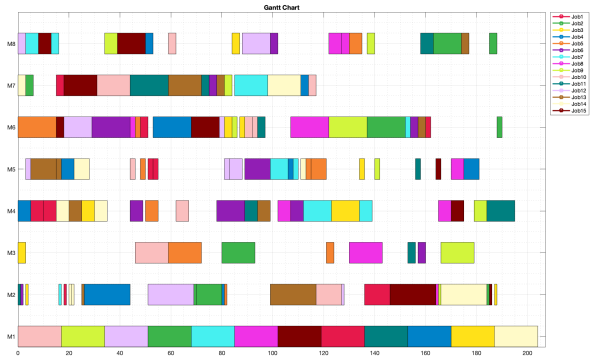
(a) A solution for mk01 with $C_{max} = 40$, $W_T = 170$ and $W_M = 36$.



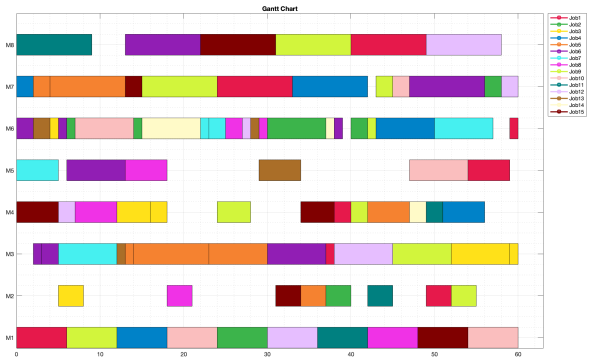
(b) A solution for mk02 with $C_{max} = 26$, $W_T = 151$ and $W_M = 26$.

Figure 4.8: Gantt charts of solutions for the mk01 and mk02 instances.

In *mk04*, the result obtained dominates all other results in comparison. In *mk05*, like in *mk01*, the solutions for MOPSO, HTSA and HSFLA also dominate the solution obtained by the proposed algorithm. In *mk03* it does not out perform the presented algorithms, as the total workload value is always higher. In *mk09* it does not out perform any of the other algorithms.



(a) A solution for mk03 with $C_{max} = 204$, $W_T = 993$ and $W_M = 204$.



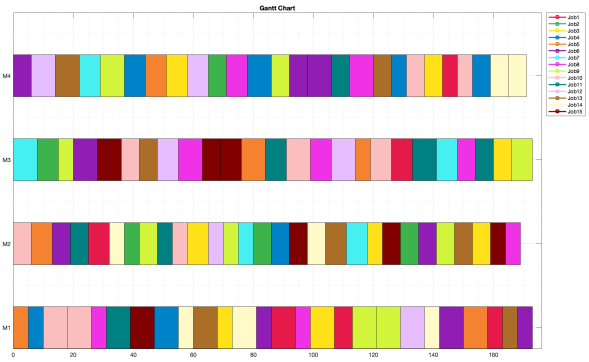
(b) A solution for mk04 with $C_{max} = 60$, $W_T = 390$ and $W_M = 60$.

Figure 4.9: Gantt charts of solutions for the mk03 and mk04 instances.

On *mk06* the solutions by MOGA and MOPSO dominate the solution obtained with the proposed algorithm. For *mk07* and *mk10* the only solution that the proposed algorithm dominates is the one proposed by Xing.

Figure 4.7 shows the difference between the different results shown in table 4.8. In this figure, it can be seen that the results from the proposed algorithm, although they are not the best for all instances, do not disperse, thus it is noticeable that this algorithm provides reasonably good results. Each sub-figure shows, for each instance, the results obtained for each objective function used. Figure 4.7(a) shows that the results obtained with the proposed algorithm do not diverge from the ones presented by the algorithms in comparison, all in terms of makespan. Despite this, shows that comparing the solutions presented in table 4.8 showing that the best algorithms overall are the MOPSO and the MOGA, the figure in questions illustrates that the dispersion from the solutions presented by these algorithms, and the proposed algorithm, is considerably small, making it nearly insignificant. Allowing the conclusion that

the proposed algorithm provides good quality solutions, for the instances that compose this Brandimarte dataset.

