Capstone Final Project

Design of a solution technique based on an integral approach for the Flexible Open-Flow Shop scheduling problem

Fernando Andrés Hurtado Villamizar[a,c], Diego Alberto Espinosa Valderrama[a,c] , Julián David Sánchez Duarte[a,c],

Carlos Eduardo Montoya Casas[b,c]

[a]*Industrial engineering student*
[b]*Associate Professor, Project Director, Departament of Industrial Engineering*
[c]*Pontificia Universidad Javeriana, Bogotá, Colombia*

Abstract

In manufacturing industries, scheduling is a form of decision-making that plays a crucial role. The determination of the methods by which a set of jobs must be manufactured in order to seek specific goals leads to the development of different schedule techniques. However, scheduling depends on the type of workshop or manufacturing environment such as open shop, job shop and flow shop. There are cases that more than one environment for the same manufacturing process could coexist. This project deals with a specific scheduling problem in which each job is processed under the combination of two shop environments; the first one is related to an open shop while the second one corresponds to a flow shop; this problem is called the Flexible open-flow shop (FOFS). These types of scheduling problems present NP-hardness, meaning the neediness of sophisticated algorithms to find solutions in reasonable computational times. Additionally, are commonly solved separately or by approximating into another workshop, leaving the interaction of both environments irrelevant. Thus, the main objective of this project is to design solution techniques based on an integral approach to minimize the maximum completion time also known as makespan.

Considering the lack of representation of this problem under mathematical formulation, a MILP formulation was made in order to present and define the constraints of the problem. Therefore, four solution techniques were designed based on the general genetic algorithm (GA), due remarkable results from literature review, to solve the FOFS. Three sets of instances (small, medium and large) were generated and adapted from flexible flow shop instances by Naderi et al [1] to represent the problem and test the solution techniques. Finally, the validation of makespan results from the solutions techniques were made through the mathematical model for small size instances, and the design of three different lower bounds for medium and large size instances. The results were statistically analyzed by three different dimensions (solution quality, solution speed and shop performance) and compared against results obtained from Mejia G et al [2] for a similar problem to test adaptability and competitiveness for other type of instances.

Key words: Scheduling, Mixed shop, Flow shop, Open shop, Flexible, MILP, Makespan, NP-hard

1. Justification

Scheduling is a decision-making process that plays a crucial role in manufacturing and services industries. It deals with the allocation of resources to tasks over a specific period and its goal is to optimize one or more objectives. In manufacturing industries scheduling resources may be machines in a workshop and tasks may be jobs that must be processed in the production system [3]. The scheduling importance is given for the competitive ambience in the market-place in which companies must comply shipping dates and satisfy the demand of their customers. For example, scheduling jobs according to their due dates could minimize tardiness and satisfy

customers demand but ignores the full use of resources. On the other hand, Scheduling jobs only optimizing resources could reduce idle times and avoid bottlenecks, but disregarding shipping dates.

Scheduling problems mainly depend on the manufacturing environment of the process i.e. flow-shop, job-shop or open-shop and their own variations. Throughout the years, these environments have acquired more complexity and dynamism. Nowadays it is possible to find the mixture of at least two of the previously mentioned shop environments. This combination and its scheduling is called *Mixed Shop Scheduling Problem (MSS).*

In the literature, the concept of mixed shop has its particular variations depending on the features of the problem. Nguyen and Bao [4] considers a combination of the job shop problem and the open shop problem. Masuda T et al [5] addresses another type of mixed shop with two subsets of jobs, one subset has jobs that must be processed in flow shop and the other subset has jobs that must be processed as open shop. Shakhlevich N et al [6]reviewed different results of mixed shops conformed by two set of jobs, one of them processed as flow shop or job shop and the other one processed as open shop.

The type of mixed shop addressed in this project corresponds to the *Flexible Open-Flow Shop Scheduling Problem.* This is a combination of two shop environments. The first part of the process is an open-shop in which a set of jobs must be processed in a set of machines arbitrarily, meaning that there is not a specific processing route. The second part of the process behaves as a flow-shop that unlike the open-shop, a set of jobs must be processed in a set of machines in a given processing route. Each machine can only process a single operation, but an operation can be executed by more than one machine. The group of machines that execute the same operation is called "stage" [7][8]. The existence of stages in the processing environment indicates that is flexible [9], allowing a simultaneous execution of an operation for more than one job. Consequently, in the first part of the process (Flexible open-shop) there are three main decisions: processing route determination, job sequence and job assignment to machines at each stage [10]. For the second part of the process (Flexible flow-shop) the main decisions are machine assignment and job sequence [11].

The proposed scheduling problem manages certain assumptions. First, processing and setup times are deterministic. Second, machines at each stage are identical, this means that at each stage machines have the same processing and setup times. Finally, setups are sequence independent, meaning that the setup time is the same for each job and independent of the sequence given by the schedule.

As mentioned, scheduling is done with the purpose of optimizing one or more objectives, such as the minimization of maximum completion time, the minimization the total tardiness of the jobs, the minimization of total machines idle time, among others. The most common objective addressed in literature is the maximum completion time also known as makespan. This objective function focus on the maximization of machines utilization and reduction of idle times.

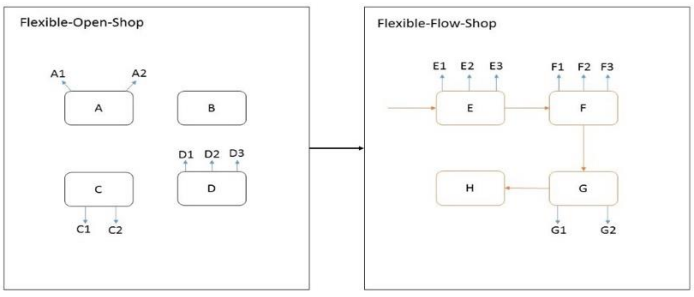

Figure 1 Representation of flexible open-flow shop environment

Generally, some processes in metalworking industries illustrate this environment. Production processes of products like crossbow springs and helical steel are distributed in the following manner: The initial operations of the process do not need previous activities to start because the process nature allows randomness, meaning that there are no order restrictions. The remaining operations must be processed according to a predefined precedence relation since the completion of one operation affects the fulfilment of the following ones.

In addition to the metalworking industry, several sectors can handle this type of manufacturing environment. For example, in the textile industry some operations must be processed in a sequential manner, as it occurs in a flow Shop environment, while the execution of the remaining processes can be related to an open shop. Furthermore, to increase productivity and improve the throughput, organizations use two or more machines per stage.

Also, there are companies that use empirical schedule methods without adopting any computational tool or methodology for that purpose. As a result, productivity gets affected, increasing the risk of not fulfilling customer´s requirements without ensuring a proper use of the resources. These issues can be handle by employing scheduling techniques related to the features of the manufacturing environment of the process.

Nevertheless, some scheduling tools such as simple dispatching rules, may not be enough for ensuring good schedules. The mixed shop scheduling problem go along with NP-hard computational complexity. Shakhlevich et al [6] survey different results on the computational complexity of MSS to establish the boundary between polynomial solvable and NP-hard problems, concluding the neediness of more sophisticated algorithms that find solutions in reasonable computational times. Consequently, given the fact that the problem proposed is a type of mixed shop, the necessity of computational tools such as metaheuristics-based algorithms become essential to find good schedules in reasonable computational times.

Despite the different types of MSS problems studied and their complexity, the flexible open-flow shop scheduling problem has not been sufficiently explored in the literature. To the best of our knowledge Bejarano et al. [12] are the only authors who dealt with the same scheduling problem. Nevertheless, they didn´t consider the interaction between the two workshops in their solution approach. Therefore, given also the practical application and complexity of the problem at hand, this project aims at finding a proper answer to the next question: “how to schedule the flexible open-flow shop without approximating it, based on an integral approach considering the interaction between the two shop environments?”

2. Literature Review

As mentioned, the problem at hand considers the interaction between two shop environments. Hence, it is important to mention studies, surveys and research found in literature for these particular workshops. For this purpose, the literature review is divided in three parts, starting with the flexible flow shop followed by the flexible open shop. Finally, a general revision of similar mixed shop scheduling problems is also included. This section is focused, mainly, on solution techniques for these scenarios that minimize maximum completion time "makespan" [7], [13],[14],[15].

Flexible flow shop:

Flexible flow shop (FFS) or Hybrid flow shop (HFS) scheduling have been treated over the time by many authors due to its common use in the industrial environment like textiles industries, electronic industries, cement factories, among others. The flexible component increases the production capacity, readjust the workload and reduce bottlenecks impact [16]. To minimize makespan the following parameters are taken into consideration: The processing time, number of jobs, number of stages, number of machines per stage, prevalence of activities, the number of units processed by machines and setup times.

Most authors have treated the subject with two machines per stage finding algorithms to solve the problem optimally. Ruiz and Vasquez [16] addressed a simplified version of the problem by considering a few number of stages. Additionally, they refer to Branch and Bound (B&B) as one of the most used approaches to solve the FFS optimally. Verma and Dessouky [17] were the first to approach the problem with two and three stages with B&B methods [18]. In the other hand, Jungwattanakit et al [8] developed a MILP (Mixed Integer Linear Programming) finding an optimal solution with up to six jobs and four stages.

For certain instance sizes, the mathematical model cannot be solved in reasonable time given the NP-hard nature of the problem. Hoogeveen et al [19] state that a soon as there are at least two machines at any stage the problem becomes NP-hard. Ranjan and Mahapatra [20] discussed that FFS is a problem with deficiency to solve in a reasonable amount of time due to its computational complexity. As a consequence, several authors have tackled the FFS problem with different solution approaches based on the utilization of heuristics and metaheuristics.

Glover [21] proposed an improved Tabu search (TS) that has been used by other authors to solve the FFS. In addition, this scheduling problem has recently been treated by PSO metaheuristic[20], since it is an effective algorithm which gives quality solutions in a reasonable computational time. Other employed metaheuristics are Ant Colony Optimization (ACO), Artificial Immune System (AIS), Neural Networks (NN) among others [14].

Another known metaheuristic is the Genetic algorithm (GA), which have been widely used for this type of problems obtaining good results when comparing against other solution approaches. For instance, Ruiz and Maroto [22] proposed a GA for a problem similar to the one stated in this project obtaining better results in comparison to other heuristics, metaheuristics such as ACO, TS, SA among others. In addition, Ruiz et al [16] made a compilation of literature for the FFS and possible variations, concluding that among the most common metaheuristics, GA represents the 32% of use, being this algorithm the most used metaheuristic for the FFS problem.

Flexible open shop:

Despite the big amount of research of flexible flow-shop scheduling problems, flexible open-shop (FOS) scheduling problems have not been worked with the same frequency. Pure open-shop and flexible open-shop configurations can be identified in metallurgic industry or automotive garages for vehicle repairs [7].

Different solution approaches have been considered to obtain optimal solutions for the open Shop scheduling problems, for example Pinedo[3] introduces the LAPT *longest alternate processing time first* rule, which leads to an optimal schedule for two machines (O2) open-shop. This rule guarantees the minimum makespan where idleness is not allowed. Other optimal solution approaches are based in the development of different mathematical formulations. For instance, Naderi et al [10] made a mathematical model for the problem formulation, likewise, Matta [23] made two MILP formulations, the first with time-based decision variables and the second formulation with sequence-based decision variables. Although MILP models are not as efficient as other optimization methods for large problem sizes, it is a well-known way to understand and present most of the scheduling problems.

As mentioned, few FOS scheduling problems have been presented in literature mainly as a result of the problem formulation complexity. Pinedo [3] proofed that the open shop scheduling O3 is NP-hard. Wang-Lian et al [24] analyzed the non-deterministic polynomial hard complexity for FOS problems. Therefore, most of the solutions and studies presented in literature, approach the problem directly with a variety of techniques such as metaheuristics.

