Flexible Services and Manufacturing Journal
https://doi.org/10.1007/s10696-025-09599-5

A genetic algorithm for seafood processing with flexible flow shops and sequence-dependent setups

Tom Servranckx[1] · **Thibault Verbanck**[1] · **Jie Song**[2] · **Mario Vanhoucke**[1,3,4]

Accepted: 29 January 2025

Abstract
This paper studies a variant of the flexible Flow Shop Scheduling Problem as encountered at a large-scale Belgian seafood processing plant. The operations are conducted in two sequential stages as the seafood products are first filleted or prepared on specialised machines and then packaged through parallel machines. Since the packaging is product-specific, sequence-dependent setup times should be considered in the second stage. Improved scheduling of the operations would require fewer setups and thus efficiently planning the operations on the machines at the packaging station will be an important objective of this research. Furthermore, since the end product quality is crucial in the food industry and this is mainly determined by the speed of processing, the makespan will be minimised in this study. However, we further contribute to the existing literature by investigating several objectives that were relevant to the company's management. The scheduling problem is solved using a single- and multi-pass algorithms that can easily be implemented in the seafood processing plant. Furthermore, a genetic algorithm with a focus on various diversity measures and problem-specific crossover and mutation operators is developed. Although the genetic algorithm is more difficult to implement, it allowed us to solve real world cases with over 100 orders daily within a reasonable computational time, resulting in an improved solution quality.

Keywords Flexible flow shop · Genetic algorithm · Sequence-dependent setup times · Multiple objectives · Seafood processing

1 Introduction

An important objective of any business is to produce goods and services that satisfy the quality requirements of their customers and deliver these products and services on time. In order to satisfy these external goals of the company, the internal processes need to be optimised such as minimising the throughput time (or makespan) of the production process, minimising the setup times needed to ensure that

Extended author information available on the last page of the article

Published online: 08 March 2025

machines function properly and minimising the idle time of these machines. Therefore, a vast majority of research has been devoted to the development of novel and improved methods for (optimally) planning the production activities subject to limited resource (e.g. machine) capacities and their impact on the parallel execution of tasks. Given that there exists a large variety of production processes in different industries, many configurations of production scheduling problems have been presented in the literature. The research presented in this study is based on a case study at a large Belgian seafood producer where different types of seafood are first processed and then packaged. Both operations generate a large part of the revenue for the company, but are highly vulnerable to increased variability.

This case study is highly interesting and challenging from an academic perspective due to the combination of several attributes that are frequently analysed independently, but are now integrated in a single scheduling problem. The production process is characterised by a pre-defined set of (alternative) parallel machines in some or all stages with distinct processing times for each machine. The first workstation with a focus on seafood processing consists of three machines with unique functions. The second workstation with a focus on packaging consists of seven machines that have similar characteristics (*parallel*), but use different moulds, gas compositions and packaging sizes, and thus require *sequence-dependent setups*. Improved scheduling of these operations would require fewer setups and thus efficiently planning these operations on the machines at the packaging station will be an important objective of this research. Additionally, the company sets forward *different objectives* for the production planning. In seafood processing, the quality (i.e. freshness) of the end product is crucial and this is mainly determined by the speed of processing. Hence, the makespan will be minimised in this study, but other objectives are also investigated such as minimising the idle time and (total or non-idle) setup times in order to provide managerial insights.

The case study is identified as a flexible Flow Shop Scheduling Problem (FSSP) in which a group of tasks need to be processed by a predefined set of (alternative) parallel machines in different sequential stages (by one machine per stage) with distinct processing times for each machine given that all tasks follow the same order. We will solve the flexible FSSP using a range of approaches varying from a single-pass heuristic to a multi-pass heuristic and a metaheuristic framework. This allows us to compare the current situation in the plant with an improved and best-case solution without any guarantee of optimality.

The contributions of this study to the existing (food) production scheduling literature are threefold. First, we investigate a variant of the flexible FSSP with a combination of serial and parallel machines and sequence-dependent setup times (SDST). Most research on flexible FSSP is focused on testing algorithms in a theoretical way using artificial (benchmark) data (Chaudhry and Khan 2016). However, it is also necessary to validate algorithms through real-world cases (Calleja and Pastor 2014), which is one of the key contributions of this study. Our problem is illustrated in a seafood processing environment, but can be applied to other production settings inside or outside the food production industry as well. Second, we investigate the impact of different types of objectives on the resulting production schedules and operational decisions. Finally, we solve our problem using a range of solution

methods, which can be installed in the production plant with varying levels of financial and technological investments (i.e. computational requirements). We aim to bridge the gap between the theory of production scheduling and its applications in a seafood processing plant in order to aid future academics in solving similar problems. Therefore, we show the importance of mapping real world business cases on theoretical problem formulations (and extensions), emphasise a pragmatic approach for selecting the objective of the optimisation problem and test different solution approaches with a varying level of (computational) complexity.

An overview of the paper is given along the following lines. In Sect. 2, we present an overview of the different problems classifications and applications that exist in the production scheduling literature. Section 3 introduces the seafood processing operations and the specifications of the two workstations in our case study as well as highlights two main problem characteristics in order to position our problem within the existing literature. Also, the problem formulation is discussed with a focus on the four objectives that are considered in this study. In Sect. 4, the three solution approaches are discussed with a focus on the different strategies for the operators of the genetic algorithm. Section 5 reports the results of the parameter tuning of the GA, the analysis of different objective functions, the comparison with benchmarks. Finally, the overall conclusions, managerial insights and future research directions are given in Sect. 6.

2 Literature overview

We will discuss the class of production scheduling problems that is tackled in this study, i.e. the flexible FSSP, in Sect. 2.1. For an overview of different metaheuristic methods used in the literature to solve the flexible FSSP, we refer the reader to Li and Gao (2016). In Sect. 2.2, we highlight the research studies that use a GA framework to solve the FSSP, stressing their novelties and main differences. Finally, we discuss the gap in the literature that presents the foundation of this research study in Sect. 2.3. An overview table summarising all the acronyms used throughout this manuscript is presented in Appendix 1.

2.1 Production scheduling

The well-known Job Shop Scheduling Problem (JSSP) consists of a set of $N = \{1, 2, ..., n\}$ jobs to be processed on a finite set of $M = \{1, 2, ..., m\}$ machines (Kato et al 2018). Each job must be processed on every machine and consists of a series of m operations that need to be scheduled in a predetermined job-specific order. Additionally, each job can be processed on only one machine at a time and each machine can process only one job at a time. Furthermore, all operations are assumed non-preemptable and setup times are typically included in the processing times and are independent of the sequencing decisions (which means that the setups are ignored). In this study, a special case of the JSSP will be considered, namely the FSSP. This problem consists of a set of N jobs to be processed on a set of M

machines with all jobs being processed sequentially on multiple machines. The different operation and machine sequences between jobs mainly differentiates the JSSP from the FSSP since the FSSP comes down to the JSSP with additional features of routing flexibility (Saqlain et al 2023). Note that the existing literature mainly focuses on a reduced version of the general FSSP, the permutation FSSP with the added assumption that jobs must be processed in the same sequence by each of the machines. In this study, we deal with the *flexible FSSP* that considers flow shops with multiple parallel machines in a multi-stage environment. It is a complex scheduling problem that is encountered in many real world settings. Jayamohan and Rajendran (2000) compare two approaches to schedule flexible flow shops. One approach uses different dispatching rules at the different stages of the flow shop, while the other approach uses the same dispatching rule at all the stages of the flow shop. In our study, we will follow the former approach and use different rules at each stage as motivated later in this study (cf. Sect. 3.1).

We next discuss research studies that address the FSSP with various types of setup times since this is a major characteristic of the case study at hand. Singh and Mahapatra (2014) present a particle swarm optimisation (PSO) algorithm to solve the FSSP with the inclusion of a mutation operator, which is commonly used in GAs to avoid premature convergence. Defersha (2011) studies the FSSP with attached and detached setup times, machine release dates and technological time lag constraints, and develops a MILP formulation as well as a GA. Amirteimoori and Kia (2023) also propose a MILP model together with a Parallel Hybrid PSO-GA (PPSOGA) to address the simultaneous scheduling of jobs and automated guided vehicles in a flexible job shop system. Rossi and Dini (2007) consider transportation times that overlap with processing times using an ant colony method in order to model a practical manufacturing environment, while Rossi et al (2014) include transportation times, varying release dates for each job and overlapping transportation and processing times as well. The authors solve the assignment and sequencing subproblem of the FSSP with related parallel machines and setup times in a single step using an advanced ant colony algorithm. Zandieh et al (2009) consider the group scheduling problem consisting of two levels, namely the scheduling of groups as well as the jobs within each group, using two new metaheuristics based on a simulated annealing (SA) and a GA. Their proposed procedure is based on the response surface methodology and considers SDST when minimising the makespan. Li and Gao (2016) focus on the flexible FSSP without job preemption or transportation times and sequence-independent setup times that completely overlap with the processing times. Abdelmaguid (2015) examines structural properties of special cases of the FSSP with SDST in which relative small setup times in relation to the processing times are put forward. Ruiz et al (2005) solve the flexible FSSP with SDST considering machine eligibility and unrelated parallel machines at every stage. Azzouz et al (2016) solve the FSSP with SDST with a bi-criteria objective function using a GA.