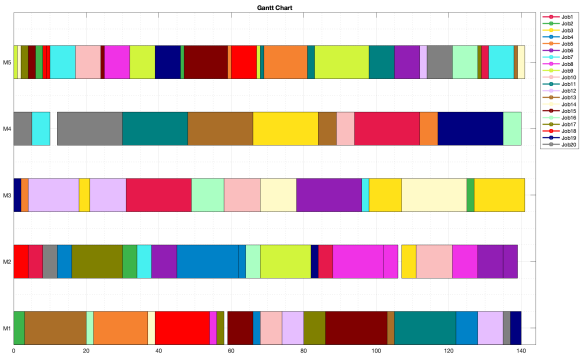


(a) A solution for mk05 with $C_{max} = 173$, $W_T = 686$ and $W_M = 173$.

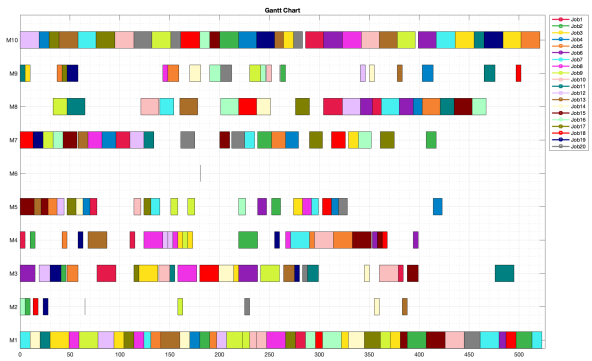
(b) A solution for mk06 with $C_{max} = 63$, $W_T = 425$ and $W_M = 57$.

Figure 4.10: Gantt charts of solutions for the mk05 and mk06 instances.

Figure 4.8 shows the Gantt chart representation of solutions for mk01 and mk02. Figure 4.9 shows the same representation but for instances mk03 and mk04, figure 4.10 shows for instances mk05 and mk06, figure 4.12 shows for instances mk07 and mk08. Finally, the same Gantt chart representation is shown in figure 4.12 for instance mk09 and mk10.



(a) A solution for mk07 with $C_{max} = 141$, $W_T = 697$ and $W_M = 141$.

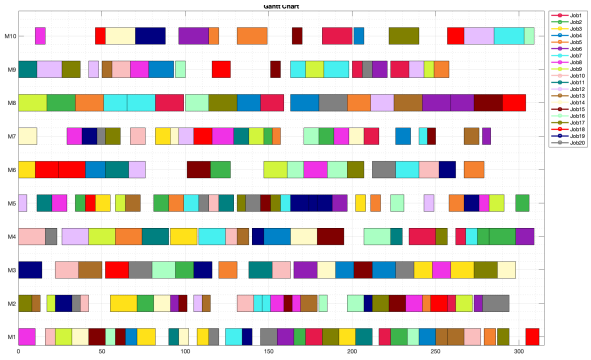


(b) A solution for mk08 with $C_{max} = 523$, $W_T = 2524$ and $W_M = 523$.

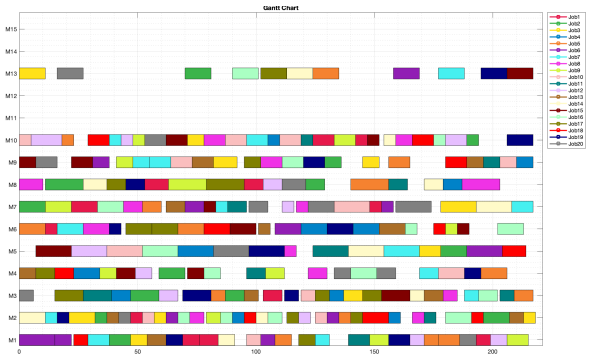
Figure 4.11: Gantt charts of solutions for the mk07 and mk08 instances.