Danyu Bai et al [7] proposed a differential evolutionary metaheuristic according to the jobs starting times. Also, Wang-Lian et al [24] used a hybrid differential evolution algorithm with splitting jobs. In addition, a General Dense Scheduling GDS algorithm was proposed by Danyu Bai et al [7] for solving large scale instances. Authors obtained that GDS-based heuristic improves solutions for large-scale problems whereas the DE

differential evolution algorithm obtains high-quality results for moderate-scale problems. B Naderi et al [10] gives a solution to FOS problem proposing six different metaheuristics based on memetic algorithms and shows a simple form of simulated annealing with its local search engine.

Matta [23] solves the "Multiprocessor open Shop" which corresponds to a FOS, by developing a Genetic algorithm. The author found that the difference between this metaheuristic and the lower bound proposed was on average 11%. In some cases, the GA reached the lower bound affirming it was the optimal makespan. The author claims it is the closest approach to the optimal solution found in literature, concluding that it is a good way to address the FOS given the quality of the obtained results.

Mixed shop:

In relation to mixed shop scheduling problems, authors have faced the NP-hard nature of the problem by designing and improving different solution techniques proposed in the literature. Masuda T et al [5] consider a two-machine shop scheduling problem with two subsets of jobs. One subset must be processed as flow shop and the remaining subset as open shop. They developed different algorithms based on Johnson´s algorithm for the flow shop and Gonzalez and Sahni algorithm for the open shop, reaching optimality under specific conditions. A. Rossi et al [25] dealt with a particular problem, in which an operation can be split in a number of micro-operations, which can be performed on the same job in an arbitrary route by different resources (machines) of the belonging stage, calling it a hybrid (flexible) shop type combination of flow shop and open shop. They solved this scheduling problem with different metaheuristics such as Ant Colony Optimization (ACO), Iterated Local Search (ILS) and Tabu Search (TS).

Regarding to the interaction between flexible open and flow shop environments, Tao Ren e.t. al [26] worked on a non-flexible flow-open shop, where the first part of the process was a flow shop and the second one corresponded to an open shop. These authors used a DS-based heuristic algorithm to solve the problem for large scale instances and a discrete differential evolution algorithm for moderate-scale instances. Bejarano L [12] approximated the flexible open-flow shop problem to a flexible flow shop (FFS) and designed a genetic algorithm to solve it. The previous authors are the most related with this problem, and none of them propose a MILP formulation considering both environments. FFS and FOS authors have designed independent MILP for each environment. Consequently, there is no mathematical model for this particular problem.

As mentioned, the flexible open-flow shop is seldom found in literature, therefore there are not instances specifically designed according to the features of the problem. Nevertheless, other authors have focused in other types of mixed shops, developing diverse solution techniques based on different metaheuristics such as Bacterial Foraging Optimization algorithm [13], Ant Colony Optimization algorithm [27] and Tabu search [28]. Nguyen and Bao [4] proposed an improved genetic optimization process based on the general genetic algorithm to solve a combination of the job shop problem and the open shop problem pointing out the performance of the GA when solving "Pure" shop scheduling problems (flow shop, job shop and open shop) and combinatory problems such as flexible job shop. Also, the authors refer to Błażewicz [29] who states that the results of GA proofs that is an effective method to solve shop scheduling problems with less computational effort and better results.

Summarizing, the literature related with the mixed shop scheduling problems have normally proposed independent solutions for each environment that compose the complete workshop, or have solved it based on an approximation to a single scheduling environment, like Bejarano et al. [12]. Hence, it takes place the opportunity of designing a solution technique for this scheduling problem by considering the interaction of the two workshops altogether. This project focuses on the utilization of a genetic algorithm given its computational efficiency and favorable results [23][30] considering it as a good starting point for the development of an integrated solution approach to solve the FOFS.

3. Objectives

To design and provide a solution technique based on an integral approach for the Flexible open-flow shop scheduling problem minimizing maximum completion time "makespan".

a. Develop a mathematical model for the Flexible open-flow shop scheduling problem minimizing maximum completion time.

b. Select and design the solution technique(s), based on a metaheuristics and/or heuristic that considers the direct interaction between the flexible open and flexible flow shop environments.

c. Generate small, medium and large size instances for the *Flexible open-flow shop scheduling problem*.

d. Validate the solution technique(s) not only with the mathematical model for small instances, but also with the adjustment of instances stablished in literature for flexible flow shop and flexible open shop that minimizes maximum completion time.

4. Methodology

4.1 MILP

By cause of the non-existence of a MILP formulation, this section presents a mathematical model for the flexible open-flow shop (FOFS) scheduling problem that considers the interaction between the two shop environments i.e FOS and FFS. This model is based on formulations made in [10],[8],[23]. This type of formulation is presented by Matta [23] as a sequence-based formulation in which decision variables indicate which jobs are processed directly after one another at each stage and identifying, primarily, completion times of each job. Therefore, general notation including sets, parameters, and variables are defined as follows.

J: set of jobs that must be processed, $T = \{0 \dots MaxJ\}$.

T: total set of stages, $T = T1 \cup T2$

$T1$: subset of flexible open shop stages : $T1 \in T$, $T1 = \{0 \dots MaxO\}$.

$T2$: subset of flexible flow shop stages : $T2 \in T$, $T2 = \{MinF \dots MaxF\}$.

R: Total set of machines

M_t: subset of machines in stage $t : t \in T$

Parameters:

P_{jt}: Processing time of job j in stage t

SU_{jt}: Setup time of job j in stage t

MM: Big number

Decision Variables:

C_{jt}: Completion time of job j in stage t

C_{max}: Maximum completion time

H_{jtk}: Binary, 1 if job j is processed at stage t inmediately before stage k, 0 otherwise

x_{rjlt}: Binary, 1 if job j is processed inmediately before job l on machine r at stage t, 0 otherwise (FOS)

W_{jktr}: Binary, 1 if job j is processed inmediately before job k on machine r at stage t, 0 otherwise (FFS)

Mathematical formulation:

MINIMIZE C_{max} (1)

$$\sum_{r \in M_t} \sum_{j \in J: j \le MaxJ-1} x_{rjlt} \ge 1 \qquad \forall t \in T1, l \in J: l \ge 1 \quad (2)$$

$$\sum_{r \in M_t} \sum_{l \in J: l \ge 1} x_{rjlt} \ge 1 \quad (3)$$
$$\forall t \in T1, j \in J: j \le MaxJ - 1$$

$$\sum_{l \in J: l \ge 1} x_{r0lt} = 1 \qquad \forall t \in T1, r \in M_t \quad (4)$$

$$\sum_{j \in J: j \le MaxJ-1} x_{rj(MaxJ)t} = 1 \qquad \forall t \in T1, r \in M_t \quad (5)$$

$$x_{rjjt} = 0 \qquad \forall t \in T1, r \in M_t, j \in J \quad (6)$$

$$x_{r0(MaxJ)t} = 0 \qquad \forall t \in T1, r \in M_t \quad (7)$$

$$\sum_{l \in J} x_{r(MaxJ)lt} = 0 \qquad \forall t \in T1, r \in M_t \quad (8)$$

$$\sum_{j \in J} x_{rj0t} = 0 \qquad \forall t \in T1, r \in M_t \quad (9)$$

$$\sum_{l \in T1: l \le MaxO,\ l>0,\ l \ne t} H_{jtl} = 1 \quad (10)$$
$$\forall j \in J, t \in T1: j \le MaxJ, t < MaxO.$$

$$\sum_{t \in T1: t<MaxO,\ t \ge 0,\ l \ne t} H_{jtl} = 1 \quad (11)$$
$$\forall j \in J, l \in T1: j \le MaxJ, l > 0.$$

$$\sum_{t \in T1: t<MaxO,\ t>0} H_{jt(MaxO)} = 1 \qquad \forall j \in J: j \le MaxJ. \quad (12)$$

$$\sum_{l \in T1: l<MaxO,\ l>0} H_{j0l} = 1 \qquad \forall j \in J: j \le MaxJ. \quad (13)$$

$$C_{lt} \ge \left(C_{jt} + SU_{lt} + P_{lt}\right) - \left(1 - \sum_{r \in M_t} x_{rjlt}\right) * MM \quad (14)$$
$$\forall t \in T1, j \in J, l \in J: j \ne l$$

$$C_{jk} \ge \left(C_{jt} + SU_{jk} + P_{jk}\right) - \left(1 - H_{jtk}\right) * MM \quad (15)$$
$$\forall t \in T1, k \in T1, j \in J: j \le MaxJ, k \ne t$$

$$C_{jt} \ge P_{jt} \quad \forall t \in T1, j \in J \quad (16)$$

$$\boldsymbol{C_{j(MinF)} \ge C_{j(MaxO)} \quad \forall j \in J} \quad (17)$$

$$\sum_{r \in M_t} \sum_{j \in J: j \le MaxJ-1} W_{jktr} \ge 1 \quad (18)$$
$$\forall t \in T2, k \in J: k \ge 1$$

$$\sum_{r \in M_t} \sum_{k \in J: k \ge 1} W_{jktr} \ge 1 \quad (19)$$
$$\forall t \in T2, j \in J: j \le MaxJ - 1$$

$$\sum_{k \in J: k \ge 1} W_{0ktr} = 1 \qquad \forall t \in T2, r \in M_t \quad (20)$$

$$\sum_{j \in J: j \le MaxJ-1} W_{j(MaxJ)tr} = 1 \qquad \forall t \in T2, r \in M_t \quad (21)$$

$$\sum_{k \in J} W_{(MaxJ)ktr} = 0 \qquad \forall t \in T2, r \in M_t \quad (22)$$

$$\sum_{j \in J} W_{j0tr} = 0 \qquad \forall t \in T2, r \in M_t \quad (23)$$

$$W_{0(MaxJ)tr} = 0 \qquad \forall t \in T2, r \in M_t \quad (24)$$

$$W_{jjtr} = 0 \qquad \forall t \in T2, j \in J, r \in M_t \quad (25)$$

$$C_{kt} \ge \left(C_{k(t-1)} + SU_{kt} + P_{kt}\right) \quad (26)$$
$$\forall t \in T2, k \in J: MaxF > t > MinF$$

$$C_{kt} \ge \left(C_{jt} + SU_{kt} + P_{kt}\right) - \left(1 - \sum_{r \in M_t} W_{jktr}\right) * MM \quad (27)$$
$$\forall t \in T2, j \in J, k \in J: MaxF > t > MinF$$

$$C_{max} \ge C_{jt} \quad \forall t \in T2, j \in J \quad (28)$$

This model is divided in two parts, constraints (2) to (16) present FOS formulation whereas (18) to (28) display FFS formulation. (1) Shows the objective function which is to minimize makespan. Constraints (2), (3), (18) and (19) ensure the construction of a consistent sequence of jobs at each stage in both environments. Notice that the model is focused on resources utilization, meaning that a machine can't be left unused. For that reason, there cannot be more machines at one each stage that the total number of jobs. Constraints (4) to (9) and (20) to (25) ensure sequence feasibility and at the same time allow flexibility of machines at each part of the model; e.g. job 0 must always start the sequence, while job MaxJ must always finish it, (7) and (24) reaffirm the resources utilization statement guaranteeing that each machine on each stage has at least one job to process. Constraints (10) to (13) represent the additional job route decision related to the FOS part, forcing the construction of a feasible route for each job. More specifically, Constraints (10) and (11) ensure that each job must be processed at every FOS stage in a sequence structure, while constraints (12) and (13) states that each job at FOS must start at 0 and finish at MaxO. Constraints (14) to (16) and (26) to (27) calculate the completion time of each job at each stage.

Constraint (17) allows the interaction between both environments granting that completion times of FOS and FFS are taken conjointly i.e. when a job is totally processed in FOS it can begin its way through FFS maintaining the completion time of the previous stage. Finally, Constraint (28) calculates the maximum completion time of the problem.