Finally, several authors focus on shop scheduling problems with multiple conflicting objectives. Kurdi (2017) summarises three approaches for solving multi-objective problems in this setting: a weighting approach, a lexicographical order-based approach and a Pareto-based approach. Mokhtari et al (2011) consider a

deterministic FSSP with resource-dependent processing time with the objective of finding a job sequencing that minimises the makespan as well as the total cost of resources. The authors develop a hybrid discrete differential evolution algorithm to solve this problem. Gao et al (2014) present a pareto-based grouping discrete harmony search algorithm, while Zhang et al (2017) propose a dynamic game optimisation model to minimise the makespan, the total workload of machines and the total energy consumption of the production proces. Gong et al (2018) formulate a scheduling model that considers machine flexibility and worker flexibility simultaneously, and propose a memetic algorithm to minimise the makespan and the total workload of all machines. Lu et al (2017) formulate a mathematical model with the objectives of minimising both the makespan and the total additional resource consumption. In order to solve this problem, they propose a new multi-objective discrete virus optimisation algorithm (MODVOA) with a three-part representation for each virus, an improved method for generating the initial population, various operators to update each virus and a problem-specific exploitation mechanism. Nouiri et al (2017) consider unexpected disruptions such as machine breakdowns and present a two-stage PSO to minimise the makespan and stability. Recently, the energy cost-aware FSSP is solved by addressing the minimisation of both the makespan and energy costs using a memetic non-dominated sorting GA (NSGA-II) (Burmeister et al 2023).

2.2 Solution approaches

Initially, research studies focussed on simple and generic production scheduling problems with limited real-world applications (Hopp and Spearman 2011), but this focus rapidly shifted to a broader spectrum of scheduling problem with a wider variety of solution methodologies such as exact methods, metaheuristics and matheuristics (i.e. hybrid exact and metaheuristic procedures). Exact methods have a high computational burden on large instances, while metaheuristics efficiently and effectively explore the search space by balancing two aspects during the search process: diversification and intensification (Blum and Roli 2003). Diversification explores all possible neighbourhoods in the solution space, while intensification explores a narrow solution space in more detail. For an overview of the metaheuristic algorithms that have been proposed in the literature to solve shop scheduling problems, we refer the reader to Li and Gao (2016) and a non-exhaustive overview is provided in Table 1. This study aims to approximately solve a complex real-world production scheduling problem using a carefully selected metaheuristic framework rather than to set up a comprehensive comparative study of different metaheuristic alternatives. Given the specific characteristics of the case study, the goal is to present a real-world application of several optimisation methods ranging from simple heuristics to more complex variants of a metaheuristic. Based on our literature review, we opted to develop a GA in this research, but alternatively a PSO algorithm could have been developed to solve our complex scheduling problem given its its good performance when balancing different objectives (decreasing the makespan and the setup time in our case study). However, the disadvantage of PSO is twofold. It is known for its slow convergence, which could be problematic given the large size of the company's

Table 1 Overview of the literature on metaheuristics for production scheduling problems GA: Genetic Algorithms ES: Evolutionary Strategies EP: Evolutionary Programming PSO: Particle Swan Optimisation ACO: Ant Colony Optimisation SS: Scatter Search NN: Neural Networks GRASP: Greedy Randomized Adaptive Search Procedure VNX: Variable Neighbourhood Search ILS: Iterative Local Search SA: Simulated Annealing TS: Tabu Search TA: Threshold Accepting

References	Evolutionary Computation			PSO	ACO	SS	NN	Explorative Local Search			SA	TS	TA
	GA	ES	EP					GRASP	VNX	ILS			
Holland (1992)	X												
Rechenberg (1973)		X											
Fogel (1962)			X										
Eberhart and Kennedy (1995)				X									
Dorigo (1992)					X								
Glover et al (2000)						X							
Foo and Takefuji (1988)							X						
Feo and Resende (1995)								X					
Hansen and Mladenović (2001)									X				
Stützle (2006)										X			
Kirkpatrick et al (1983)											X		
Glover and Greenberg (1989)												X	
Dueck and Scheuer (1990)													X

operations, and its smaller degree of customisation since there are only two different parameters that can be set by the company, which reduces the flexibility of the company's operations to guide the search process. Therefore, we opted to develop a GA framework over a PSO framework. In the remainder of this section, we will provide a non-exhaustive overview of studies that are used for the development of our GA procedure.

Reeves (1995a) introduces a good-performing Nawaz-Enscore-Ham (NEH) initialisation step for the FSSP as well as a fitness rank distribution and uniform distribution-based selection of parent solutions and applies a one-point crossover to these parents. Furthermore, the generated offsprings do not – by definition – replace their parents, but rather they replace solutions with a below average fitness value. This approach was further extended by Reeves and Yamada (1998) as they implement a hybrid GA with a multi-step crossover fusion operator (MSXF) and performed a biased local search on one parent using the other parent as a reference point. Murata et al (1996) propose an elitist strategy for the FSSP together with a two-point crossover and shift mutation. Ruiz et al (2005) combine the operation sequence- and job-based representation with four new crossover operators in order to identify and create problem-specific building blocks. They use a shift mutation operator and an adapted version of the NEH heuristic to generate the initial population. Li and Gao (2016) develop a hybrid GA and Tabu Search metaheuristic, and implement an elitist and tournament selection scheme. They also randomly select between a precedence-operation crossover (POX) and a job-based crossover (JBX) as well as a swap and neighbourhood mutation, both with an equal probability of selection.

2.3 Gap in the literature

From the above literature review, we can conclude that flexibility is an inherent feature of many workshop environments and should thus be considered when making scheduling decisions in order to improve the production efficiency. However, studies on the flexible FSSP including SDST are more challenging, yet scarcely investigated in the literature, in comparison to flexible FSSP research without considering SDST. Therefore, this study introduces an implementation of a GA to solve the flexible FSSP with SDST that is encountered in a real world manufacturing plant, but can easily be adapted to scheduling environments in other production settings.

Based on the discussion in Sect. 2.2, we observe that different GA variants have been studied for the FSSP. However, to the best of our knowledge, most studies do not deal with flexible machines and/or SDST that are central to our problem formulation. Therefore, we propose a GA framework based on best-known operators from the literature in order to implement simple, yet effective scheduling strategies for the proposed case study, considering multiple alternative objectives. The performance of the GA will be compared in our computational experiments with a single- and multi-pass approach, both well-known benchmarks in the production literature (Geyik and Elibal 2017; Balin 2011; Li et al 2013; Neufeld et al 2016).

3 Problem description

In this section, we first discuss the data collection process and present the details of the two key workstations in the case study (Sect. 3.1). Afterwards, we position our real world problem setting within the existing classifications in the production literature (see Sect. 2). Section 3.3 details the different objectives of the production scheduling problem that are investigated in this study. Finally, a mathematical formulation of the flexible FSSP with SDST is presented in Sect. 3.4.

3.1 Case study

This research has been conducted in close collaboration with a large seafood processing plant in Belgium. The first step was to analyse the context and specifications of the plant with regard to the production process, corporate culture and its clients. Therefore, several interviews were setup with the management of the plant and a thorough inquiry was needed to collect all necessary data. When we were familiar with the required policies, the production process and the demand distribution, a lot of decisions were made in close collaboration with management such as the type of solution approach and the relevant process parameters. Once all these decisions were made, we generated a high-quality production schedule and validated its robustness. Finally, we sat together with management to ensure that our production schedules could be implemented on a day-to-day basis.

Based on discussions with management, we found that it was a struggle to come up with a robust planning for the main workstations at the plant due to the high demand volatility and the influence of extensive setups in the production process. The plant receives both fresh seafood (such as salmon) directly from its supplier, which need to be filleted, as well as semi-processed seafood products that need to be cut in smaller, processable pieces. There are over 180 seafood products that are processed at the plant, which can all be categorised as either filleting of fresh seafood (with a different category for salmon filleting due to its unique properties) or semi-finished products:

Filleting This is the most value-adding activity since the fish are filleted manually by skilled workers. Several different operations per product need to be executed sequentially. We impose a fixed workforce since these skilled workers have a fixed contract at the plant.

Salmon All filleting and partitioning operations with regard to the salmon products are performed separately in order to keep up with high order volumes, avoid cross-contamination with other fresh fish and meet the high quality requirements for salmon products. These activities require lower-skilled workers and thus the company can appeal to temporary workers on a daily basis (i.e. a variable workforce).

Semi-finished products There are some products that do not need to be filleted or partitioned before they can go to the next workstation since they are delivered

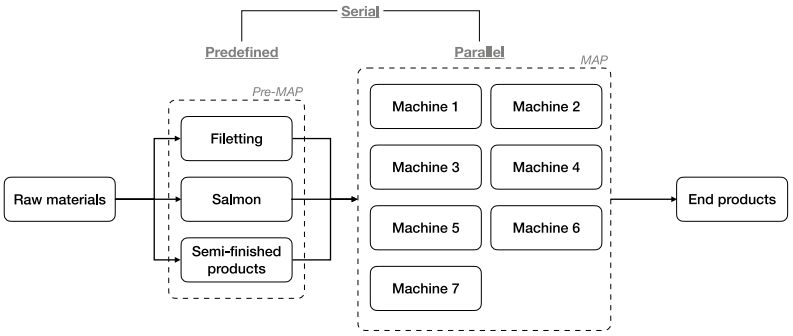

Fig. 1 Shopfloor layout of the case study

at the plant as semi-finished products. Before these products can be packaged as finished products, workers need to sort the big batches into processable sizes. Again, the company considers a variable workforce since there is no high-skilled labour required.