Instance	Objective	Xing [42]	MOGA [46]	HTSA [43]	HSFLA [47]	MOPSO [48]	Proposed MO-ABC
MK01	C_{max}	42	40	**40**	40	**40**	40
	W_T	162	169	**167**	165	**167**	170
	W_M	42	36	**36**	37	**36**	36
MK02	C_{max}	28	**26**	**26**	26	**26**	**26**
	W_T	155	**151**	**151**	152	**151**	**151**
	W_M	28	**26**	**26**	26	**26**	**26**
MK03	C_{max}	204	204	**204**	**204**	**204**	204
	W_T	852	855	**852**	**852**	**852**	993
	W_M	204	199	**204**	**204**	**204**	204
MK04	C_{max}	68	66	61	62	61	**60**
	W_T	352	345	366	364	382	**390**
	W_M	67	63	61	61	60	**60**
MK05	C_{max}	177	**173**	172	173	**173**	173
	W_T	702	**683**	687	685	**683**	686
	W_M	177	**173**	172	173	**173**	173
MK06	C_{max}	75	**62**	65	64	**62**	63
	W_T	431	**424**	398	403	**424**	425
	W_M	67	**55**	62	55	**55**	57
MK07	C_{max}	150	**139**	140	141	**139**	141
	W_T	717	**693**	695	696	**693**	697
	W_M	150	**139**	140	141	**139**	141
MK08	C_{max}	523	**523**	523	523	523	523
	W_T	2524	**2524**	2524	2524	2524	2524
	W_M	523	**515**	523	523	523	523
MK09	C_{max}	311	311	**310**	**311**	310	312
	W_T	2374	2290	**2294**	**2275**	2514	2424
	W_M	299	299	**301**	**299**	299	299
MK10	C_{max}	227	**214**	214	215	**214**	218
	W_T	1989	**2082**	2053	1957	**2082**	2038
	W_M	221	**204**	210	198	**204**	204

Table 4.8: Results comparison of the ten Brandimarte instances.



(a) A solution for mk09 with $C_{max} = 312$, $W_T = 2424$ and $W_M = 299$.



(b) A solution for mk10 with $C_{max} = 218$, $W_T = 2038$ and $W_M = 204$.

Figure 4.12: Gantt charts of solutions for the mk09 and mk10 instances.

Chapter 5

Conclusions

The objective of this thesis was to develop tools able to create a schedule for a multi-objective optimization problem. To achieve this, it was implemented an Artificial Bee Colony algorithm.

To accomplish the proposed objective, the implementation of the Artificial Bee Colony algorithm was done using *Matlab*. The presented algorithm makes use of an aggregation of techniques to generate a good quality initial population, a coding scheme that generates only feasible solutions and a non-dominated sorting algorithm to sort and maintain the fittest individuals. To enhance the search capabilities of the algorithm, a local search based on critical path was implemented.

To test the proposed algorithm, a Bayesian optimization algorithm was first implemented to find the set of parameters that maximizes the performance of the algorithm according to metrics that evaluate both convergence and diversity within the solutions found.

After testing the algorithm using two benchmark datasets and comparing it with several different algorithms, the implementation and quality of the solutions obtained were both verified. Although the solution found by the proposed algorithm showed not to be always the best solution for the instances from the Brandimarte dataset, one instance in particular outperformed other algorithms, the *mk03*. As for the other instances of this particular dataset when the solution presented by the proposed algorithm was not the best, the difference between the solution found and, the others presented by the algorithms in comparison, is relatively small, thus making them good quality solutions. Differently, in the instances from the Kacem dataset, the proposed algorithm obtained high quality solutions. For all five instances, the proposed algorithm found a good diversity of non-dominated solutions, when compared to the other algorithms from the state-of-art the solutions found were of considerably good quality.

5.1 Future Work

While the results obtained were good, some improvements are needed to achieve a better approximation of a real work environment. While a static schedule is a good tool to start, in reality activities are disrupted by dynamic events such as job arrivals, job cancellations, machine breakdowns, and more. For future work, an implementation on dynamic environment is proposed, using the dynamic events

mentioned. Another proposition of future work is to include other objective functions such as tardiness or lateness, and implement the problem with release and due dates.

Bibliography

[1] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering*, 3(5):616–630, 2017. ISSN 20958099. doi: 10.1016/J.ENG.2017.05.015. URL http://dx.doi.org/10.1016/J.ENG.2017.05.015.

[2] M. Russmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch. Industry 4.0: World Economic Forum. *The Boston Consulting Group*, pages 1–20, 2015.

[3] A. A. Letichevsky, O. O. Letychevskyi, V. G. Skobelev, and V. A. Volkov. Cyber-Physical Systems. *Cybernetics and Systems Analysis*, 53(6):821–834, 2017. ISSN 15738337. doi: 10.1007/s10559-017-9984-9.

[4] M. Lorenz, M. Rüßmann, R. Strack, K. L. Lueth, and M. Bolle. Man and Machine in Industry 4.0. *Boston Consulting Group*, page 18, 2015.