As mentioned, this mathematical model is based from the ones presented in the literature for FFS and FOS. For that reason, some adaptations are required to fulfill feasibility and guarantee that the makespan is well computed. First, Jobs 0 and MaxJ are non-real jobs. Likewise, in set T1 stages 0 and MaxO and in set T2 stages MinF and MaxF also are non-real stages. The creation and utilization of non-real elements are in order to ease the mathematical formulation, for instance, constraints (4), (5), (20) and (21) make use of the non-real jobs in every machine at each stage to create a feasible sequence i.e. job 0 is always scheduled as first and job MaxJ is always scheduled at the end of every machine sequence. Additionally, to make the adaptation previously mentioned, constraints (2), (3), (18) and (19) were modified from the originals with the incorporation of a "≥" instead of the traditional "=" to allow that job 0 and MaxJ were assigned and evaluated in more than one sequence i.e. more than one machine. Finally, it is necessary to recall that this MILP only can solve instances to optimality when the number of machines at each stage is less or equal than the total number of jobs.

4.2. Genetic algorithms (GA):

Since the flexible open-flow shop scheduling problem (FOFS) is NP-hard, techniques for finding optimal solutions are unlikely to exist[8]. Thus, heuristics and metaheuristics are developed to seek good solutions in reasonable computational time.

According to Frunza [31], the genetic algorithm is an evolutionary and iterative method that emulates a reproductive mechanism found in nature, where solutions are build based on biological behavior. To reach a solution, Pose [32] proposed a series of steps, starting with an initial population formed by individuals. A single individual (chromosome) represents a solution of the problem, then, new individuals are generated according to the "natural selection" principle, aiming to obtain better solutions.

The basic iteration for the GA proposed by Nearchou [33] is presented as follows:

Generation of initial population.
Evaluate population (fitness).
WHILE** (termination criteria not reached) **DO
Select two parents using a defined probabilistic strategy.
Apply crossover operator to the two parents so offspring population is created.
For each offspring apply a mutation operator with a defined probability.
Evaluate the fitness of new individuals.
Replace parents in the global population with the created offspring.

In this section four solution techniques were designed considering the interaction between the FOS and FFS. These solution techniques are based on a traditional genetic algorithm (GA) but considering certain variations that allow the synergy between both environments. These variations are presented for FOS and FFS, then each solution technique is described by showing the general interaction of both workshops.

4.2.1 Chromosome representation:

Matta [23] describes the chromosome as an encoding of the solution. Traditionally, this chromosome can be presented as a string of binary numbers, random numbers, among others. In scheduling, the traditional way to represent the solution is through a permutation of jobs and/or stages. The chromosome representation can vary depending on the features of the scheduling environment, therefore is necessary to describe the corresponding chromosome encoding for the FOS and FFS.

FOS:

One of the most common chromosome representation for the FOS is the operation-based chromosome with *n x m* genes (n being the number of jobs and m the number of stages) in which each gene indicates the job and the stage [2],[23]. The order in which each gene is presented displays the sequence and route for each job in the problem.

The proposed chromosome for the FOS is particularly different from the ones presented before, but in essence it represents both the sequence and the route for each job. This representation can be named as a dual chromosome, influenced by the one presented by Zambrano et al [34], given that is shaped by two parts (P1 and P2): P1 being an array that represents a permutation for job sequencing and P2 being a matrix of (number of FOS stages *x* number of jobs) in which each column contains the route for the job located in the same position for P1 and each row represents the sequence of stages that each job must follow to be totally processed.

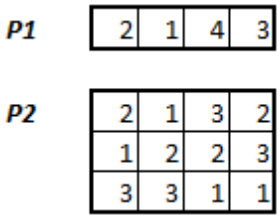

Figure 2. Chromosome representation for the FOS

The previous chromosome example represents a problem conformed by 4 jobs and 3 FOS stages. P1 shows the jobs processing sequence and P2 shows the processing route for each job presented in P1. For instance, job 2 must be the first job to be scheduled thus, its route through FOS must begin at stage 2 followed by stage 1 and concluding at stage 3. This same structure applies for the rest of the jobs in P1's permutation and their routes in P2.

Notice that this dual chromosome always returns a feasible schedule. The representation tracks each job's route throughout the schedule but the machine assignation of a job at each stage entirely depends on the decoding or assignation method used.

It can be outlined that this is the general chromosomal representation, but it may slightly vary between the proposed solution techniques.

FFS:

Considering the chromosomal representation presented in literature[35],[36] for FFS, the traditional chromosome consists on an array of n (number of Jobs) that represents the processing sequence i.e. the first element is the first job to be processed the second element is the second job to be processed and so on. Traditionally, this sequence remains the same for the rest of the FFS stages.

This chromosome is used in the literature generally to represent an FFS [35],[31]. Unlike FOS, the processing route of each job is already known thus this chromosome contains the necessary information to schedule the jobs. This representation is presented as follows:

1	2	3	4	5	6	7	…	Jobs

Figure 3. Chromosome representation for the FFS

4.2.2 Initial population

Generally, as Jitti et al [8] and Frunza[31], the initial population is generated randomly from the solution space, where each individual of this population must represent a solution of the problem[32].

FOS:

Given the previous chromosomal representation for FOS, feasibility takes an important place in the random generation of the initial population due to its dual structure and permutation framework. Hence, a random generation strategy has been developed in order to achieve feasibility without compromising computational time.

The population generation strategy is described in the following steps:

1. *Creation of the array (P1*) with the same size of P1*
2. *Fill in P1* with random numbers U[0,1]*
3. *Create a reference array for P1 sequence initially composed with the total number of jobs and set J as its current size.*
4. ***FOR** i=1 **TO** P1* length*
   Be (P1[i] x J) a position of the reference array.*
   The job placed in this position in the reference array is now placed in P1[i] and removed from the reference array.
   J=J-1
   END FOR

The same procedure is applied for each column in P2 i.e. for each job's route. This strategy is graphically described as it is shown in the next figure:

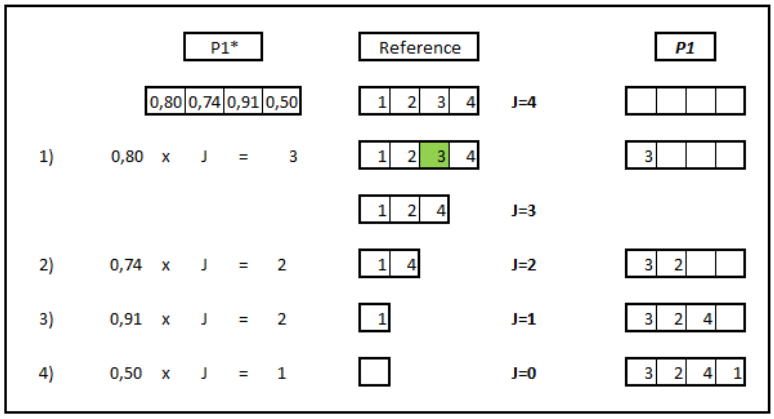

Figure 4. Generation of initial population for FOS

The presented metodology ensures two important aspects. First, feasibility is guaranteed inasmuch as the permutation framework is respected, thus no jobs are going to be repeated in P1 or stages in P2s columns. Second, computational agility is not given up as a result of the verification avoidance in order to ensure permutation's feasibility. This affair takes relevance when big size *instances* are executed e.g. random generation of 100 jobs sequence permutation.

The number of individuals in the population is determinated through experimentation with different instance sizes relying on the makespan result. Finally, the population's size for all solution techniques was defined by the one that experimentally allowed the obtention of a better makespan.

FFS:

The initial population for the FFS was generated following the same procedure presented for the FOS and it is illustrated in figure 4.

Finally, the population's representation is allocated in a matrix (Population size X Number of jobs) and is illustrated in the following figure:

Individual (1)	2	6	7	8	10	3	1	9	4	5
Individual (2)	3	4	6	1	9	10	8	2	5	7
	:	:	:	:	:	:	:	:	:	:
	:	:	:	:	:	:	:	:	:	:
Individual (10)	8	3	1	7	5	2	10	4	9	6

Figure 5. FFS population

4.2.3 Parents' selection strategy

Pose [32] defines the selection algorithms as in charge of choosing what individuals of the population may have the possibilities of reproduction and which ones not. There is plenty of criteria and algorithms in literature about this topic. Most common algorithms are roulette selection and tournament selection (either probabilistic or deterministic)[31].

Roulette selection, or Monte Carlo selection created by Blickle and Thicle [37], is a strategy in which each individual is assigned with a portion of the roulette in function of its objective function, giving the best suited individuals a greater portion of the roulette. To select the individual is necessary to generate a random number U[0,1] and choosing the individual placed in that position (by cumulative portions). Tournament selection works by selecting K individuals of the population and picking the most suited ones in terms of objective function (deterministic) or defining a probability *P* then, for each individual generate a random number U[0,1] and if this probability is greater than *P* the individual is chosen (probabilistic). The performance of both algorithms entirely depends on the GA requirements.

FOS:

The parents' selection strategy chosen was roulette selection because it is attempted to pick the most suited individuals for reproduction process but giving all population the possibility to be a parent, thus a more open diversification strategy is pursued. Since the problem at hand focuses on makespan minimization, the assignation of the roulette's portion must be greater for the individuals with less objective function, so this procedure partially changes from the one presented in literature, but the selection itself remains the same for all solution techniques. Two parents are selected in every iteration for all the solution techniques.

FFS:

The selection strategy for the FFS is made following the same procedure presented for the FOS.

4.2.4 Genetic operators

The performance of a genetic algorithm depends very much on the selection of the proper genetic operators. The most frequently used genetic operators are reproduction, crossover, and mutation. The reproduction operator emulates the

process of natural selection, as it was described in the previous section. Hence, this subsection is focused on crossover and mutation operators applied to the solution techniques for the FOFS scheduling problem.

Nearchou [33] presents the crossover as a mechanism that allows two individuals to exchange portions of their structure based on specific(s) cross point(s) given for a specific probability. The result of this combination is two new chromosomes aiming to obtain the best characteristics of both parents. Additionally, mutation which can be recognized as a diversification strategy, is applied to a selected child altering randomly its structure.

The use of these genetic operators on the proposed chromosomal representation could lead to unfeasible solutions, thus, repairing mechanisms must be considered. This perturbation to the generated offspring yields to performance reduction so mutation operator can help to increase performance.

FOS:

Typically, crossover operators, are applied to a one array chromosomal representation. Given that the developed solution techniques consider a dual chromosome (in some cases just the matrix P2) it is necessary to adapt these operators, although due to their complexity, some of them may not apply. Therefore, in order to select one crossover operator, the *One-Point Crossover*, *Two-Point Crossover* and *Position-Based Crossover* were experimentally tested and compared. Finally, the *One-Point Crossover* was the one used in each of the proposed solution techniques.

The general representation of the *One-Point Crossover* is presented in image 8.1. The adaptation of this operator for the FOS is made considering the dual chromosome although in some of the proposed solution techniques just matrix P2 is used. In the background array P1 is always used as a reference point of the traditional *One-Point Crossover*.

The applied crossover operator is presented as follows:

1) *Randomly generate a cross point C with 1 < C < Total number of jobs*
2) *Copy all first parent's genes in array P1 that are placed before C into first child's array P1; do the same procedure for matrix P2 maintaining the exact same route that corresponds to each job.*
3) *Identify and mark the missing elements (jobs) in child's P1*
4) *Create a reference array and place the missing jobs in the order in which they are located in the second parent's array P1.*
5) *Making use of P1* and the procedure presented in initial population section for generation of random sequences, fill the rest of child's P1.*
6) *Copy the routes located in the second parent for the same jobs that were placed in the previous step.*
7) *Create the second child using the same method but reversing the order of the parents.*

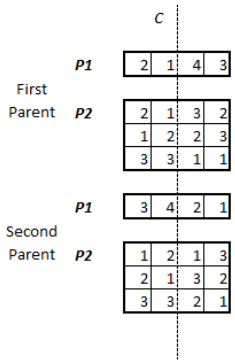

Figure 6.1. One-Point crossover representation for FOS.