Afterwards, all seafood products (both fresh and semi-processed) need to be packaged for shipment to wholesalers and stores. At the plant, there are seven packaging machines and each machine can be used for packaging all types of products (fresh seafood and semi-finished products). All products are packaged using a Modified Atmosphere Packaging (MAP) approach, which is a technology used for extending the shelf life of fresh food products, as it substitutes the atmospheric air inside a package with a gaseous atmosphere. In the remainder of this work, we will therefore refer to the first workstation as the *pre-MAP* workstation and refer to the second workstation as the *MAP* workstation. These two consecutive workstations, processing and packaging, need to be perfectly coordinated in order to optimise the throughput of the plant. In case that a machines needs to package a different type of product, a setup is required that depends on the type of mould, gas or packaging of the previous and current type of product (i.e. sequence dependency).

As can be seen in Fig. 1, there is a total of 10 different machines in the shopfloor: three in the pre-MAP station (processing) and seven in the MAP station (packaging). At the pre-MAP station, every order must be processed on a *predefined* line[1]. This workstation cannot be considered a parallel machine scheduling problem since the machines are pre-defined rather than interchangeable and, hence, it should rather be approached as three independent single machine scheduling problems. As a result, the decision-making in the pre-MAP workstation is rather straightforward since the arriving products are assigned to their corresponding lines (pre-defined assignment) and processed according to a first-in-first-out principle. In contrast, every order at the MAP workstation requires one operation on an undefined line and thus the seven machines at this workstation work in *parallel*. As a result, two interrelated decisions need to be made at

[1] Note that we sometimes use the terminology *lines* in the workstations when we refer to *machines*.

Table 2 Data summary

Entity	Instances	Additional information
Orders	32,132	average = 100 products per day
Activities	180	average = Two activities per product
Machines	10	Seven identical machines in the MAP station
Configurations	32	Gas, mould & packaging

the MAP workstation: the operations must be sequenced and must be assigned to the different lines. This complex problem with sequence-dependent setup times in combination with the different objectives (cf. Sect. 3.3) is solved using a GA. Although the production process is divided into two separate stations, we decide to solve the problem in an integrated way in order to provide the company's management with a complete plan for the entire production process. From an academic perspective, the most challenging and interesting scheduling problem is that of the MAP station with parallel machines and sequence-dependent set-up times, while the problem in the pre-MAP station is rather straightforward. However, by extending our problem formulation (Sect. 3.2) and methodologies (Sect. 4) to incorporate both stations, we aim to present an off the shelf solution for the company including all specific characteristics derived from their production process. However, our solution procedures can, without loss of generality, be applied to a scheduling problem with solely a MAP station as well.

We discuss the benchmark data of the case study below and present a summary in Table 2:

Order The company provided us with a sample of all orders for a period of 316 days. We received a total of 32,132 orders and an average of just over 100 orders per day, with an average order quantity of 59.90 kg per order.

Activity There is only data available on the activities that are related to the MAP station and the lines related to the pre-MAP station (i.e. filleting, salmon and semi-finished products), this explicitly defines the scope of our work.

Product There are 180 products in the database, which corresponds with the vast majority of the products of the seafood company and all products relevant for the MAP planning are considered.

Configuration After an analysis of all available data, 32 configurations could be composed. These configurations link activities to a certain machine or workstation. It determines the predefined machine assignment in the pre-MAP workstation and contains information about which mould, gas and packaging is required in the MAP station.

Machine As discussed before, there are 10 different machines in our database: three in the pre-MAP station and seven in the MAP station.

Mould/Gas/Packaging If a change in mould, gas or packaging is required, a setup time must be taken into account. Changing a mould requires on average twenty

minutes, changing the gas composition on average five minutes and changing the packaging only takes two minutes on average.

3.2 Problem classification

Based on the problem formulation presented in Sect. 2.1, our problem should be classified as a flexible FSSP with SDST as discussed along the following lines.

The shopfloor layout consists of multiple machines in two sequential workstations where the machines in the MAP workstation are able to execute equivalent operations, which means that at least a subproblem can be classified as *parallel* machine scheduling. In parallel machine scheduling, each job can be processed on any of the machines and processing times are independent of the machine (i.e. identical machines). Therefore, our system can be seen as a mix of serial and parallel machine scheduling, also known as a hybrid shopfloor with K serial workstations and one or more identical parallel machines at each workstation (Zobolas et al 2008). It is important to note that our case study can be classified as a **flexible FSSP** because there are two or more parallel machines for *at least* one workstation (Pinedo 2008). According to Ruiz and Vazquez-Rodriguez (2010), the following properties present in our case study define a flexible FSSP:

- There are n jobs denoted by j that require a processing time $p_{j,l}$ in workstation $l = \{1, ..., m\}$.
- There are at least two workstations ($l \geq 2$).
- There are parallel machines in at least one workstation.
- The jobs should be processed in the same flow in all workstations.

The flexible FSSP is a well-known strongly NP-hard problem, even when it is restricted to two workstations (Gupta et al 2002). By considering SDST, as discussed in the following paragraph, the scheduling problem becomes even more complex (Pinedo 2008).

If a change in mould, gas composition or packaging is required in the MAP machines due to a change in the type of order (filleting, salmon or semi-finished products), a setup time must be taken into account. These setups cannot be executed simultaneously, so multiple setups should be cumulated. Since the setup times depend on the sequence of the orders on the (parallel) machines in the MAP workstation, we refer to them as **sequence-dependent setup times**.

A brief overview of the **assumptions** in our research is presented along the following lines. First, there is no data available on the perishability and supply of the products, and thus all raw materials are assumed to be available at any point in time. Second, transportation times between workstations typically depend on the product, product quantity and route, but they are assumed negligible in this study. Finally, a variable workforce could be set at different lines and machines in the plant. Nevertheless, a fixed

workforce at the pre-MAP and a fixed number of machines at the MAP workstation are considered in this research.

3.3 Objectives

After discussions with management, it became apparent that several time-related objectives were highly relevant to their business problem. Four variants of such objectives are considered in this study and briefly discussed below:

Makespan: When all operations are scheduled, the makespan of the schedule can be calculated. Therefore, the differences between the completion time of the last operation and the starting time of the first operation of the same job are summed over all jobs, which includes any setups done in the MAP workstation.

Setup time: The objective function consists of the project makespan (which includes setups) extended with the total setup time of all machines in the MAP workstation. Hence, this objective additionally penalises the number of setups. Since setups are labour-intensive and present a potential risk factor in the production process, these alternative production schedules are also requested by management.

Non-idle setup time: The objective function consists of the project makespan (which includes setups) extended with the setup times of all machines in the MAP workstation that cannot be executed during idle times. This objective is similar to the previous objective, however, only the inefficient setups are additionally penalised.

Idle time: The objective function consists of the project makespan (which includes setups) extended with the total setup time and the total idle time of all machines in the MAP workstation. In this case, the aim is to put more emphasis on the efficient use of the machines in the production schedules.

3.4 Mathematical model

In this section, we present a mathematical model for the flexible FSSP with SDST in general. An overview of the indices, parameters and dependent variables can be found in Table 3. We have derived the model from the work of Hasani et al (2022). The binary decision variable $x_{i,j,k,l}$ is equal to 1 when job j is processed directly after job i on machine k in workstation l, or 0 otherwise.

$$\text{Min } C_{max} \tag{1}$$

subject to

$$\sum_{i \in N} \sum_{k \in K} x_{i,j,k,0} = 1 \quad \forall j \in N; i \neq j \tag{2}$$

Table 3 Overview notation of mathematical model

Symbol	Explanation
Index	
i, j, h	Indices for jobs, with $i, j, h \in N$
l	Index for workstation, with $l \in L$
k	Index for machine, with $k \in K$
Parameter	
$p_{k,j,l}$	Processing time of job j on machine k in workstation l
$s_{i,j,k,l}$	Sequence-dependent setup time between jobs i and j on machine k in workstation l
M	Large value (big M)
Dependent variable	
$C_{j,l}$	Completion time of job j in workstation l

$$\sum_{j \in N} x_{i,j,k,l} \leq 1 \quad \forall i \in N; i \neq j; \forall k \in K; \forall l \in L \tag{3}$$

$$\sum_{i \in N} x_{i,h,k,l} - \sum_{j \in N} x_{h,j,k,l} = 0 \quad \forall h \in N; h \neq i; h \neq j; \forall k \in K; \forall l \in L \tag{4}$$

$$C_{j,0} \geq x_{i,h,k,0} \times (p_{k,j,0} + s_{i,j,k,0}) \quad \forall k \in K; \forall i, j \in N; i \neq j \tag{5}$$

$$C_{j,l} + \sum_{k \in K} x_{i,h,k,l} \times (p_{k,j,l} + s_{i,j,k,l}) + M \times (\sum_{k \in K} x_{i,h,k,l} - 1) \leq C_{j,l} \quad \forall l = 1, ..., |L|; \forall i, j \in N; i \neq j \tag{6}$$

$$C_{j,l-1} + x_{i,h,k,l} \times (p_{k,j,l} + s_{i,j,k,l}) \leq C_{j,l} \quad \forall l = 1, ..., |L|; \forall k \in K; \forall i, j \in N; i \neq j \tag{7}$$

$$C_{j,l-1} + M \times (1 - x_{i,j,k,l}) \geq C_{i,l-1} \quad \forall l = 1, ..., |L|; \forall k \in K; \forall i, j \in N; i \neq j \tag{8}$$

$$C_{j,l} \leq C_{max} W \quad \forall l \in L; \forall j \in N \tag{9}$$

$$x_{i,j,k,l} \in \{0, 1\} \forall i, j \in N; \forall k \in K; \forall l \in L \tag{10}$$

The objective of the standard problem formulation in this study is to minimise the makespan C_{max} (Eq. (1)). Equations (2) guarantee that each job j in the first workstation (i.e. $l = 0$) must be assigned to at least one machine k. Equations (3) indicate that every job i has at most one succeeding job j in each workstation l, while Eqs. (4) indicate that each job h should be processed precisely once at each workstation

l. Equations (5) calculate the completion time of jobs in the first workstation ($l = 0$). Equations (6) indicate that the completion times $C_{i,l}$ and $C_{j,l}$ are calculated consecutively if job i is processed immediately before job j on the same machine. Equations (7) guarantee that the completion time of each job in each workstation l is equal or greater than its completion time in the previous workstation plus the corresponding setup and processing time in workstation l. Equations (8) ensure that the processing order of the jobs on the machines is the same for the subsequent workstations. Equations (9) calculate the maximum completion time of jobs, which is equal to the makespan C_{max}. Finally, the decision variables $x_{i,j,k,l}$are defined by Eqs. (10).