[5] S. Smuts, A. van der Merwe, and H. Smuts. A Strategic Organisational Perspective of Industry 4.0: A Conceptual Model. In *International Federation for Information Processing*, volume 2, pages 307–318. 2020. ISBN 978-3-030-45001-4. doi: 10.1007/978-3-030-44999-5. URL http://dx.doi.org/10.1007/978-3-030-45002-1{_}2{%}0Ahttp://link.springer.com/10.1007/978-3-030-45002-1.

[6] F. Jovane and E. Westkämper. *The ManuFuture Road*. 2009. ISBN 9783540770114. doi: 10.1007/978-3-540-77012-1.

[7] A. Azevedo and A. Almeida. Factory templates for digital factories framework. *Robotics and Computer-Integrated Manufacturing*, 27(4):755–771, 2011. ISSN 07365845. doi: 10.1016/j.rcim.2011.02.004.

[8] E. H. D. Ribeiro da Silva, A. C. Shinohara, E. Pinheiro de Lima, J. Angelis, and C. G. Machado. Reviewing digital manufacturing concept in the Industry 4.0 paradigm. *Procedia CIRP*, 81:240–245, 2019. ISSN 22128271. doi: 10.1016/j.procir.2019.03.042. URL https://doi.org/10.1016/j.procir.2019.03.042.

[9] L. Li and J. Z. Huo. Multi-objective flexible Job-shop scheduling problem in steel tubes production. *Xitong Gongcheng Lilun yu Shijian/System Engineering Theory and Practice*, 29(8):117–126, 2009. ISSN 10006788. doi: 10.1016/s1874-8651(10)60063-4. URL http://dx.doi.org/10.1016/S1874-8651(10)60063-4.

[10] L. Venditti, D. Pacciarelli, and C. Meloni. A tabu search algorithm for scheduling pharmaceutical packaging operations. *European Journal of Operational Research*, 202(2):538–546, 2010. ISSN 03772217. doi: 10.1016/j.ejor.2009.05.038. URL http://dx.doi.org/10.1016/j.ejor.2009.05.038.

[11] F. Oliveira, P. M. Nunes, R. Blajberg, and S. Hamacher. A framework for crude oil scheduling in an integrated terminal-refinery system under supply uncertainty. *European Journal of Operational Research*, 252(2):635–645, 2016. ISSN 03772217. doi: 10.1016/j.ejor.2016.01.034.

[12] C. Bohle, S. Maturana, and J. Vera. A robust optimization approach to wine grape harvesting scheduling. *European Journal of Operational Research*, 200(1):245–252, 2010. ISSN 03772217. doi: 10.1016/j.ejor.2008.12.003. URL http://dx.doi.org/10.1016/j.ejor.2008.12.003.

[13] Y. Zhang, D. W. Gong, X. Y. Sun, and Y. N. Guo. A PSO-based multi-objective multi-label feature selection method in classification. *Scientific Reports*, 7(1):1–12, 2017. ISSN 20452322. doi: 10.1038/s41598-017-00416-0. URL http://dx.doi.org/10.1038/s41598-017-00416-0.

[14] K. Fang, G. Y. Ke, and M. Verma. A routing and scheduling approach to rail transportation of hazardous materials with demand due dates. *European Journal of Operational Research*, 261(1):154–168, 2017. ISSN 03772217. doi: 10.1016/j.ejor.2017.01.045. URL http://dx.doi.org/10.1016/j.ejor.2017.01.045.

[15] M. L. Pinedo. *Planning and Scheduling in Manufacturing and Services*. 2006. ISBN 9780387303031.

[16] A. Nagar, J. Haddock, and S. Heragu. Multiple and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81(1):88–104, 1995. ISSN 03772217. doi: 10.1016/0377-2217(93)E0140-S.

[17] T. C. Cheng and C. C. Sin. A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292, 1990. ISSN 03772217. doi: 10.1016/0377-2217(90)90215-W.

[18] D. Lei. Multi-objective production scheduling: A survey. *International Journal of Advanced Manufacturing Technology*, 43(9-10):925–938, 2009. ISSN 02683768. doi: 10.1007/s00170-008-1770-4.