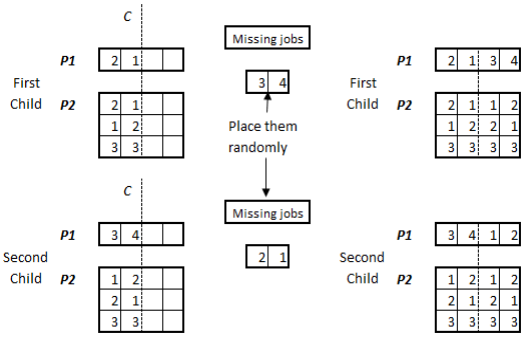

Figure 6.2. One-Point crossover representation for FOS

Notice that this crossover considers feasibility as a part of the procedure, thus every generated child is going to achieve a feasible schedule. In the other hand, the selected mutation operator is the *random exchange mutation*[33] where two genes of a single chromosome are randomly selected to be exchanged. To adapt this operator to the dual chromosome, one additional step is required. This step, implies the random selection of a job, and then to apply the same process described to the route of the selected job as illustrated in figure 7.

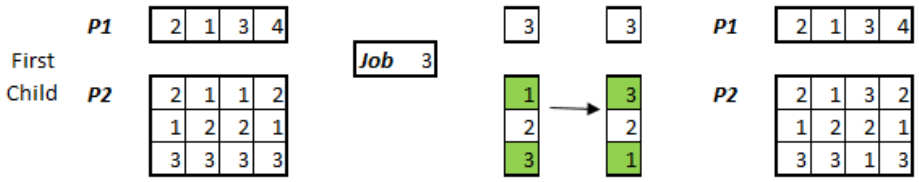

Figure 7. Mutation operator for FOS

For each generation of the genetic algorithm, the offspring may or may not be selected for mutation; this is defined by a probability parameter *P*. For each child a random number U[0,1] is generated, if it is less than *P* then the child is submitted to mutation. The parameter *P* for the solution techniques in FOS is 0.1.

FFS:

According to Frunza[31], selected individuals are recombined to generate the next generation. These crosses were performed non-destructively [32]. In each generation, the two individuals with the best objective function are chosen regardless of whether they are parents or offspring. In addition, the *One-Point Crossover*, *Two-Point Crossover* and *Position-Based Crossover* were experimentally tested and compared. Finally it was decided to cross over by the *One-Point Crossover* method (image 8.1 and 8.2), where the cross point of the selected individuals consists of selecting a random number U[1, total Jobs-1]. An offspring is compound from parent 1 until cross point and from parent 2 for the rest of the child. A second child is also generated using the same method but reversing the order of the parents. Repairing mechanisms are applied when feasibility issues are presented.

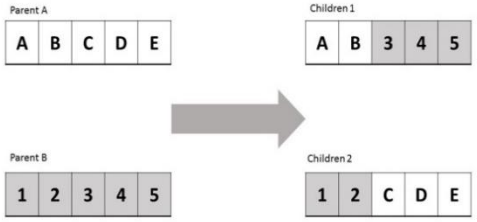

Figure 8.1. One-Point crossover for FFS

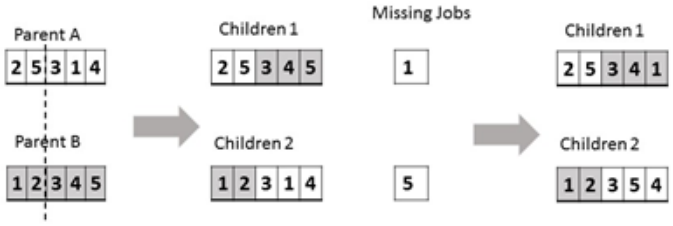

Figure 8.2. One-Point crossover representation for FFS with reparation mechanism

For the realization of the mutation operator it is necessary to evaluate a random number U[0,1] for each child. If this value is less than *P (*mutation probability), two jobs are exchanged randomly. This mutation process is illustrated in figure 9 Considering that a high mutation ensures that the chromosome is diversified finding new possible solutions, the parameter *P* for the solution techniques in FFS is 0.6 [38].

Figure 9. Mutation operator for FFS

4.2.5 Objective function evaluation

The evaluation of the objective function plays a fundamental role for the integral approach of the problem. Although, the characteristics of the genetic algorithm have been defined for the two environments separately, the evaluation of the objective function is composed by the FFS and FOS schedules. Therefore, two different heuristics were developed in order to organize and create the final schedule, based on the information provided from both environments. The makespan result quantify the solution from FOS and FFS conjointly i.e. both solutions impact the final result.

Scheduling heuristic 1:

This heuristic organizes the jobs considering the processing route and job sequence provide from FOS. The FFS part considers the job sequence provided from the genetic algorithm of the FFS and the release times obtained at the end of the FOS part. The machine assignment is made by choosing the one that has the minimum release time to process the next job on that stage. This heuristic aims to ease the scheduling and facilitates diversification by relying exclusively on the solutions presented by the genetic algorithm in each type of environment, therefore it is a quick method to calculate the makespan.

The heuristic pseudocode is presented as follows:

*Take **P1** and **P2** as job sequence and processing route parameters from FOS*
*Take **FSE** as job sequence parameter from FFS*
*Define **C** as Partial completion time for every job*
*Define **TE** as minimum release time for every stage*
*-------*OPEN*---------*
***FOR** i=1 **TO** TotalStagesOpen*
    ***FOR** j=1 **TO** TotalNumberOfJobs*
    Select the job that corresponds to P1(j)
    Select the stage that corresponds to P2(i, P1(j))
        ***IF** TE(Stage) <= C(job)*
          Select the first machine and schedule the job
        Else
        *Select the machine that has the minimum release time of the stage **TE(stage)** and schedule the job*
        End if
    Update C(job) as completion time at the stage
    Update TE (stage) as the minimum release time of all machines in the stage
     ***NEXT** j*
***NEXT** i*

*-------*FLOW*---------*
***FOR** i=FirstStageFlow **TO** TotalStagesFlow*
    ***FOR** j=1 **TO** TotalNumberOfJobs*
    Select the job that corresponds to FSE(j)
    Take the stage as i
        ***IF** TE(Stage) <= C(job)*
        Select the first machine and schedule the job

Else
Select the machine that has the minimum release time of the stage ***TE (stage)*** *and schedule the job*
End if
Update C(job) as completion time at the stage
Update ***TE (stage)*** *as the minimum release time of all machines in the stage*
NEXT *j*
NEXT *i*
Makespan = Max C
END

Scheduling heuristic 2 :

This procedure is based on the schedule generation heuristic for open shop proposed by Anard and Ellur [39], where no further left-shift of jobs is possible, ensuring that the fastest jobs are scheduled first considering its processing route. This is also known as shortest processing time rule (SPT). Machine assignment is done by the same methodology of the previous heuristic. This methodology aims to find a more efficient job sequence for the final schedule by comparing the processing and release times of each job throughout the makespan calculation, making it a non-delay-SPT strategy.

The heuristic pseudocode is presented as follows:

Take ***P2*** *as processing route parameter from FOS*
Define ***H*** *as matrix containing the processing route for each job in FFS*
Define ***S*** *as set of stages with no predecessors*
Define ***C*** *as Partial completion time for every job*
Define ***TE*** *as minimum release time for every stage*
Let A be TotalStagesOpen
Let B be P2
For each job Let S be the first stage to process stablished on B
WHILE *Counter <= TotalNumberOfJobs * A* ***DO***
Select the job with minimum release time + Processing time on its stage S
Let S be the stage of the selected job to schedule
IF *TE(Stage) <= C(job)*
Select the first machine and schedule the job
Else
Select the machine that has the minimum release time of the stage ***TE(stage)*** *and schedule the job*
End if
Update C(job) as completion time at the stage
Update ***TE (stage)*** *as the minimum release time of all machines in the stage*
Udpate S(job) as the next stage for the job stablished on B
Udpate Release time of the job as C(job)
Counter = Counter + 1
LOOP
Let A be TotalFlowStages
Let B be H
REPEAT THE PREVIOUS LOOP WITH THE NEW VARIABLES
Makespan = Max C
END

4.2.6 Termination / stop criteria

The termination criteria for the solution techniques is defined with a fixed number of generations. This number was defined experimentally considering different instance sizes, obtaining that, usually best solutions are not improved with more than 6000 generations.

4.2.7 Solution techniques

The final design of the solution techniques are composed by the methodologies and strategies previously described. Each solution technique aims to solve the FOFS by combining different scheduling heuristics. Table 1 resumes the main strategy of each solution technique as well as its implication on each particular environment (FOS and FFS). Solution techniques' files can be found in attachments 1 to 4.

Solution technique / Characteristics	FOS	FFS	GLOBAL CHARACTERISTICS
Solution Technique 1 (ST1)	The FOS used the genetic algorithm with dual chromosome generating the job sequence array(P1) and processing route matrix (P2). The objective function evaluation is made by **scheduling heuristic 1**	The FFS used the genetic algorithm generating the Job sequence array for this part. The objective function is evaluated by **scheduling heuristic 1** considering release times at the end of FOS part.	This solution technique aims to solve the problem by considering a diversification strategy since both parts used its complete genetic algorithm. The best solution of each part is transmitted to the other
Solution Technique 2 (ST2)	The FOS used the genetic algorithm generating only the processing route matrix (P2). The objective function evaluation is made by **scheduling heuristic 2**	The FFS used the genetic algorithm generating the Job sequence array for this part. The objective function is evaluated by **scheduling heuristic 1** considering release times at the end of FOS part.	This solution technique aims to solve the problem focusing on a possible dominance of the FFS part over FOS part.
Solution Technique 3 (ST3)	The FOS used the genetic algorithm generating only the processing route matrix (P2). The objective function evaluation is made by **scheduling heuristic 2**	The FFS does not used the genetic algorithm. The objective function is evaluated by **scheduling heuristic 2** considering the release times and partial completion times at the end of FOS part	This solution technique aims to solve the problem by considering the performance of the **scheduling heuristic 2**, which leads to less diversification due to its constructive structure.
Solution Technique 4 (ST4)	The FOS used the genetic algorithm with dual chromosome generating the job sequence array(P1) and processing route matrix (P2). The objective function evaluation is made by **scheduling heuristic 1**	The FFS does not used the genetic algorithm. The objective function is evaluated by **scheduling heuristic 2** considering the release times and partial completion times at the end of FOS part	This solution technique aims to solve the problem focusing on a possible dominance of the FOS part over FFS part.

Table 1. General comparison between solution techniques.