4 Methodology

We test three solution approaches to solve the scheduling problem presented in Sect. 3.3. First, a single-pass algorithm from the literature (Sect. 4.1) is coded in order to resemble the current scheduling approach at the seafood processing plant. Second, a more complex multi-pass algorithm is tested that builds upon the knowledge obtained from the single-pass algorithm (Sect. 4.2) and illustrates a relatively simple improvement of the current scheduling actions at the plant. Finally, we develop a GA to further improve the solutions and come up with the best found schedules for the specific setting at the plant (Sect. 4.3), which requires more computational resources and acts as an indicator for the (estimated) long-term improvement potential of the scheduling processes at the plant.

4.1 Single-pass approach

A well-known single-pass algorithm that is widely used in many real-world situations is applied to solve our flexible FSSP with more than two machine centers. It combines the Longest Processing Time (LPT) with a search-and-prune approach to find a (near)optimal makespan (Hong et al 1998). Given a set of n independent tasks (T_1 to T_n), each with arbitrary execution time (t_1 to t_n), and a set of m parallel processors or machines (P_1 to P_m), the LPT approach first assigns the task with the longest execution time (among those not yet assigned) to an idle machine whenever it becomes available. For tied cases, an arbitrary tie-breaking rule is used. Subsequently, the search-and-prune approach is used to deal with job sequencing. We will opt to apply this technique to mimic the current scheduling process in our seafood processing plant.

4.2 Multi-pass approach

In contrast to single-pass approaches, multi-pass approaches need more computation time, but they are likely to provide a better solution. We opted for the Logendran and Nudtasomboon (LN) approach (Logendran and Nudtasomboon 1991) since it is up to today used in novel metaheuristic frameworks for solving subparts of more

complex scheduling problems (Huang et al 2006). The steps associated with the LN heuristic are based on the conversion of an m-machine group scheduling problem into an equivalent 2-machine problem (Petrov 1966).

First, the processing time matrix is divided in two equal time matrices T (m is even) and $T^{\star}$ (m is odd) and the sum and the difference of the sum of the two matrices are also given for m being odd and even. When searching for a good job sequence, the following two heuristic rules are applied and the best sequence is retained:

Rule 1: For those jobs in which $\sum T^{\star} - \sum T \leq 0$, the job sequence is constructed based on the ascending order of $\sum T$ values, while the descending order of $\sum T^{\star}$ values is used otherwise.

Rule 2: The job sequence is constructed using the descending order based on the numerical value of $\sum T^{\star} - \sum T$.

In two special cases, the multi-pass heuristic is reduced to a single-pass heuristic. First, when all values of $\sum T^{\star} - \sum T$ are exclusively positive or negative since *rule 1* will then provide only a single job sequence. Second, when a number of identical values are present in $\sum T^{\star} - \sum T$ since *rule 2* will then also provide only a single job sequence.

4.3 Genetic algorithm

A genetic algorithm is an evolutionary metaheuristic approach frequently used for optimisation problems in general and scheduling problems in particular (see Sect. 2.2). It searches for improved solutions over several generations by combining information about existing solutions in the population combining two strategies. On the one hand, *intensification* focuses on combining the information of good solutions, called parents, to create even better solutions, called offspring. On the other hand, *diversification* avoids that the search process gets stuck in a local optima by altering some information in the genes in order to investigate different parts of the search space. The search process of a GA is guided by a combination of operators such as initialisation, parent selection, crossover and mutation.

As can be seen from Fig. 2, three initial solution generation methods, three parent selection techniques, five crossover operators and two types of mutation will be tested in our GA. Some of these operators are derived from the existing literature and their performance is thus validated by other researchers in the past, while we also include two unique features that distinguish our GA from the traditional GA framework. First, we initiate the search process partly with solutions obtained for other instances (different from the instance solved in the current search process). Second, we test and compare three diversity measures to ensure sufficient diversification and avoid over-intensification in the search process.

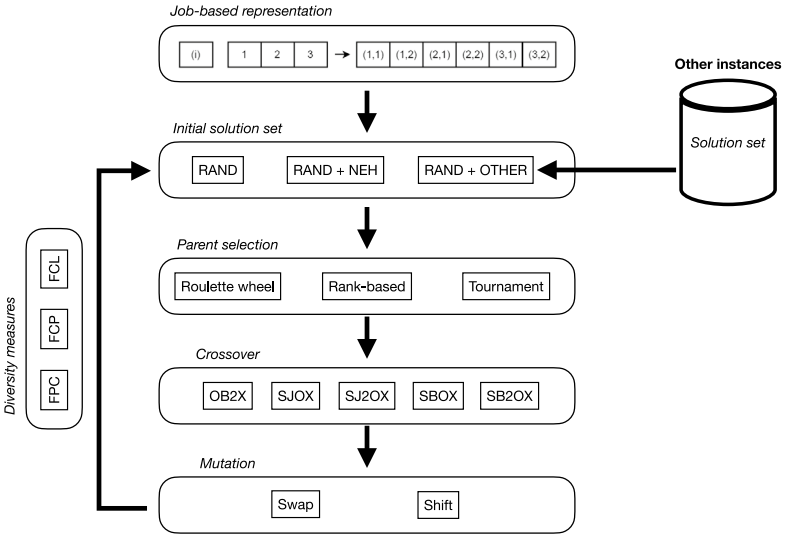

Fig. 2 Overview of the different variants for the GA operators

In this study, we opted for a genetic algorithm (GA) since Azzouz et al (2016) have shown that it outperforms several other existing methods. The difference between the proposed GA and other GAs in the literature is highlighted in Table 4.

4.3.1 Representation

The job-based representation of Ruiz and Maroto (2006) is chosen since their problem setting is highly similar to our case study. Using this representation, we need to implement a heuristic rule for the machine assignment that considers the SDST. Given a predefined job sequence, the following decoding is then applied on all operations of the current job. First, the orders should be assigned to their predefined machine in the pre-MAP workstation and, subsequently, the orders should be assigned to the parallel machine in the MAP workstation that completes the operation the earliest taking into account sequencing constraints between operations of the same job. If there are no valid gaps in the schedule, the operation is scheduled at the slot with the earliest possible completion time rather than the earliest start due to the SDST.

Below, we show an illustration of the encoding and decoding process. The illustrative example consists of 10 jobs to be processed through two main workstations (Pre-MAP and MAP) with three machines in the pre-MAP workstation (i.e. filleting, salmon and semi-finished products) and three machines in the MAP workstation, which is a simplification of the actual shopfloor layout. The following notation is used $I = \{i_1, i_2, ..., i_{10}\}$, $M_1 = \{m_{11}, m_{12}, m_{13}\}$ and $M_2 = \{m_{21}, m_{22}, m_{23}\}$. As explained before, the assignment to the first workstation is predefined, i.e. jobs 2, 6 and 8 should be filleted and jobs 1, 5 and 10 contain salmon products, while jobs 3, 4, 7 and 9 are semi-finished products. In

Table 4 Overview of research studies using GA for the flexible FSSP

Reference	Initialisation	Selection	Crossover	Mutation	Elite
Reeves (1995a)	NEH	Fitness rank Uniform distribution	C1	Shift	No
Murata et al. 1996	Random	adjusted selection probability	two-point	shift	Yes
Reeves and Yamada (1998)	NEH	Fitness rank Uniform distribution	MSXF	MSMF or MSXF	No
Ponnambalam and Aravindan (2001)	Random	–	GPX	Shift	No
Zheng and Wang (2003)	NEH	SA	multi-crossover	SA	No
Our research	Random, NEH, Other solutions	RWS,Ranked, Tournament	OB2X, SJ(2)OX,SB(2)OX	Swap, Shift	No

Table 5 Job processing time on each machine (input) and machine allocation (output) given a user-specified job sequence

	M_1				M_2			
	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$	A	$m_{2,1}$	$m_{2,2}$	$m_{2,3}$	A
i_1	–	5	–	2	**7**	8	7	1
i_2	2	–	–	1	8	9	**6**	3
i_3	–	–	3	3	8	**7**	9	2
i_4	–	–	4	3	6	6	**8**	3
i_5	–	4	–	2	**5**	7	9	1
i_6	6	–	–	1	7	5	**6**	3
i_7	–	–	3	3	8	**9**	7	2
i_8	5	-	–	1	**5**	6	8	1
i_9	–	–	5	3	9	**7**	8	2
i_{10}	–	6	–	2	**6**	7	7	1