[19] D. Danneberg, T. Tautenhahn, and F. Werner. A comparison of heuristic algorithms for flow shop scheduling problems with setup times and limited batch size. *Mathematical and Computer Modelling*, 29(9):101–126, 1999. ISSN 08957177. doi: 10.1016/S0895-7177(99)00085-0.

[20] K. Genova, L. Kirilov, and V. Guliashki. A survey of solving approaches for multiple objective flexible job shop scheduling problems. *Cybernetics and Information Technologies*, 15(2):3–22, 2015. ISSN 13144081. doi: 10.1515/cait-2015-0025.

[21] R. Wardlaw. Genetic algorithms for Shop Scheduling Problems : a Survey. 125(February):1–12, 1999.

[22] M. R. Garey, D. S. Johnson, and R. Sethi. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1(2):117–129, 1976. ISSN 0364765X. doi: 10.1287/moor.1.2.117.

[23] I. A. Chaudhry and A. A. Khan. A research survey: Review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591, 2016. ISSN 14753995. doi: 10.1111/itor.12199.

[24] A. Turkyılmaz, O. Senvar, I. Unal, and S. Bulkan. A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing*, 31(8):1949–1983, 2020. ISSN 15728145. doi: 10.1007/s10845-020-01547-4.

[25] G. K. Rand and S. French. Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop, 1982. ISSN 01605682.

[26] H. Hoogeveen. Multicriteria scheduling. *European Journal of Operational Research*, 167(3):592–623, 2005. ISSN 03772217. doi: 10.1016/j.ejor.2004.07.011.

[27] Y. Demir and S. Kürşat Işleyen. Evaluation of mathematical models for flexible job-shop scheduling problems. *Applied Mathematical Modelling*, 37(3):977–988, 2013. ISSN 0307904X. doi: 10.1016/j.apm.2012.03.020.

[28] P. Fattahi. A hybrid multi objective algorithm for flexible job shop scheduling. *World Academy of Science, Engineering and Technology*, 38(2):555–560, 2009. ISSN 2010376X. doi: 10.5281/zenodo.1332696.

[29] M. B. Horta Mesquita Da Cunha. *Scheduling of Flexible Job Shop Problem in Dynamic Environment*. PhD thesis, 2017.

[30] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 32(1):1–13, 2002. ISSN 10946977. doi: 10.1109/TSMCC.2002.1009117.

[31] J. Xiong, T. Xu, K. W. Yang, L. N. Xing, and Y. W. Chen. A hybrid multiobjective evolutionary approach for flexible job-shop scheduling problems. *Mathematical Problems in Engineering*, 2012, 2012. ISSN 1024123X. doi: 10.1155/2012/478981.

[32] F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the Flexible Job-shop Scheduling Problem. *Computers and Operations Research*, 35(10):3202–3212, 2008. ISSN 03050548. doi: 10.1016/j.cor.2007.02.014.

[33] M. Watanabe, K. Ida, and M. Gen. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. In *Computers and Industrial Engineering*, volume 48, pages 743–752, 2005. doi: 10.1016/j.cie.2004.12.008.

[34] G. Moslehi and M. Mahnam. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1):14–22, 2011. ISSN 09255273. doi: 10.1016/j.ijpe.2010.08.004. URL http://dx.doi.org/10.1016/j.ijpe.2010.08.004.

[35] Y. Yuan and H. Xu. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12(1):336–353, 2015. ISSN 15455955. doi: 10.1109/TASE.2013.2274517.

[36] L. Wang, G. Zhou, Y. Xu, and M. Liu. An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. *International Journal of Advanced Manufacturing Technology*, 60(9-12):1111–1123, 2012. ISSN 02683768. doi: 10.1007/s00170-011-3665-z.

[37] J. Q. Li, Q. K. Pan, and K. Z. Gao. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 55(9-12):1159–1169, 2011. ISSN 02683768. doi: 10.1007/s00170-010-3140-2.

[38] J. qing Li, Q. ke Pan, and S. xian Xie. A hybrid variable neighborhood search algorithm for solving multi-objective flexible job shop problems. *Computer Science and Information Systems*, 7(4):908–930, 2010. ISSN 18200214. doi: 10.2298/CSIS090608017L.

[39] S. Jia and Z. H. Hu. Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem. *Computers and Operations Research*, 47:11–26, 2014. ISSN 03050548. doi: 10.1016/j.cor.2014.01.010. URL http://dx.doi.org/10.1016/j.cor.2014.01.010.