Pseudocodes of each solution technique are presented below:

Solution technique (1):

Generation of initial population for FOS
Generation of initial population for FFS
*Set the first individual of the FFS population as '**IndF**'*
*Evaluate the population with **scheduling heuristic 1** giving **IndF** and FOS population as parameters*
WHILE** (termination criteria not reached) **DO
FOS iteration:
Select two parents using the defined strategy (roulette selection).
Apply FOS crossover operator (One-Poist Crossover) to the two parents so offspring is created.
Define whether to apply a FOS mutation operator to each child with the defined probability.
*Evaluate the fitness of new individuals with **scheduling heuristic 1** giving **IndF** and FOS population as parameters*
Replace parents in the global population with the created offspring
*Set the best individual of FOS population as **IndO**.*
FFS iteration:
Select two parents using the defined strategy (roulette selection).
Apply FFS crossover operator (One-Point Crossover) to the two parents so offspring is created.
Define whether to apply a FFS mutation operator to each child with the defined probability.
*Evaluate the fitness of new individuals with **scheduling heuristic 1** giving FFS population and **IndO** as parameters*
Replace parents in the global population with the created offspring
*Set the best individual of FFS population as **IndF**.*
END

Solution technique (2):

Generation of initial population for FOS
Generation of initial population for FFS
*Set the first individual of the FFS population as '**IndF**'*
*Evaluate the population with a combination of **scheduling heuristic 2** for FOS population and **scheduling heuristic 1** for **IndF**.*
WHILE** (termination criteria not reached) **DO
FOS iteration:
Select two parents using the defined strategy (roulette selection).
Apply FOS crossover operator (One-Poist Crossover) to the two parents so offspring is created.
Define whether to apply a FOS mutation operator to each child with the defined probability.
*Evaluate the fitness of new individuals with a combination of **scheduling heuristic 2** for FOS population and **scheduling heuristic 1** for **IndF**.*
Replace parents in the global population with the created offspring
*Set the best individual of FOS population as **IndO**.*
FFS iteration:
Select two parents using the defined strategy (roulette selection).
Apply FFS crossover operator (One-Point Crossover) to the two parents so offspring is created.
Define whether to apply a FFS mutation operator to each child with the defined probability.
*Evaluate the fitness of new individuals with **scheduling heuristic 1** for FFS population and **scheduling heuristic 2** for **IndO***
Replace parents in the global population with the created offspring
*Set the best individual of FFS population as **IndF**.*

END

Solution technique (3):

Generation of initial population for FOS

Evaluate the population with ***scheduling heuristic 2*** *using just Matrix P2 of FOS chromosomal representation.*
WHILE *(termination criteria not reached)* ***DO***
FOS iteration:
Select two parents using the defined strategy (roulette selection).
Apply FOS crossover operator (One-Poist Crossover) to the two parents so offspring is created.
Define whether to apply a FOS mutation operator to each child with the defined probability.
Evaluate the fitness of new individuals with ***scheduling heuristic 2*** *using just Matrix P2 of FOS chromosomal representation.*
Replace parents in the global population with the created offspring.
END

Solution technique (4):

Generation of initial population for FOS

Evaluate the population with a combination of ***scheduling heuristic 1*** *for FOS population and* ***scheduling heuristic 2*** *for FFS*
WHILE *(termination criteria not reached)* ***DO***
FOS iteration:
Select two parents using the defined strategy (roulette selection).
Apply FOS crossover operator (One-Poist Crossover) to the two parents so offspring is created.
Define whether to apply a FOS mutation operator to each child with the defined probability.
Evaluate the fitness of new individuals with a combination of ***scheduling heuristic 1*** *for FOS population and* ***scheduling heuristic 2*** *for* ***FFS.***
Replace parents in the global population with the created offspring
END

4.3. Instances

To measure the effectiveness of the solution techniques, it is necessary to use instances that represent the problem at hand[23]. FOFS instances were created and adapted from instances presented in literature for different shops environments. The characteristics and parameters needed to represent the problem at hand are:

1. *Total jobs*
2. *Total stages*
3. *Open Shop stages*
4. *Flow Shop Stages*
5. *Machines per station*
6. *Set up time by stage*
7. *Process time of the job in each stage*

Generated instances were divided in three categories: small, medium and large. Small instances were defined by evaluating if it was possible to reach a feasible solution in less than one hour with the mathematical model. Experimentally, it was found that the more jobs and / or stages are added, the complexity of the problem increases and so the execution time. Overall, in was not possible to obtain a feasible solution with the MILP within an hour for instances with more than 5 jobs and stages did not meet the criteria.

Medium instances consider between 6 and 25 jobs and between 6 and 10 stages. If any instance exceeds any of the previous ranges, it is classified as large instance.

Small instances were randomly generated with the following criteria.

1. *Total Jobs are a random number U [2,6]*
2. *Total Stages are a random number U [3,6]*
3. *FOS Stages are a random number A U [2, round up (total stages/2)]*
4. *FFS Stages are Total stages - A*
5. *Processing time is a random number U [1,50]*
6. *Set up time per stage is a random number U [4,16]*
7. *Machines per stage are a random number U [1,3]*

To test the proposed solution techniques, 29 medium and large instances were adapted from the ones found in literature [1]. These instances, which originally correspond to the FFS environment with sequence dependent setup times, were adapted to the features of the problem. The number of stages for each environment i.e. the proportion of FOS and FFS stages generated for each instance, was calculated with the random criteria proposed by Matta [23]and Mejia G et al[2].

1. *Stages FOS is a random number B U [2, total stations-2]*
2. *Stages FFS is total stations-B*
3. *Setup per station is* $\sum_{j=1\ j\in J}^{|J|} Set\ up_{j,i} \quad \forall\, i$

Table 2 summarizes the main features of the small, medium and large instances in terms of number of jobs and stages.

Range	No	Name of instance	# Jobs	Stages	Range	No	Name of instance	# Jobs	Stages
Small	1	ModeloM 1	3	O2 F4	Large	21	n50 m4 15	50	O2 F2
	2	ModeloM 2	2	O2 F2		22	n50 m4 31	50	O2 F2
	3	ModeloM 3	2	O2 F4		23	n50 m4 48	50	O2 F2
	4	ModeloM 4	3	O3 F2		24	n50 m8 11	50	O3 F5
	5	ModeloM 5	4	O2 F3		25	n50 m8 31	50	O4 F4
	6	ModeloM 7	3	O2 F2		26	n50 m8 62	50	O3 F5
	7	ModeloM 8	3	O2 F4		27	n50 m8 79	50	O3 F5
	8	ModeloM 10	2	O2 F2		28	n80 m4 21	80	O2 F2
	9	ModeloM 11	3	O2 F3		29	n80 m4 24	80	O2 F2
	10	ModeloM 12	3	O2 F2		30	n80 m8 8	80	O4 F4
	11	ModeloM 13	5	O2 F1		31	n80 m8 9	80	O5 F3
	12	ModeloM 14	2	O2 F1		32	n80 m8 15	80	O3 F5
Medium	13	n20 m4 107	20	O2 F2		33	n80 m8 20	80	O3 F5
	14	n20 m4 122	20	O2 F2		34	n120 m4 16	120	O2 F2
	15	n20 m4 144	20	O2 F2		35	n120 m4 2	120	O2 F2
	16	n20 m4 153	20	O2 F2		36	n120 m4 21	120	O2 F2
	17	n20 m8 12	20	O6 F2		37	n120 m4 6	120	O2 F2
	18	n20 m8 21	20	O4 F4		38	n120 m8 11	120	O2 F6
	19	n20 m8 58	20	O2 F6		39	n120 m8 4	120	O5 F3
	20	n20 m8 60	20	O2 F6		40	n120 m8 7	120	O4 F4
						41	n120 m8 8	120	O5 F3

Table 2. Set of instances to be executed in the solution techniques.

NOTE: The notation O and F represent the number of stages for FOS and FFS, for example, O2F2 is referred to instances with two FOS stages and two FFS stages. The total number of stages for a single instance corresponds to the sum of these values.

4.4. Validation of the solution techniques

At first, the validation of the solution techniques is made by comparing with the results obtained from the mathematical model for small size instances. Nevertheless, for medium and large-scale problems, for which optimal solutions cannot be obtained with the mathematical model, Nahavandi et al [40] outlines the need to create methodologies for evaluating the quality of the solutions. One of these methodologies consists in the development of a lower bound (LB).

To propose a LB for the FOFS scheduling problem it is necessary to ensure that it is less or equal than the optimal solution. To guarantee that characteristic, it is necessary to evaluate the features of the problem (processing times, set up times, stages, machines and jobs) and estimate a minimum value for the makespan. The main goal is to obtain a tight LB that could be close to the optimal schedule and validate the performance of the solution techniques with more precision.

Therefore, three lower bounds are proposed. The maximum of these lower bounds is selected to test the performance of the solution techniques.

Lower Bound 1:

This lower bound is adapted from a stage-based lower bound for FFS proposed by Nahavandi et al [40]. It is calculated as the sum of four components: (i) the workload of the bottle neck stage (BN) in the FFS part (equation 1), (ii) the estimated time for a job to start in the BN stage (including all stages before BN), (iii) an estimated time for a job to start in the last job on sequence at stages following BN; and (iv) the waiting time of every FFS stage before the bottle neck

To identify the bottle neck stage, the following expression is presented:

$$BN: \text{the stage } t \in T2 \text{ such that} \qquad min_t \left\{ \frac{1}{M_t} * \left(\sum_{j=1}^{J} P_{jt} + SU_t \right) \right\}$$

Equation 1. Identification of the bottleneck stage

Therefore, the equation for lower bound 1 is:

$$LB1 = \left[\frac{1}{M_{BN}} * \left(\sum_{j=1}^{J} P_{j(BN)} + SU_{BN} \right) \right] + min \left\{ \sum_{t=1,t\in T}^{BN-1} \sum_{j=1}^{J} P_{jt} + SU_t \right\} + min \left\{ \sum_{t=BN+1,t\in T}^{T} \sum_{j=1}^{J} P_{jt} + SU_t \right\} + \sum_{t=1,t\in T2}^{BN-1} min_j \{P_{jt}\}(M_{BN} - M_{BN-1})$$

Equation 2. Mathematical expression for lower bound 1

Lower bound 2:

This lower bound is based on the fact that the maximum completion time is given by the last stage i.e. the last FFS stage. This LB is calculated as the sum of two components: (I) the workload of the last stage, considering the total number of machines in that specific stage; and (ii) an estimated time for a job to start in the last stage (including all stages before last FFS stages). The equation that calculates the proposed lower bound is:

$$LB2 = \left[\frac{1}{M_{|T|}} * \left(\sum_{j=1}^{J} P_{j|T|} + SU_{|T|}\right)\right] + min\left\{\sum_{t=1,t\in T}^{|T|-1}\sum_{j=1}^{J} P_{jt} + SU_t\right\}$$

Equation 3. Mathematical expression for lower bound 2

Lower bound 3:

This lower bound is an estimation based on the premise that the makespan cannot be less than the time given by the job with the maximum total processing time. The expression that represents this affirmation is presented as follows:

$$LB3 = max_j\left\{\sum_{t=1,t\in T}^{|T|} P_{jt} + SU_t\right\}$$

Equation 4. Mathematical expression for lower bound 3

Finally, to select the LB to assess the solution techniques the following expression is applied:

$$LB = max\{LB_1, LB_2, LB_3\}$$

Equation 5. Selection of the final lower bound

To finally proceed with the general validation, all instances were executed on each solution technique and compared with the results from the mathematical model for small instances and the resultant LB for medium and large instances. To be considered as a valid solution, the result of the solution technique for each instance must be greater or equal than the mathematical model or resultant LB whether it applies. The obtained results, gathered in attachments 5 and 6, proves that the four solution techniques fulfill the requirements to give feasible solutions for different instances of the FOFS problems. The comparison and impact of the results for each solution technique is analyzed in section 6.

5. Engineering design

5.1 Design statement

This project developed solution techniques based on an integral approach that minimizes makespan for the flexible open-flow shop scheduling problem. The integral approach means that the solution techniques took into consideration the interaction between the two shop environments altogether. Initially, a mathematical model was formulated, in order to identify the principal constraints, decision variables and parameters involved. Furthermore, four solution techniques were designed based on the genetic algorithm metaheuristic and scheduling heuristics adapted from literature.

To verify the performance, in terms of maximum completion time, small, medium and large instances were executed. These instances were generated and/or adapted from already existent set of FFS instances found in literature. Results obtained with the solution techniques were compared with the feasible solutions obtained with the mathematical model within a time limit of one hour for small instances. To test the performance of the techniques for the rest of the instances (medium and large) a set of lower bounds were designed based on lower bounds proposed in literature. Additionality, a comparison was made with flow shop instances created by Mattfeld and Vaessens[41], and adapted to an open-flow shop by Mejia G et al [2].