The job processing times on the selected machines for the second workstation are given in bold

Table 5, we indicate the job processing times on the different machines. Given that the jobs in the pre-MAP workstation are predefined, a processing time is only provided for those machine and a dash indicates that the other machines cannot be used. In the MAP workstation, however, each machine corresponds with different processing times for each job. According to Ruiz and Maroto (2006), a single job sequence suffices to encode a solution. Given the following job sequence $I_S = \{i_2, i_5, i_1, i_7, i_4, i_8, i_9, i_{10}, i_3, i_6\}$, also shown in the top of Fig. 3, the machine allocation A in both the pre-MAP and MAP workstation can be decoded. In Table 5, the job processing times on the selected machine is highlighted in bold and the corresponding Gantt chart is shown in Fig. 3. Notice that no job is

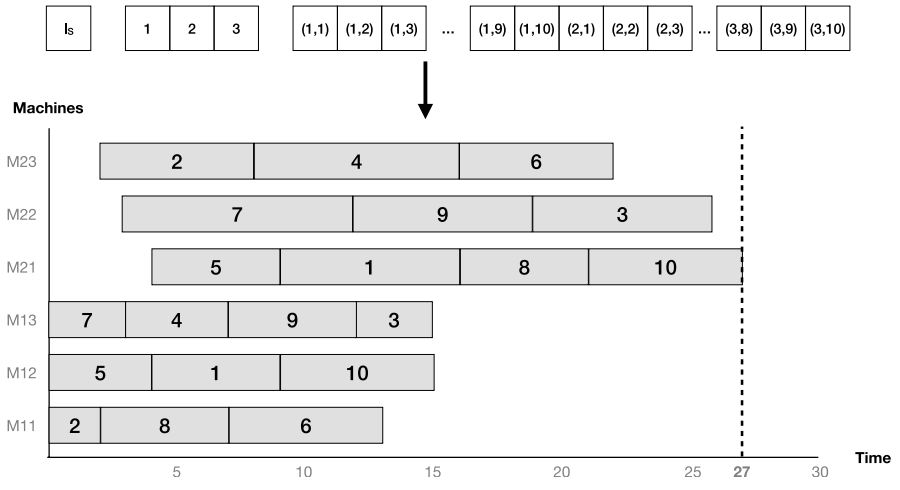

Fig. 3 Gantt chart of solution in Table 5 for job sequence I_S

processed on an allocated machine before another job that is positioned later in the job sequence I_S, both in the pre-MAP and MAP workstation, which indicates that the user-specified job sequence is satisfied.

In the first workstation, the solution has a relatively long completion time to make full use of idle machines, while it also schedules short jobs (e.g. job 2 on machine 1 and job 7 on machine 3) first to trigger an early start of the second workstation. In this example, the best solution has a makespan equal to 27, which also corresponds with the optimal solution for this illustrative example.

4.3.2 Initial solutions

The creation of an initial population with good solutions can improve the performance of the GA significantly. Therefore, three different approaches are proposed and tested in this study.

In the *first approach*, the initial population will be randomly initialised. In the *second approach*, part of the individuals will be initialised randomly, while another part will be initialised with solutions for other instances. In the *third approach*, a part of the individuals will be randomly initialised, while another part will be initialised using the NEH algorithm (Nawaz et al 1983). In the literature, it is shown that the NEH algorithm performs best in a permutation flowshop sequencing context (Reeves 1995b). The method was further adjusted by Ruiz et al (2005) in a permutation FSSP with SDST. In this work, the evaluation of the objective value is slightly adjusted to incorporate the SDST for a flexible FSSP. The NEH algorithm can be summarised as follows.

1. Create an original NEH instance according to Reeves (1995b).
   (a) Calculate the total processing time for all jobs and sort the jobs in descending order of processing time.
   (b) The first two jobs $i = \{1, 2\}$ are scheduled in the first and second place and vice versa. These two possible schedules are then evaluated and the best schedule in terms of the project makespan is retained.
   (c) For every job $i = \{3, ..., n\}$, the best schedule is searched by placing job i in all possible positions in the sequence of jobs that are already scheduled.
2. Create other NEH instances with the two initial jobs being randomly selected and swapping them with the two jobs in the original NEH instance.

We illustrate the above approach to generate NEH instances based on a problem with four jobs and five machines obtained from Nawaz et al (1983). In step (a), the total processing times T_i are calculated for each job i and the results are presented in the last column of Table 6. Subsequently, we order the jobs according to descending processing time as follows: 1 – 3 – 2 – 4. In step (b), we pick the first two jobs (i.e. jobs 1 and 3) and find the optimal partial sequence for these

Table 6 Processing time of each job on each machine (top) and makespan for partial sequences (bottom)

Jobs	Machines					T_i
	1	2	3	4	5	
1	5	9	8	10	1	33
2	9	3	10	1	8	31
3	9	4	5	8	6	32
4	4	8	8	7	2	29
1	5/5	9/14	8/22	10/32	1/33	
3	9/14	4/18	5/27	8/40	6/**46**	
3	9/9	4/13	5/18	8/26	6/32	
1	5/14	9/23	8/31	10/41	1/**42**	
3	9/9	4/13	5/18	8/26	6/32	
1	5/14	9/23	8/31	10/41	1/42	
2	9/23	3/26	10/41	1/42	8/**50**	
3	9/9	4/13	5/18	8/26	6/32	
2	9/18	3/31	10/31	1/32	8/40	
1	5/23	9/32	8/40	10/50	1/**51**	
2	9/9	3/12	10/22	1/23	8/31	
3	9/18	4/22	5/27	8/35	6/41	
1	5/23	9/32	8/40	10/50	1/**51**	
3	9/9	4/13	5/18	8/26	6/32	
1	5/14	9/23	8/31	10/41	1/42	
2	9/23	3/26	10/41	1/42	8/50	
4	4/27	8/35	8/49	7/56	2/**58**	
3	9/9	4/13	5/18	8/26	6/32	
1	5/14	9/23	8/31	10/41	1/42	
4	4/18	8/31	8/39	7/48	2/50	
2	9/27	3/34	10/49	1/50	8/**58**	
3	9/9	4/13	5/18	8/26	6/32	
4	4/13	8/21	8/29	7/36	2/38	
1	5/18	9/30	8/38	10/48	1/49	
2	9/27	3/33	10/48	1/49	8/**57**	
4	4/4	8/12	8/20	7/27	2/29	
3	9/13	4/17	5/25	8/35	6/41	
1	5/18	9/27	8/35	10/45	1/46	
2	9/27	3/30	10/45	1/46	8/**54**	

The maximum completion time of all jobs on all machines in each partial sequence is given in bold since this corresponds with the makespan

two jobs. In Table 6, we show the sequences 1 – 3 and 3 – 1 and it can be seen that the sequence 3 – 1 is the best (makespan equal to 42). In step (c), we add job 2 in all possible sequences and we observe that sequence 3 – 1 – 2 is the

best (makespan equal to 50). Finally, we add job 4 in all possible sequences and we observe that sequence 4 – 3 – 1 – 2 is the best (makespan equal to 54). It turns out that this sequence is optimal, which can be proven by solving $n! = 24$ sequences. However, the NEH approach found this sequence by only completing nine sequences (i.e. four complete and five partial sequences).

4.3.3 Parent selection

In our GA, two different selection operators are analysed. The first operator selects the individuals, or solutions, for generating offsprings (*parent selection*) and the second operator selects the individuals that determine the next generation given a selection pool of candidates (*survivor selection*). For both selection operators, three types are tested: roulette-wheel, rank-based and tournament selection:

Roulette-wheel selection is a mechanism to apply a fitness proportionate selection that finds potentially useful solutions for recombination.

Rank-based selection orders the individuals according to their fitness value from 1 to n with a higher fitness value corresponding with a higher rank. This is particularly useful when the individuals in the population have very close fitness values. In traditional roulette-wheel selection, this would lead to all individuals having an almost equal chance of being selected, which in turn results in a huge loss of selection pressure.

Tournament selection randomly selects T individuals from the complete population with T being the tournament size. Out of these T individuals, the best one is selected for crossover (parent selection) or goes through to the next generation (survivor selection). A tournament is executed for a number of times equal to the population size in order to ensure that the number of individuals in the population remains constant. A low T results in a lower selection pressure and a decreased runtime, but the chance of losing qualitative genetic material increases.

4.3.4 Crossover

Five different crossover operators are tested in this study (OB2X, SJOX, SJ2OX, SBOX and SB2OX) derived from Ruiz et al (2005) and Ruiz and Maroto (2006). We focus on these operators because the existing literature indicates that these crossovers have potentially a good performance for our case study. Some crossovers keep the building blocks of the schedule fixed by retaining one or more jobs in the same position as both parents. We expect that this could be crucial when solving the flexible FSSP with SDST given the inclusion of setup times and the importance of machine assignment (Ruiz et al 2005; Ruiz and Maroto 2006).