[40] L. N. Xing, Y. W. Chen, and K. W. Yang. Double layer ACO algorithm for the multi-objective FJSSP. *New Generation Computing*, 26(4):313–327, 2008. ISSN 02883635. doi: 10.1007/s00354-008-0048-6.

[41] T. Hsu, R. Dupas, D. Jolly, and G. Goncalves. Evaluation of mutation heuristics for the solving of multiobjective flexible job shop by an evolutionary algorithm. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 5:655–660, 2002. ISSN 08843627. doi: 10.1109/icsmc.2002.1176444.

[42] L. N. Xing, Y. W. Chen, and K. W. Yang. An efficient search method for multi-objective flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 20(3):283–293, 2009. ISSN 09565515. doi: 10.1007/s10845-008-0216-z.

[43] J. Q. Li, Q. K. Pan, and Y. C. Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 59(4):647–662, 2010. ISSN 03608352. doi: 10.1016/j.cie.2010.07.014. URL http://dx.doi.org/10.1016/j.cie.2010.07.014.

[44] W. Xia and Z. Wu. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers and Industrial Engineering*, 48(2):409–425, 2005. ISSN 03608352. doi: 10.1016/j.cie.2005.01.018.

[45] G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers and Industrial Engineering*, 56(4): 1309–1318, 2009. ISSN 03608352. doi: 10.1016/j.cie.2008.07.021. URL http://dx.doi.org/10.1016/j.cie.2008.07.021.

[46] X. Wang, L. Gao, C. Zhang, and X. Shao. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 51(5-8):757–767, 2010. ISSN 02683768. doi: 10.1007/s00170-010-2642-2.

[47] J. Li, Q. Pan, and S. Xie. An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Applied Mathematics and Computation*, 218(18):9353–9371, 2012. ISSN 00963003. doi: 10.1016/j.amc.2012.03.018. URL http://dx.doi.org/10.1016/j.amc.2012.03.018.

[48] S. Huang, N. Tian, Y. Wang, and Z. Ji. Multi-objective flexible job-shop scheduling problem using modified discrete particle swarm optimization. *SpringerPlus*, 5(1), 2016. ISSN 21931801. doi: 10.1186/s40064-016-3054-z.

[49] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007. ISSN 09255001. doi: 10.1007/s10898-007-9149-x.

[50] A. Rajasekhar, N. Lynn, S. Das, and P. N. Suganthan. Computing with the collective intelligence of honey bees – A survey. *Swarm and Evolutionary Computation*, 32:25–48, 2017. ISSN 22106502. doi: 10.1016/j.swevo.2016.06.001. URL http://dx.doi.org/10.1016/j.swevo.2016.06.001.

[51] S. M. Vieira, J. M. Sousa, and T. A. Runkler. Multi-criteria ant feature selection using fuzzy classifiers. *Studies in Computational Intelligence*, 242:19–36, 2009. ISSN 1860949X. doi: 10.1007/978-3-642-03625-5_2.

[52] F. C. Duran, C. Cotta, and A. J. Fernández. *Nature-Inspired Algorithms for Optimisation*, volume 193. 2009. ISBN 978-3-642-00266-3. doi: 10.1007/978-3-642-00267-0. URL http://dblp.uni-trier.de/db/series/sci/sci193.html{#}DuranCF09.

[53] S. K. Jha, J. Bilalovic, A. Jha, N. Patel, and H. Zhang. Renewable energy: Present research and future scope of Artificial Intelligence. *Renewable and Sustainable Energy Reviews*, 77(April): 297–317, 2017. ISSN 18790690. doi: 10.1016/j.rser.2017.04.018.

[54] S. Ozturk, R. Ahmad, and N. Akhtar. Variants of Artificial Bee Colony algorithm and its applications in medical image processing. *Applied Soft Computing*, 97:106799, 2020. ISSN 15684946. doi: 10.1016/j.asoc.2020.106799. URL https://doi.org/10.1016/j.asoc.2020.106799.

[55] F. Xu, H. Li, C. M. Pun, H. Hu, Y. Li, Y. Song, and H. Gao. A new global best guided artificial bee colony algorithm with application in robot path planning. *Applied Soft Computing Journal*, 88:

106037, 2020. ISSN 15684946. doi: 10.1016/j.asoc.2019.106037. URL https://doi.org/10.1016/j.asoc.2019.106037.