5.2 Design Process

The design process is described in section 4.

5.3 Design requirements and performance test

The proposed design requirements validate the efficiency and reliability of the presented solution techniques when solving different sizes of the problem. In general terms, each solution technique must guarantee feasible solutions for the at hand and must handle small, medium and large instances in reasonable computational time and must be adaptable to different parameters settings i.e. Processing times, setup times, number of jobs, number of stages and number of machines per stage.

At first, feasibility is guaranteed with the use of the chromosomal representations and genetic operators in addition with the scheduling heuristics ensuring that each job is processed at every stage, with its corresponding processing time, in a sequence structure (see section four). In the other hand, feasibility was also verified by means of the realization of the Gantt diagram, which was depicted for the final schedule obtained for each of the tested instances. Finally, another feasibility procedure was made by verifying the processing and set up times for each job in the resulting schedules.

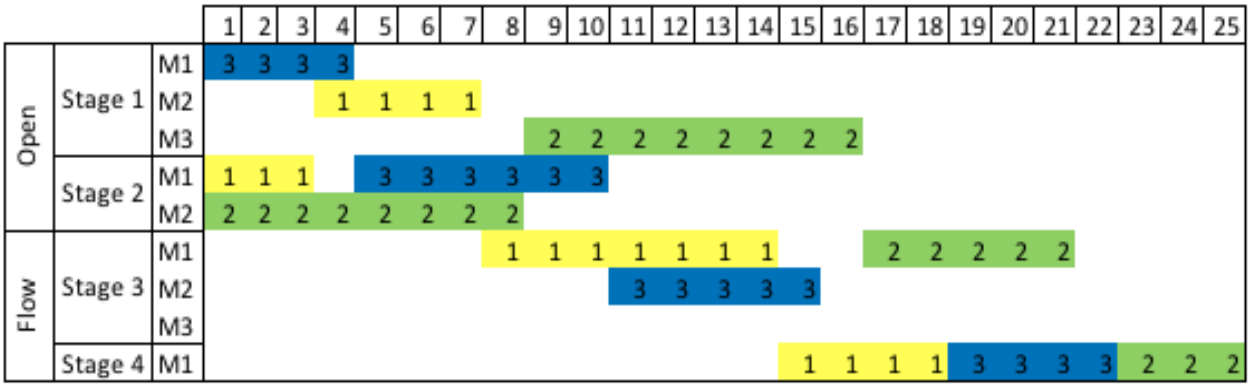

Figure 10. Gantt Diagram for a hypothetical solution.

Figure 10 represents a possible schedule for the FOFS. Each job is processed on the flow shop stages after complete the open shop part, The flexibility is illustrated by the different number of machines per stage and all jobs are schedule in structure sequence without overlap.

5.4 Design restrictions.

The following restrictions were taken into consideration:

- Given the NP-hard nature of the problem, solution techniques must have a reasonable execution time for large instances.
- Processing and setup times on machines are deterministic.
- Each stage has identical machines i.e. same processing and setup times.
- Machine setups are sequence independent meaning that the setup time is the same and independent of the sequence given for the schedule.
- The solution techniques must adapt to the existence of two workshop environments, flexible open shop and flexible flow shop.

Considering the scope of this project, a time reference of one hour (3600 s) was considered in order to validate that large instances were solved in reasonable computational time. As was described in section 4, instances were generated considering deterministic setup and processing times, assuming identical machines in each stage and that setup times doesn't depend on the sequence. As was already mentioned, proposed solution techniques consider the interaction of both environments (see table 1).

5.5 Norms and standards

The methodology used in this project is based on the DMAIC technique, ISO 13053-1:2011 that describes a methodology for the business improvement known as Six Sigma. [42]. Considering the problem complexity, the solution techniques must be rigorous between input-output data relation. DMAIC provides a data driven strategy for improving processes meaning that this methodology is very suitable for this project. The general strategy is developed in five phases: Define, Measure, Analyze, Improve and Control. The Define and Measure phases summarizes the principal requirements and restrictions, leading the developing of the mathematical model as an essential part to understand the problem and the selection of the genetic algorithm as the base metaheuristic strategy to design the solution techniques. To ensure that the rest of the methodology applies in the design of every solution technique, all instances were executed under the same PCs characteristics and conditions in order to reduce biases between obtained results from each solution technique. This allows improvement opportunities to be focused only on the features of each technique. Finally, results obtained from each technique were compared to identify attributes of each scheduling methodology.

6. Results

6.1. Impact measurement

To evaluate the effectiveness of the solution techniques the methodology proposed by Matta[23] is used. In this methodology, the performance is analyzed in three different dimensions: solution quality, solution speed and shop performance. This methodology ensures to cover up all technical and statistical aspects of the results obtained from the solution techniques.

The solution quality is measured in terms of average percentage difference from the mathematical model and the LB and the comparison of best makespan found by each solution technique. The speed is measured in terms of average running time and number of generations needed to find the best solution. Shop performance is measured by Average percentage difference between each solution technique on different type of instances and the efficiency given by the running time and number of generation for different instance sizes (medium and large). All solution techniques were code on Visual basic for application (VBA) and executed on HP Z240 series core i7 7700 16Gb Ram PC .These performance measures are shown below.

Solution quality

Solution Technique	Mean Dif %	Min Dif %	Max Dif %
# 1	1,79%	0%	17,89%
# 2	1,79%	0%	17,89%
# 3	2,46%	0%	17,89%
# 4	2,08%	0%	17,89%

Table 3. Percentage difference for small instances Vs mathematical model

Solution Technique	Mean Dif %	Min Dif%	Max Dif%
# 1	13,12%	0,00%	67,00%
# 2	10,18%	0,00%	43,82%
# 3	8,76%	0,00%	28,24%
# 4	6,33%	0,00%	28,76%

Table 4. Percentage difference for Medium-Large instances Vs LB

Table 3 and 4 presents the average percentage difference (Mean Dif%), the minimum percentage difference (Min Dif%) and Maximun percentage difference (Max Dif%) for all its corresponding size instances.The results display display that every solution technique found at least one optimal solution (0% difference). For small instances, on average each solution technique is below 5% being ST3 the one with higher difference (2.46%) and ST1 the one with a lower difference (1,79%) from the optimal. For medium and large size instances, the maximum mean difference is 13.12% given by ST1. Also, the difference between Mean dif% and Min dif%, for all techniques, is shorter than Mean and max% difference. This experimentally proof that there are more results near to the minimum difference found.

Frequency distribution

This indicator categorizes the number of instances, for each solution technique, that reach optimal solutions, close to optimal solution (Dif<=5%) and not close to the optimal solution. These criteria are explained below.

Solution Technique	Optimal	Dif <=5%	Dif >5%
# 1	9	0	1
# 2	9	0	1
# 3	7	2	1
# 4	7	2	1

Table 5. Frequency distribution for small instances.

Solution Technique	Optimal	Dif <=5%	Dif >5%
# 1	2	8	19
# 2	3	12	14
# 3	2	12	15
# 4	8	9	12

Table 6. Frequency distribution for Medium-Large instances.

Since the optimal solution is unknown for FOFS problem, LB is used for makespan comparison. The results that are greater than 0% and less or equal than 5% are highly probable to be close to the optimal solution. Additionality, the results that have exactly 0% difference against LB, confirm that an optimal solution was found. The results shown in table 5 and table 6 display the number of instances that fulfills these criteria. For small instances, all solution techniques display good solutions in 90% of tested instances. Nevertheless, ST1 and ST2 reached optimality on 90% in comparison to the 70% obtained with ST3 and ST4. For medium and large instance sizes, solution techniques ST2 and ST4, reached a percentage difference less or equal than 5% in more than 50% of tested instances. More specifically, ST2 and ST4, reached an optimal solution (gap between solution techniques and lb is 0%)in 10,3% and 27,6% respectively of tested instances. Additionally, for 2 instances the mathematical model could not find a feasible solution in a time limit of one hour i.e. do not meet the requirement explained in section 4.3. These 2 instances were not taken into consideration for table 3 The specific results can be found in attachment 5 and 6.

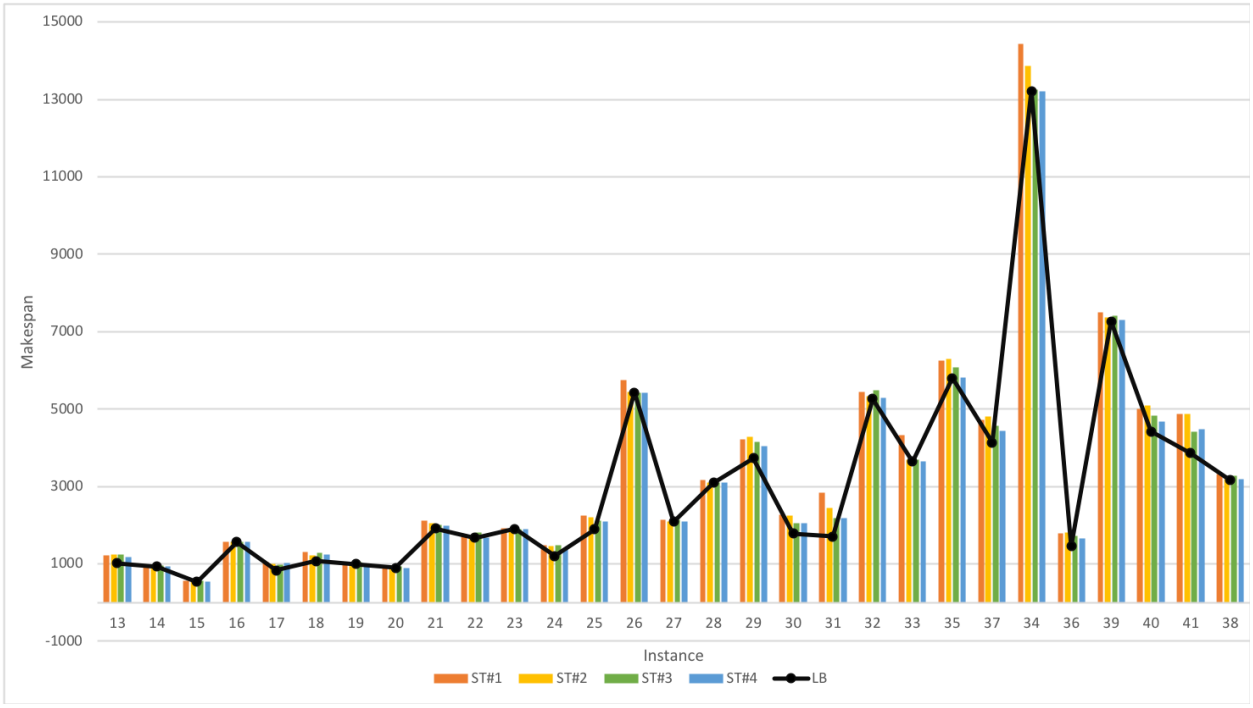

Figure 11.Best reported makespan of each solution technique vs LB for medium-large instances

In general, the performance measures evidence a good performance of all 4 techniques in terms of solution quality, finding a percentage difference against the LB lower than 5%, in 56 out of 116 (4 STs x 29 medium-large instances) executed instances (48.3%). The figure 11 illustrates the makespan behavior of all 4 techniques vs LB stablished. Overall, the solution techniques were able to maintain quality in obtained results as the instance sizes increased.

Considering results presented in tables 3 to 6 and image 11, for solution quality, ST4 overcome the other solution techniques given that for 66,7% of the all executed instances (10 small, 8 medium and 21 large instances) the results are close to the optimal solution (Dif% <=5%). Despite the results obtained for ST1 and ST2 for small instances, ST4 performed better for medium and large ranges with an average percentage difference from the LB of 6,33%.

Solution speed

This dimension measures the computational performance in terms of average running time and number of generations needed to find the best solution for small and medium-large instances.