- **OB2X**: This operator creates two random crossover points in the parent and copies the building blocks between them from the first parent to the first off-

spring. Then, starting from the second crossover point in the second parent, it copies the remaining unused building blocks from the second parent to the first child. Subsequently, this process is repeated for the second child with reversed parent roles.
- **SJOX**: Both parents are examined on a position-to-position basis and the building blocks are directly copied to their offspring. Then, a random crossover point is selected and the part before this point is directly copied to the offspring, while the missing elements of each offspring are copied in the relative order of the other parent.
- **SBOX**: This operator is similar to SJOX, but the building blocks are considered to have a minimum length of two jobs.

Further note that SJ2OX and SB2OX are highly similar to SJOX and SBOX, respectively. The only difference is that two crossover points are used and the middle part is directly copied from the parent to the offspring.

4.3.5 Mutation

Two mutation operators are implemented in this study to avoid getting stuck in a local optima. The first operator is the well-known *swap mutation* operator. The second operator is a *shift mutation* operator in which a randomly selected job is relocated to another random location, while all the jobs between these two positions are shifted one position to the left (or right).

4.3.6 Diversity measures

In this study, we opt for a steady state GA since the offsprings replace the worst individuals in the population with a higher probability. In order to avoid a huge loss in diversity, however, three *diversity measures* are implemented:

- **Forbid parent if child is selected (FPC)** This measure ensures that a parent is not allowed in the new generation, when its child is already selected.
- **Forbid child if parent is selected (FCP)** This measure ensures that a child is not allowed in the new generation, when its parent is already selected.
- **Forbid clones (FCL)** This measure ensures that individuals with the same sequencing of jobs (i.e. clones) are not allowed in the new generation.

Based on these three diversity measures, the following four scenarios will be considered: no diversity measures are activated (NON), only FCL is activated, both FPC and FCP are activated and all diversity measures are activated (ALL).

4.4 Procedure implementation

As discussed at the start of this section, the single- and multi-pass algorithms are programmed to mimic the current scheduling approach at the processing plant and

a relatively simple improvement, respectively. However, these methods are merely used as theoretical benchmarks and the goal of this study was not to implement them in practice. We agreed to develop a conceptual design including data preprocessing, a graphical user interface, a database design and a functional programming design with model solver (i.e. the GA framework of Section 4.3).

The overall implementation design is presented in Appendix 2 (see Fig. 14). The customer orders are the main drivers of the production planning for a particular day and all activities are initialised as work orders for every customer. Subsequently, the optimiser assigns these workorders to machines and time slots in some cases accessing a supportive entity where previous schedules were stored (as discussed in Section 4.3.2). A relational (MySQL) database is used to store all long-term data, which can be easily linked to the company's existing servers. This was considered a main advantage of our implementation design. The production schedule could then be shown in a Graphical User Interface (GUI) together with some performance indicators (as discussed in Section 3.3) to monitor and validate the proposed daily production schedule.

Finally, a general overview of the implemented program can also be found in Appendix 2 (see Fig. 15. The initialiser is called once to extract information about the orders, machines and configurations from the database, and generate the work orders. The workorders allow to store information about the start and finish times of tasks/orders as well as machine assignments, and they are initialised using information from the database and the already initialised orders. Typically, the GA will be run multiple times to allow a comparison in terms of the different objectives (cf. Section 5.3) or a specific variant can be selected by management given the objective with the highest priority for that day.

5 Computational results

In Sect. 5.1, we discuss the best set of parameter values for the different GA operators obtained after parameter tuning. In Sect. 5.2, the performance of the best GA configuration is compared with the single-pass (see Sect. 4.1) and multi-pass heuristic (see Sect. 4.2) in case that the objective is to minimise the project makespan since the single- and multi-pass algorithm were only developed in the literature for the basic FSSP with this makespan objective. In Sect. 5.3, subsequently, three different variants of the problem in terms of the objective function are analysed. Finally, the performance of our GA framework will be compared with a random approach and the NEH approach for all objectives (Sect. 5.4). A set of historic orders for a period of 316 days were used to design, set and tune the GA. The experiments are conducted on a 2.20GHz Intel Core i7-8750 H laptop with 16GB RAM using the Windows 11 Professional operating system. All algorithms were implemented in the Visual Studio Code environment (version 1.81) using the C++ programming language. The computational times of the different algorithms tested in this study are reported in Table 7. Since the run times did not significantly differ between the different variants of the GA, we

Table 7 Running times of all algorithms

Method	CPU time (seconds)
Random	752
Single-pass LPT	801
Multi-pass LN-Rule 1	1,341
Multi-pass LN-Rule 2	1,380
GA	1,479
Gurobi	1,800

Table 8 Results of parameter tuning for various components

Component	Settings				
Population size	**150**	**250**	**350**		
Avg. obj. value	120.14	103.12	97.07		
Best. obj. value	107.12	97.22	93.70		
Crossover operator	**OB2X**	**SBOX**	**SB2OX**	**SJOX**	**SJ2OX**
Avg. obj. value	101.52	102.17	102.66	100.87	101.53
Best. obj. value	96.52	99.98	97.23	98.57	97.42
Crossover probability	**0.2**	**0.5**	**0.8**		
Avg. obj. value	121.69	102.63	96.84		
Best. obj. value	109.22	97.46	94.01		
Tournament size	**0.01**	**0.02**	**0.05**		
Avg. obj. values	112.95	101.93	96.38		
Best. obj. value	112.95	96.94	93.66		

The different parameter settings are given in bold

report the average run time of the GA. We notice that these run times are in line with run times reported in the literature for problems of a similar complexity. Given that these algorithms will be used by project managers for making tactical (rather than operational) decisions, the observed computational run times are not considered problematic since the maximum run time is less than 25 min.

5.1 Parameter tuning

The main goal of the GA is to find a close-to-optimal and robust solution by selecting and recombining good solutions using good operators over different generations. In order to avoid getting stuck in local optima, diversification and intensification need to be balanced in the search process. On the one hand, the GA might be reduced to a random search if there is too much diversification (inefficient). On the other hand, the GA might get stuck in a local optimum if there is too much intensification (ineffective). In our procedure, diversification is ensured by means of a large population size and the mutation operator, while intensification is ensured by means of two selection mechanisms that retain good (or better) solutions. In the remainder

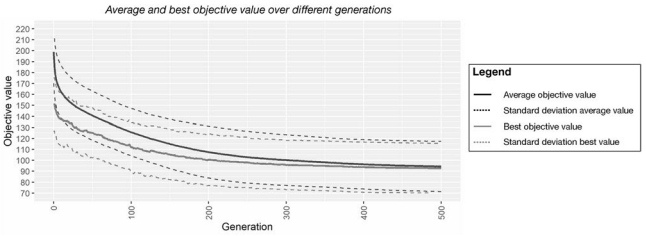

Fig. 4 Impact of the number of generations

of this section, different operator parameters are tested using a Taguchi design-of-experiments (DoE) ($L25 = 5^5$) of which some key results are shown in Table 8.[2]

5.1.1 Generations

When the average objective value in the current generation lies sufficiently close to the objective value of the best performing individual over all generations, convergence is obtained (i.e. the convergence gap is sufficiently low). This means that the intensification effects are leveled out and there is not enough diversity left within the search process to get improved individuals in reasonable computational time. In preliminary experiments, we test the average number of generations after which a sufficiently low convergence gap is obtained. In the upper part of Fig. 4, the average objective value of the population is plotted in black and the best objective value of the current generation in dark grey, both with their standard deviation presented as a dotted line (below and above the average values). We observe that convergence is typically reached after 500 generations, yet most of the diversity is worked out after 300 generations. Therefore, a GA with 500 generations will be used for our final experiments.

5.1.2 Population size

The population size is a key factor to ensure diversity within a GA with larger populations resulting in a larger diversity. Other research studies on the FSSP usually use a population size between 50 and 100. Since our problem is more complex than the basic FSSP due to a number of extensions, however, it requires more diversity and thus a larger population. Based on our parameter tuning, we observe that a population size of 350 results in a slightly better objective value than a population size of 250, but twice as much computational time would be needed to reach convergence. Therefore, we decide that a population size of 250 presents a perfect balance in the trade-off between solution quality and computational time.

[2] One exception has been allowed in the DoE as the mutation probability was fixed to zero in order to observe a clear impact of the crossover operators.

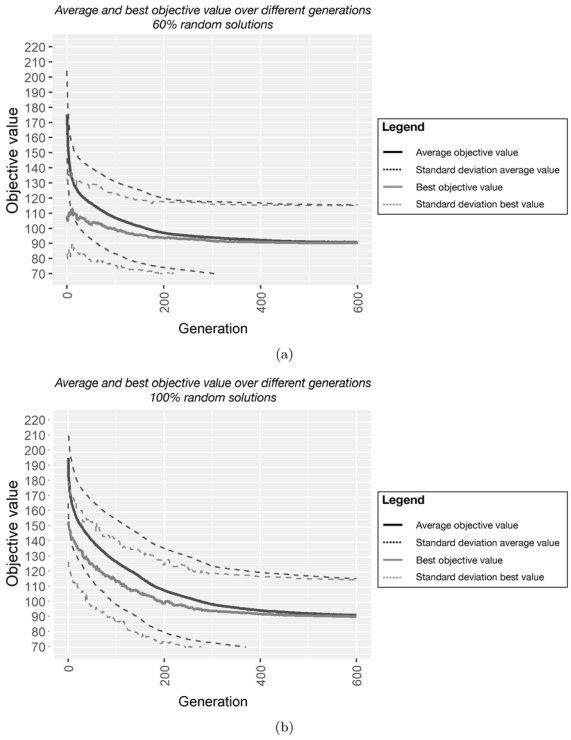

Fig. 5 Impact of the initialisation of the population

5.1.3 Initial population

As explained in Sect. 4.3, our population is partially initialised with random solutions and solutions obtained from other instances in the training set. More specifically, we have tested four settings of initialisation with, respectively, 0%, 10%, 25% and 40% of the initial solution being obtained from other instances, while the remaining solutions were generated randomly. The results in the middle part of Fig. 5 show the two extreme situations where 60% (a) and 100% (b) random solutions were used in the initialisation phase. However, the best setting is observed in an intermediate situation where 90% of the population is randomly initialised and 10% of the population is initialised with solutions of other instances. This setting positively influences the convergence time, but leaves room for exploring new areas in the solution space.