[56] M. Marinaki, Y. Marinakis, and G. E. Stavroulakis. Fuzzy control optimized by a Multi-Objective Differential Evolution algorithm for vibration suppression of smart structures. *Computers and Structures*, 147:126–137, 2015. ISSN 00457949. doi: 10.1016/j.compstruc.2014.09.018. URL http://dx.doi.org/10.1016/j.compstruc.2014.09.018.

[57] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University. Technical Report TR06, 2005.

[58] H. Nasiraghdam and S. Jadid. Optimal hybrid PV/WT/FC sizing and distribution system reconfiguration using multi-objective artificial bee colony (MOABC) algorithm. *Solar Energy*, 86(10):3057–3071, 2012. ISSN 0038092X. doi: 10.1016/j.solener.2012.07.014. URL http://dx.doi.org/10.1016/j.solener.2012.07.014.

[59] D. Oliva, E. Cuevas, and G. Pajares. Parameter identification of solar cells using artificial bee colony optimization. *Energy*, 72:93–102, 2014. ISSN 03605442. doi: 10.1016/j.energy.2014.05.011.

[60] I. M. S. De Oliveira and R. Schirru. Swarm intelligence of artificial bees applied to in-core fuel management optimization. *Annals of Nuclear Energy*, 38(5):1039–1045, 2011. ISSN 03064549. doi: 10.1016/j.anucene.2011.01.009.

[61] A. E. Hassanien, E. Emary, and H. M. Zawbaa. Retinal blood vessel localization approach based on bee colony swarm optimization, fuzzy c-means and pattern search. *Journal of Visual Communication and Image Representation*, 31:186–196, 2015. ISSN 10959076. doi: 10.1016/j.jvcir.2015.06.019. URL http://dx.doi.org/10.1016/j.jvcir.2015.06.019.

[62] L. Wang, G. Zhou, Y. Xu, S. Wang, and M. Liu. An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 60(1-4):303–315, 2012. ISSN 02683768. doi: 10.1007/s00170-011-3610-1.

[63] I. Roman, J. Ceberio, A. Mendiburu, and J. A. Lozano. Bayesian optimization for parameter tuning in evolutionary algorithms. *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, pages 4839–4845, 2016. doi: 10.1109/CEC.2016.7744410.

[64] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 4:2951–2959, 2012. ISSN 10495258.

[65] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58 (5):3250–3265, 2012. ISSN 00189448. doi: 10.1109/TIT.2011.2182033.

[66] Y. Tian, R. Cheng, X. Zhang, and Y. Jin. PlatEMO: A matlab platform for evolutionary multi-objective optimization. *arXiv*, (november):73–87, 2017.

[67] U. K. Wickramasinghe and X. Li. A distance metric for evolutionary many-objective optimization algorithms using user-preferences. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5866 LNAI(3):443–453, 2009. ISSN 03029743. doi: 10.1007/978-3-642-10439-8_45.

[68] M. R. Singh, M. Singh, S. S. Mahapatra, and N. Jagadev. Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, 85(9-12):2353–2366, 2016. ISSN 14333015. doi: 10.1007/s00170-015-8075-1. URL http://dx.doi.org/10.1007/s00170-015-8075-1.

[69] C. A. C. Coello, G. B. Lamont, D. A. V. Veldhuizen, D. E. Goldberg, and J. R. Koza. *Evolutionary Algorithms for Solving Multi-Objective Problems Second Edition Genetic and Evolutionary Computation Series Series Editors Selected titles from this series :*. 2007. ISBN 978-0-387-33254-3. URL http://www.ingentaconnect.com/content/oso/5066902/2001/00000001/00000001/art00001.

[70] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, 1993. ISSN 02545330. doi: 10.1007/BF02023073.

[71] I. Kacem, S. Hammadi, and P. Borne. Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, 60(3-5):245–276, 2002. ISSN 03784754. doi: 10.1016/S0378-4754(02)00019-8.

[72] J. Q. Li, Q. K. Pan, and M. F. Tasgetiren. A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Applied Mathematical Modelling*, 38(3):1111–1132, 2014. ISSN 0307904X. doi: 10.1016/j.apm.2013.07.038. URL http://dx.doi.org/10.1016/j.apm.2013.07.038.