Solution Technique	Small		Medium-large	
	Mean Tmin (s)	Mean gen	Mean Tmin (s)	Mean gen
# 1	0,12	13,7	233,60	4529,1
# 2	0,04	1,2	2570,53	4182,4
# 3	0,02	1,0	837,11	2868,8
# 4	0,38	14,6	237,29	3867,6

Table 7. Average execution time to reach the best solution for all size instances.

In terms of solution speed, small instances were solved in less than one second on average with every solution technique. On the other hand, ST2 performed poorly for medium and large size instances with a mean time to find the best solution of 43 minutes unlike ST1 that with a similar mean number of generations, found the best solution in less than five minutes. Additionally, in terms of number of generations, ST3 performed better than the other solution techniques with the least number of generations (on average) to find the best solution for medium and large ranges, and the fastest in terms of time and generations for small instances with only **ONE** generation needed to find the best

makespan. Even though ST3's quality solution measures were not remarkable, it finds good solutions iterating less times than the other techniques for all size instances.

Solution technique	Small instances	Medium- large instances
	T (s) / gen	T (s)/ gen
# 1	0,01	0,05
# 2	0,03	0,61
# 3	0,02	0,29
# 4	0,03	0,06

Table 8. Relation between mean execution time and mean number of generations

The relation between running time and number of generations are shown in table 8. ST1 shows to be the fastest technique to make an iteration of the GA on all instances with 0,01 and 0,05 seconds per generation. Furthermore, ST2 showed to be the slowest technique to perform a single generation. This measure highlights the importance of the coding of the solution techniques based on the logic presented in section 4.2. The combination of the scheduling heuristics, in a specific order, can increase the number of procedures to be executed. Although ST4 is also made out of a combination of scheduling heuristics, it shows to be better and faster than ST2 by cause of its particular logic that allows the elimination of several number of procedures.

Shop performance

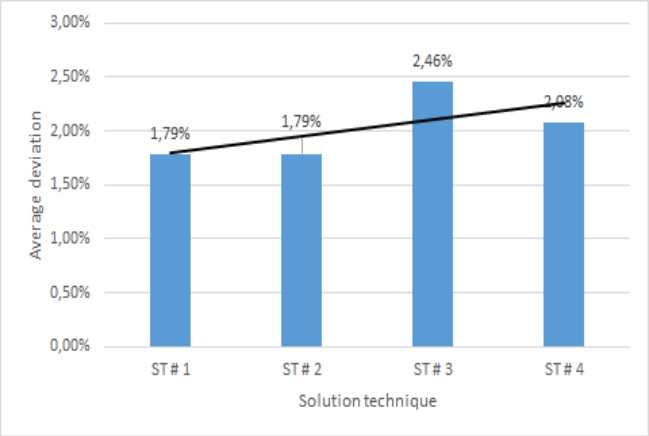

Figure 1312. Average percentage difference for small instances

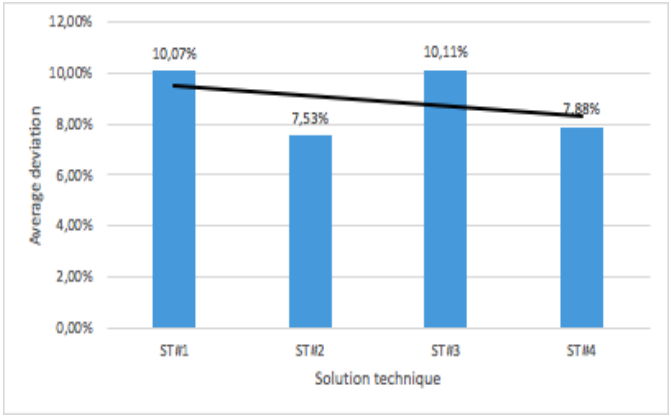

Figure 1213.Average percentage difference for medium instances

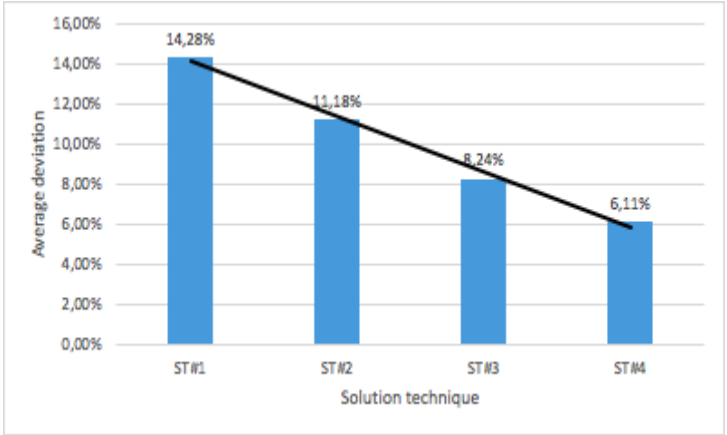

Figure 14. Average percentage difference for large instances

Images 12 to 14 show the behavior of solution techniques for small, medium and larger instance sizes. In general terms, the 4 presented solution techniques showed to be capable to solve and handle different instance sizes for the FOFS problem. At first, solution techniques perform well for small instances with the 90% being less than 5% away of the optimal result. For medium size instances, ST2 and ST4 outperforms the remaining techniques. Finally, for large size instances, ST4 stands out over the rest of solution techniques with 6,11% average difference against LB.

Solution Technique	mean Tmin(s) medium	mean Gen medium	T(s)/gen Medium	mean Tmin(s) large	mean Gen large	T(s)/gen Large
# 1	42,84	3054,5	0,01	306,28	5090,8	0,06
# 2	153,16	2829,5	0,05	3.491,43	4697,9	0,74
# 3	143,17	3391,8	0,04	1.101,47	2669,6	0,41
# 4	37,38	3279,0	0,01	313,45	4091,8	0,08

Table 9. Execution time and number of generations for medium and large instances

Table 9 resumes the speed measures for medium and large instances. The ST1 and ST4 show to be the more effective in comparison. However, ST4 outperforms ST1 in term of quality solution with an average percent difference of 2,19% and 8,17% for medium and large instances respectively. Despite ST2 shows the lower difference against LB for medium size, ST4 is closer with 0,35% percent difference with ST2, a negligible value considering that took 9.25 times more seconds per generation. The comparisons made over the three dimensions (solution quality, solution speed and shop performance) imply that ST4 overcomes the others solution technique overall.

Additional impact measure

Another type of instances were taken into consideration in order to compare the performance of the solution techniques with already presented results in literature. These additional instances were originally designed by Carlier, Heller and Reevs[41] for the flow shop scheduling problem (categorized as medium and large size instances for this project) and adapted to the open-flow shop scheduling problem solved by Mejia G et al [2] which is a similar type of mixed shop than the one stated in this project but without considering the flexibility of stages and the intervention of set up times.

The results obtained by the four solution techniques and a general comparison with the best results obtained by Mejia G et al [2] for the same instances are presented in attachment 7. It's necessary to clarify that the best results obtained by the authors are considered as upper bounds (UB) for this analysis since the proposed techniques could find better solutions. Therefore, the percentage difference is calculated considering the deviation between the best makespan obtained with each solution technique and the mentioned UB.

Solution Technique	Mean Dif %	Min Dif%	Max Dif%
# 1	-3,15%	-42,63%	8,00%
# 2	-4,52%	-43,48%	11,58%
# 3	-3,03%	-43,94%	14,67%
# 4	-7,50%	-45,80%	6,04%

Table 10. Average percentage difference between STs Vs UB

According to table 10, all techniques were able to outperform, on average, the results obtained by Mejia G et al[2] . Particularly, ST4 obtained the best results for this set of instances in comparison with the other solution techniques, achieving the minimum difference of 7,50% below the author's results and the lowest maximum difference of 6,04%. Additionally, all techniques have significant minimum differences less than -40%. These results imply that all solution techniques can adapt and solve these types of instances and perform properly.

Solution technique	Dif< 0%	Dif = 0%	Dif <= 5%	Dif > 5%
# 1	16	0	8	6
# 2	18	0	9	3
# 3	18	0	5	7
# 4	24	2	4	0

Table 11. Frequency distribution for STs Vs UB

Table 11 shows the number of instances that overcome the upper bound. It is important to notice that all solution techniques, for the majority of instances, surpass the results obtained by Mejia G et al [2] with 65% of the 120 (30 instances X 4 STs) execution instances being less or equal than the UB, indicating the competitiveness of the solution techniques when adapting to this type of problem. Additionally, reported results show that ST4 improves or equals 89.66% of author's results, reaffirming the good performance of ST4 overall. Moreover, it is the only technique where all instances reached a percentage difference of less than 5% against the UB. Image 15 shows a general comparison between the authors' best results and the solution techniques, showing once again how the proposed solution techniques were able to maintain quality in obtained results as the instance sizes increased.

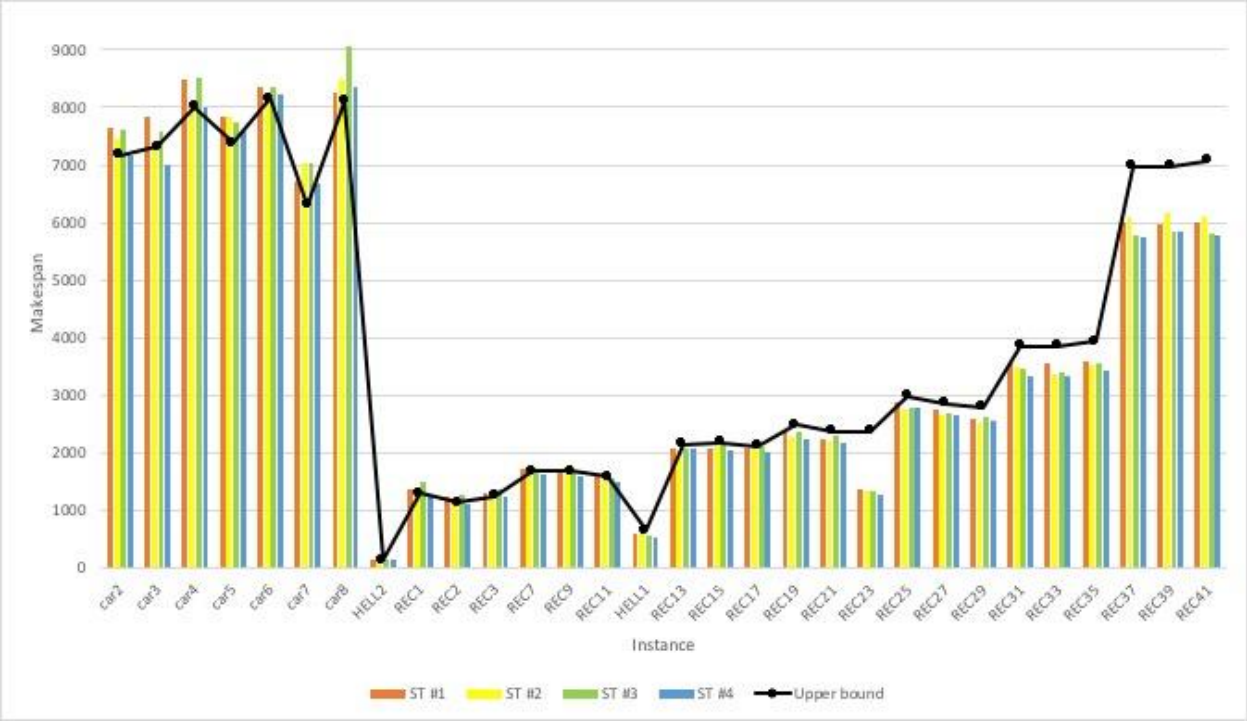

Figure 15. Best reported makespan of each solution technique Vs UB

The performance measures analyzed under the three-dimensional approach show that all solution technique achieves good results to solve the FOFS problem. However, the methodology to schedule jobs and evaluate the objective function ended up differentiating one of another. The design of ST2 and ST4 were focused on finding a dominance ambience when solving the problem, ST4 presents a methodology that combines Scheduling heuristic 2 for the Flexible open shop and scheduling heuristic 1 for the flexible flow shop. Results for ST4 confirms that this combination reached better solutions in reasonable computational time. This methodology target the FOS environment as the dominant ambience in the problem, giving more diversification when scheduling jobs. Meantime the FFS only creates a sequence depending on the release times of jobs at the end of the FOS part, maintaining this sequence for

the rest of FFS. Nonetheless, the design of ST2 was made considering that the interaction of both environment mainly depends on FFS, but results obtained by ST4 prove otherwise.