5.1.4 Selection mechanism and crossover operator/probability

The parameter tuning shows that tournament selection outperforms roulette-wheel and rank-based selection as the survivor selection method. Furthermore, we observe that different levels of selection pressure (0.01;0.02;0.05) have no significant effect on the solution quality and, hence, the lowest selection pressure (0.01) is preferred since it requires the lowest computational time. With respect to the crossover

operators, the SJOX and SBOX seem to have a lower performance compared to the other three operators. This suggests that using a one-point crossover that keeps building blocks in solutions over multiple generations hinders the search process in a large solution space. Furthermore, one would expect SJ2OX and SB2OX to outperform the benchmark OB2X operator, however, the computational results show otherwise. This again validates our assumption that keeping building blocks fixed in solutions over multiple generations is a too restrictive strategy in our problem context. Hence, our GA will be run with the OB2X crossover operator at a 50% crossover probability. Using a lower probability results in a lower performance of the GA, while a higher probability results in more or less the same solution quality, but it requires more computational time.

5.1.5 Mutation operator/probability

The shift mutation outperforms the swap mutation operator by a large margin given a similar computational time. Note that compared to the swap operator, the shift operator only adjusts the individuals in a very incremental way, which keeps some information of the individuals over different generations. This emphasises the importance of fixing some building blocks in the solutions, which is an observation that contradicts the insights obtained when tuning the crossover parameters. In Fig. 6, four levels of the mutation probability are tested: 0%, 25%, 45% and 50%. Since a probability of 0% results in significantly worse results than the other probabilities, the added value of the mutation operator is clearly shown. In general, the mutation rates in this study are set higher than in other works dealing with similar problems. The reason is twofold. On the one hand, the selection mechanism focuses mainly on intensification and thus this should be balanced with sufficient diversification. On the other hand, the specific context of our case study with setup times in a flexible environment implies a large solution space that requires a sufficient degree of diversification.

5.1.6 Diversity measures

The impact of the five scenarios with respect to the diversity measures discussed in Sect. 4.3 is shown in Fig. 7. We observe a faster convergence when the diversity measures are not activated (NON), but a better solution quality when all measures are activated (ALL). FCL slows down the convergence since a good chromosome can only be selected for crossover and mutation once. Since FCP and FPC exclude even more similar individuals than FCL, the results of FCP and FPC are not better than the results of FCL.

5.2 Benchmarking

Using the best combination of parameter settings for the GA discussed in Sect. 5.1, we compare this metaheuristic framework with the single- and multi-pass heuristic as well as a random benchmark. The random solution can be considered a worst-case

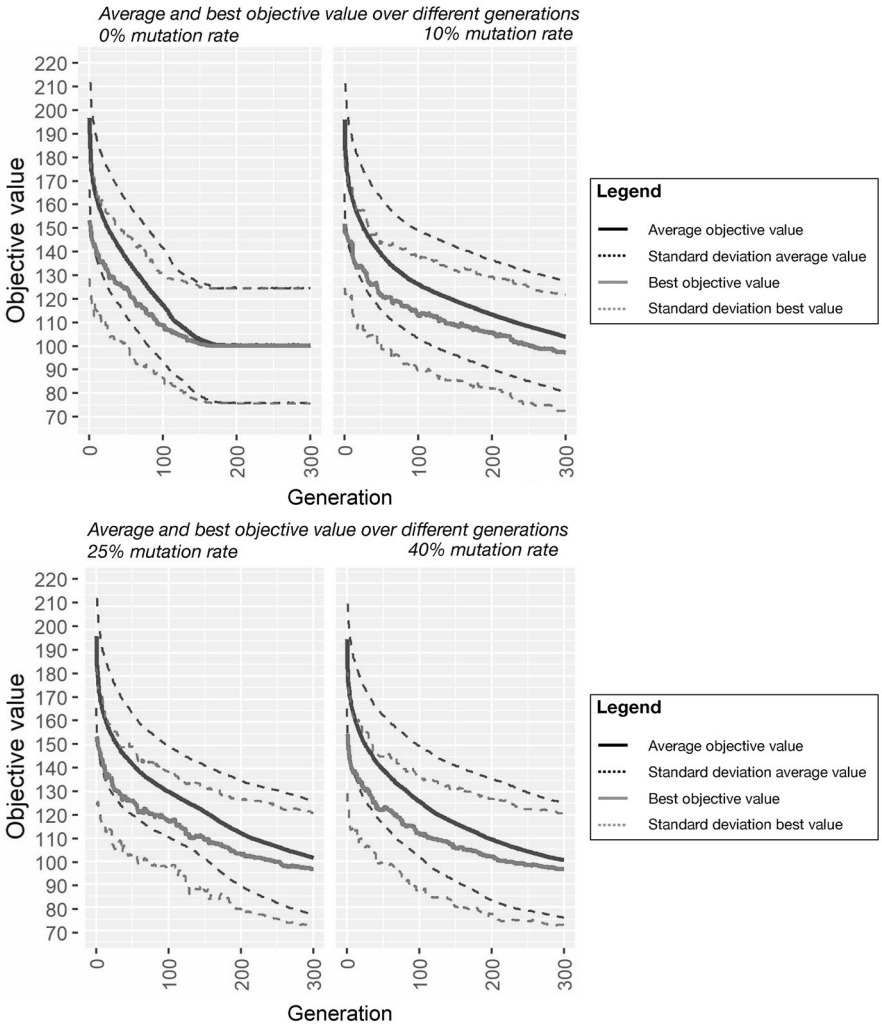

Fig. 6 Impact of the mutation rate

solution for the actual production schedule, while the single-pass LPT approach is used as a proxy for the fast and easy scheduling approach currently used at the plant. By comparing the random and single-pass heuristic, we can show the added value of a pragmatic generation of production schedules. In contrast, the multi-pass heuristic corresponds with an improved solution or production schedule, which is the next step in the company's scheduling efforts. Fortunately, the increased complexity compared to the single-pass heuristic is rather limited since the multi-pass LN method only requires the construction of multiple schedules in contrast to a single schedule. Finally, the GA presents another ambitious next step in terms of methodological complexity for the company.

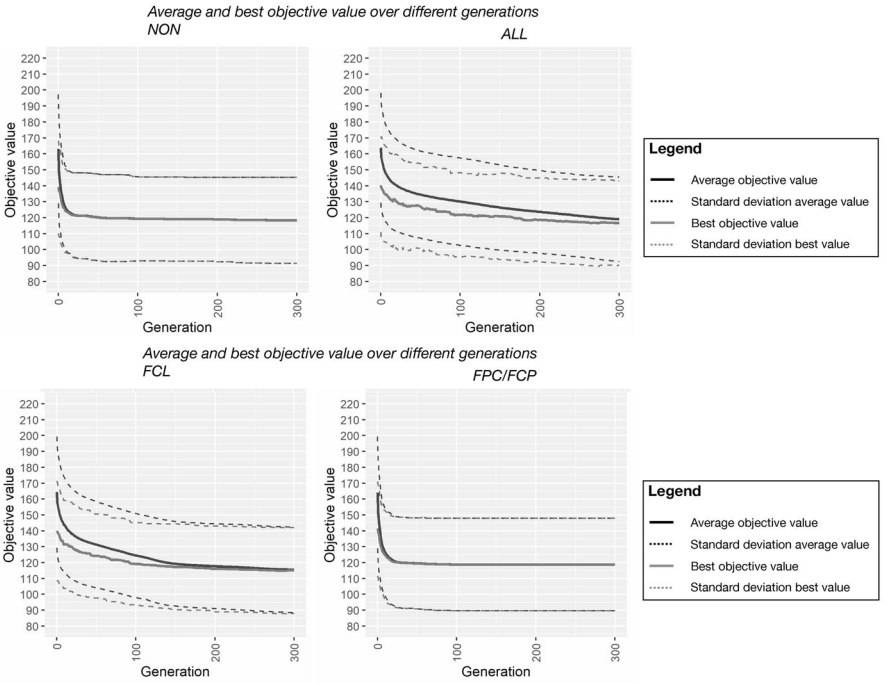

Fig. 7 Impact of the diversity measures

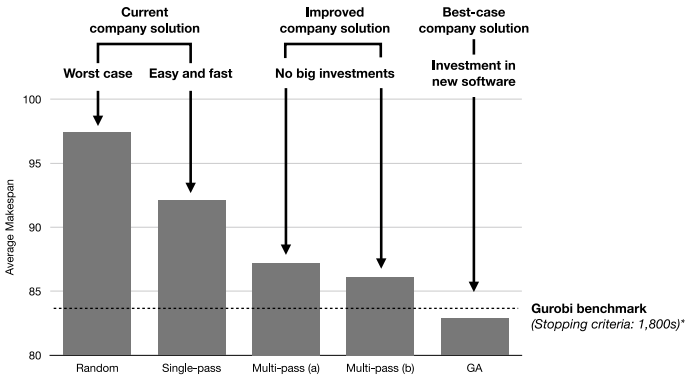

Fig. 8 Comparison of performance of GA with single- and multi-pass algorithm on the historic company data