Considering the content presented in this section, the performance measures, the adaptability to different problems size, among others, To solve the Flexible Open-flow shop scheduling problem without approximating it based on an integral approach considering the interaction between these two shop environments the fourth solution technique is the one that fit the most. Taking into account the dominance of FOS over the FFS it is possible to explore more solutions and achieve better results i.e. a better makespan in reasonable computational time.

7. Conclusions and recommendations

- The resolution of the FOFS problem was carried out by the design of 4 solution techniques under the structure of the genetic algorithm, each technique with a different logic to approach the problem integrally. The results obtained evidence that it is possible to solve this problem based on an integral approach, reaching results equals or near to the optimal on 58,97% of all generated instances (small, medium and large).
- The mathematical model solves the FOFS problem considering all its characteristics. However, its application is limited due to the complexity of the problem, so, its use is restricted to instances composed of 5 jobs and / or 5 stations at most.
- Based on the computational performance, the genetic algorithm is an effective solution procedure to approach to the FOFS and to scheduling problems in general. This methodology is able to adapt to different instance sizes as well as their characteristics.
- The instances generation process was aimed to contextualize the problem posed under the classification on three type of sizes in terms of number of jobs and stages (small, medium and large). These instances tested the performance of the solution techniques in terms of quality, speed and efficiency, by varying the number of stages (open and flow), number of machines, number of jobs, processing and setup times.
- The lower bound is an element necessary to measure the impact of the solution techniques. Since the optimal values for medium and large sizes of instances are unknown, 3 different LBs with different logics were proposed to reliably measure the impact of the solution techniques.
- According to the performance measures, the fourth solution technique obtained the best results overall for this problem. Hence, this proves that the principle of focusing on the dominance of FOS over FFS may lead to obtain better results for this type of mixed shop.
- The comparison made with the results obtained by Mejia G et al [2] for the instances found in [41] proved that the solution techniques are capable of adapting to this manufacturing environment. Moreover, the proposed techniques evidence to be competitive given that at least the results of half of the executed instances were outperformed by each solution technique.

Recommendations

- The results obtained can be influenced by the programming language, specifically on computational time. In this project, VBA is used to code and execute the solution techniques. Hence, it's important to consider other programming languages, such as Python, C++ or Java that could lead to an efficient coding method.
- In order to explore deeper on the impact of genetic algorithm on the solving process, an experimental design could be made in order to find more suitable GA parameters such as mutation probability, number of generation, population size among others. This could lead to a parameterization for different size instances.
- Makespan is a widely known objective function, but in order to extend the knowledge of the problem some other objective functions can be explored such as, total tardiness, total idle time, among others, as well as other algorithms or solution techniques.

Glossary

- Scheduling: Is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives [3].
- Feasible solution: Is a solution for which *all* the constraints are *satisfied*[43].
- Optimal solution: Is a feasible solution that has the most favorable value of the objective function [43].
- Non-delay schedule: A feasible schedule is call non-delay if no machine is kept idle while an operation is waiting for processing [3].
- ST1: Solution technique 1. The same abbreviation is used for solution techniques 2, 3 and 4.

References

[1] B. Naderi, R. Ruiz, and M. Zandieh, "Algorithms for a realistic variant of flowshop scheduling," *Comput. Oper. Res.*, vol. 37, no. 2, pp. 236–246, 2010.

[2] G. Mejía, L. Bejarano, and C. Alberto, "Programación de una línea de producción bietápica mixta Open Shop y Flow Shop .," 2016.

[3] M. L. Pinedo, *Scheduling*. 2008.

[4] V. Nguyen and H. P. Bao, "An Efficient Solution to the Mixed Shop Scheduling Problem Using a Modified Genetic Algorithm," *Procedia - Procedia Comput. Sci.*, vol. 95, pp. 475–482, 2016.

[5] T. Masuda, H. Ishii, and T. Nishida, "The mixed shop scheduling problem," *Discret. Appl. Math.*, vol. 11, no. 2, pp. 175–186, Jun. 1985.

[6] N. V. Shakhlevich, Y. N. Sotskov, and F. Werner, "Complexity of mixed shop scheduling problems: a survey," *Eur. J. Oper. Res.*, vol. 120, no. 2, pp. 343–351, 2000.

[7] D. Bai, Z. H. Zhang, and Q. Zhang, "Flexible open shop scheduling problem to minimize makespan," *Comput. Oper. Res.*, vol. 67, pp. 207–215, 2016.

[8] J. Jungwattanakit, M. Reodecha, P. Chaovalitwongse, and F. Werner, "A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria," *Comput. Oper. Res.*, vol. 36, no. 2, pp. 358–378, 2009.

[9] E. Mokotoff, "Multi-objective Simulated Annealing for Permutation Flow Shop Problems," in *Computational Intelligence in Flow Shop and Job Shop Scheduling*, U. K. Chakraborty, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 101–150.

[10] B. Naderi, S. M. T. Fatemi Ghomi, M. Aminnayeri, and M. Zandieh, "Scheduling open shops with parallel machines to minimize maximun completion time," *J. Comput. Appl. Math.*, vol. 235, no. 5, pp. 1275–1287, 2011.

[11] G. J. Kyparisis and C. Koulamas, "Flexible flow shop scheduling with uniform parallel machines," *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 985–997, 2006.

[12] B. Lorena Mª Aranzazu and Gonzalo Enrique Delgadillo Mejía, "Análisis y caracterización para la optimización del Plan de Ventas y Operaciones (S & OP) en una empresa metalmecánica," pp. 2–15, 2015.

[13] S. Q. Liu, H. L. Ong, and K. M. Ng, "Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem," *Adv. Eng. Softw.*, vol. 36, no. 3, pp. 199–205, 2005.

[14] C. Koulamas and G. J. Kyparisis, "The three-machine proportionate open shop and mixed shop minimum makespan problems," *Eur. J. Oper. Res.*, vol. 243, no. 1, pp. 70–74, 2015.

[15] M. Gu, X. Lu, and J. Gu, "An asymptotically optimal algorithm for large-scale mixed job shop scheduling to

minimize the makespan," *J. Comb. Optim.*, vol. 33, no. 2, pp. 1–23, 2015.

[16] R. Ruiz, J. A. Vázquez-Rodríguez, and V. J. Antonio, "The hybrid flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 1–18, 2010.

[17] S. Verma and M. Dessouky, "Multistage Hybrid Flowshop Scheduling with Identical Jobs and Uniform Parallel Machines," *J. Sched.*, vol. 2, no. 3, pp. 135–150, Sep. 1999.

[18] C. J. Tapias-Isaza, A. A. Galeano-Ossa, and R. A. Hincapie Isaza, "Planeación de sistemas secundarios de distribución usando el algoritmo Branch and Bound," *Ing. y Cienc. - ing.cienc.*, vol. 7, no. 13, pp. 47–64, 2011.

[19] J. A. Hoogeveen, J. K. Lenstra, and B. Veltman, "Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard," *Eur. J. Oper. Res.*, vol. 89, no. 1, pp. 172–175, 1996.

[20] M. R. Singh and S. S. Mahapatra, "A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks," *Int. J. Adv. Manuf. Technol.*, vol. 62, no. 1–4, pp. 267–277, 2012.

[21] F. Glover, "Tabu Search and Adaptive Memory Programming --- Advances, Applications and Challenges," in *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*, R. S. Barr, R. V Helgason, and J. L. Kennington, Eds. Boston, MA: Springer US, 1997, pp. 1–75.

[22] R. Ruiz and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 781–800, 2006.

[23] M. E. Matta, "A genetic algorithm for the proportionate multiprocessor open shop," *Comput. Oper. Res.*, vol. 36, no. 9, pp. 2601–2618, 2009.

[24] W. L. Wang, H. Y. Wang, Y. W. Zhao, L. P. Zhang, and X. L. Xu, "Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm," *Comput. Oper. Res.*, vol. 40, no. 5, pp. 1196–1206, 2013.

[25] A. Rossi, S. Soldani, and M. Lanzetta, "Hybrid stage shop scheduling," *Expert Syst. Appl.*, vol. 42, no. 8, pp. 4105–4119, 2015.

[26] T. Ren, B. Liu, P. Zhao, H. Yuan, and H. Li, "Two-Stage Flow-Open Shop Scheduling Problem to Minimize Makespan," vol. 10362, pp. 527–535, 2017.

[27] V. Ravibabu, "A Novel Metaheuristics to Solve Mixed Shop Scheduling Problems," *Int. J. Found. Comput. Sci. Technol.*, vol. 3, no. 2, pp. 31–39, 2013.

[28] S. Q. Liu, H. L. Ong, and K. M. Ng, "A fast tabu search algorithm for the group shop scheduling problem," *Adv. Eng. Softw.*, vol. 36, no. 8, pp. 533–539, 2005.

[29] J. Błażewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *Eur. J. Oper. Res.*, vol. 93, no. 1, pp. 1–33, 1996.

[30] Z. Michalewicz, "Genetic algorithms+data structures=evolution programs," *Springer*, vol. Third edit, 1996.

[31] M.-C. Frunza, "Chapter 2D - Genetic Algorithms," in *Solving Modern Crime in Financial Markets*, M.-C. Frunza, Ed. Academic Press, 2016, pp. 151–161.

[32] M. G. Pose, "Introducción a los Algoritmos Geneticos," *Univ. la Coruña*, p. 16.

[33] A. C. Nearchou, "The effect of various operators on the genetic search for large scheduling problems," vol. 88, pp. 191–203, 2004.

[34] G. Zambrano, N. Ghazi, A. Bekrar, D. Trentesaux, and M. Tadjine, "A preliminary study on integrating operation flexibility within semi-heterarchical FMS control," no. October, 2015.

[35] A. Gadicherla, "A TSP-GA multi-objective algorithm for flow-shop scheduling," pp. 909–915, 2004.

[36] A. Costa and F. Antonio, “A hybrid genetic algorithm for minimizing makespan in a flow-shop sequence-dependent group scheduling problem,” *J. Intell. Manuf.*, pp. 1269–1283, 2017.

[37] T. Blickle and L. Thiele, “A Comparison of Selection Schemes used in Genetic Algorithms,” vol. 2, no. 11, 1995.

[38] K. Y. Orhan, Engin Cengiz, Kahraman Mustafa, *A Scatter Search Method for Multiobjective Fuzzy Permutation Flow Shop Scheduling Problem: A Real World Application*, Springer-V. 2009.

[39] R. Anand, Ellur Panneerselvam, “Active scheduling Generation heuristic to minimize the makespan for an Open Shop Scheduling Problem,” in *ICAIEA*, 2014.

[40] N. Nahavandi and E. A. Gangraj, “A New Lower Bound for Flexible Flow Shop Problem with Unrelated Parallel Machines,” pp. 2–7, 2008.

[41] D. C. Mattfeld and R. J. M. Vaessens, “OR-library,” 2004. [Online]. Available: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt.

[42] ISO, “International Organization for Standardization,” *Quantitative methods in process improvement -- Six Sigma -- Part 1: DMAIC methodology*, 2011. .

[43] Hillier and Lieberman, *Introduction to Operations Research*, vol. Seventh Ed, no. 34. McGrawHill, 2010.