The results are summarised in Fig. 8[3]. We observe that the makespan is significantly reduced when the LN heuristic is used compared to the random and single-pass

[3] Remark that we need to investigate two variants of the LN heuristic since this multi-pass heuristic is reduced to a single-pass heuristic in two special cases (see Sect. 4.2), which have a negative impact on

heuristic, which clearly presents an incentive to the company for investing in computational resources on the short- to mid-term. Moreover, we notice that the performance can be further improved by implementing a GA framework, which can be interpreted in two ways. On the one hand, the results further incentivise the company to invest in a state-of-the-art scheduling and decision-support tool on the long-term. Although the GA is rather complex compared to the single- and multi-pass heuristics and hence would require a significant investment of the company in better (algorithms) software, its overall complexity is kept limited in order to facilitate an easy implementation in the plant. On the other hand, the results of the multi-pass heuristic (attainable on the short- to mid-term) are relatively good compared to the best-found solutions of the GA. Hence, management could decide that this gap in performance is acceptable on the short- to mid-term, especially considering the improvements compared to the random (worst-case) and the single-pass (current) solution. Finally, we also solve the mathematical model presented in Sect. 3.4 using Gurobi to obtain a good benchmark for the (meta)heuristics as shown by the dotted horizontal line in Fig. 8. Due to the size of the case study, no optimal solution could be obtained within a feasible time limit using the mathematical solver. The time limit was set based on the computational run times observed for the other approaches and slightly exceeded the maximum run time required by the GA as shown in Table 7, namely a stopping criteria equal to 1,800 s. The results show that our GA can find slightly better solutions than the mathematical solver within similar time limits, which again highlights the complexity of the case study and the relative solution quality of our GA framework. An additional advantage of the GA is the robustness of this framework for changing objectives, where the single- and multi-pass heuristics from the literature have been developed with a sole objective (i.e. makespan minimisation) in mind. Therefore, we will investigate the performance of this framework for three different objectives in the next section.

5.3 Different objectives

Initially, the basic problem formulation of Sect. 3.2 is solved with the standard makespan objective, labelled GA1, and the results are shown in Fig. 9. This objective has always been the research focus of many scheduling problems in the production literature (Li and Gao 2016) and can also serve as a surrogate for the minimised cost objective (Blackstone et al 1982). In the following experiments, however, different bi-objective variants of the flexible FSSP will be investigated by modelling penalties for non-idle setup times (GA2), total & non-idle setups (GA3) and idle times (GA4), and these results will be compared with the basic makespan objective. Based on preliminary experiments and after consultation with management, we opted to penalise the iddle time with a factor equal to 1 such that it is weighed equally compared to the makespan objective, and penalise the setup times with a factor equal to 5.

Footnote 3 (continued)

the overall performance of the LN heuristic. We observe that these special cases only occur in 4% of all cases, but we still opt to show the results without excluding these special cases (Multi-pass (a) in Fig. 8) and with the exclusion of these special cases (Multi-pass (b) in Fig. 8).

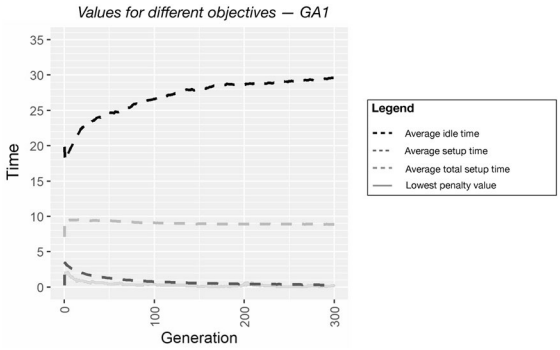

Fig. 9 Performance of GA1

5.3.1 Non-idle setups (GA2)

We test the effect of using a penalty for the non-idle setup times besides the traditional makespan objective, labelled GA2. Considering these setup times should decrease the makespan by decreasing the waiting time to start the production process. Also, the more setups are allocated to idle time, the better the utilisation rate of the machines. The results for GA2 are presented in Fig. 10 and should be compared with GA1. The results show that the decrease in setup time (the dotted dark grey line) is somewhat faster in GA2, while the idle times (dotted black line) do not increase as much in GA2. However, the rather limited decrease in setup time implies that GA1 is able to decrease the non-idle setup times in a satisfiable way even without using a dedicated penalty function.

5.3.2 Total & non-idle setups (GA3)

Decreasing the total setup time is possible by rescheduling different operations with the same or similar configuration on the same machine such that less changes in moulds, gas composition and packaging are required. We include these setups in the objective function besides the traditional makespan objective, labelled GA3. These results for GA3 are shown in Fig. 11 and should be compared with solely focusing on the non-idle setups (GA2). In GA3, the overall setup time is significantly decreased compared to only penalising the non-idle setup time (GA2) or not using a penalty at all (GA1). It is shown that taking into account both makespan and total setup time leads to slower convergence, but drastically improves the overall solution quality. As a result, GA3 is preferred over GA2 since the idle time and the total setup time are lower for similar makespan values.

5.3.3 Idle time (GA4)

In the last experiment, we aim to decrease the total setup time without drastically increasing the idle times by penalising these idle times in combination with a traditional makespan objective, labelled GA4. The results for GA4 are shown in Fig. 12.

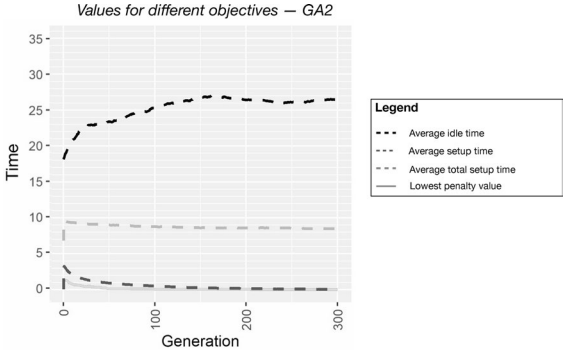

Fig. 10 Performance of GA2

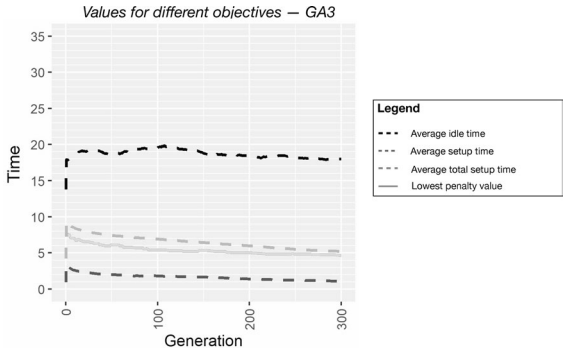

Fig. 11 Performance of GA3

Since inventory is not taken into account in our case study, idle times are less prominent. However, the consideration of idle times would become more important in case that the unavailability of some raw product in inventory could incur waiting times. We observe that both the idle times as well as the total and non-idle setup times significantly decrease compared to GA1, yet they are slightly higher compared to GA3. The main drawback of GA4 is the increase in makespan (almost 7%), which implies that decreasing the idle time comes at a cost. Hence, there is a clear trade-off between balancing the idle time and the makespan that is more prominent than the trade-off between minimising the makespan and the total setup time. In Table 9, we present the average objective values after 500 generations of GA1, GA3 and GA4.[4] The bold values highlight the best variant of GA for each objective. Three main observations can be made.

- The more performance measures (total and non-idle setup times as well as idle time) are included in the objective, the higher the makespan.
- Penalising the total setup time allows us to significantly decrease the overall setup time, while only a slightly higher makespan and non-idle setup time is observed.
- Penalising the total setup time and idle time allows us to significantly decrease both, however, resulting in a significantly increased makespan and non-idle setup time.

[4] Since GA2 and GA1 were not significantly different, GA2 is not included in this analysis.

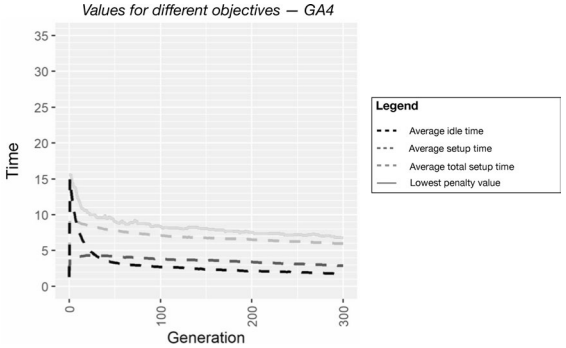

Fig. 12 Performance of GA4

Table 9 Trade-offs between the different objective functions

Objectives	GA1	GA3	GA4
Makespan	**82**	84	90
Total setup time	9	**5**	6
Non-idle setup time	≤ **1**	1	2.5
Idle time	30	18	**2**

The best values for the different objectives are given in bold with lower values being preferred

5.4 Evaluating performance

The performance of the four variants of GA discussed in the previous section is also compared with two benchmarks considering three different objectives: makespan, total setup time and idle time. The first benchmark considers the average performance of randomly generated solutions, while the second benchmark uses the average performance of NEH solutions. The results are presented in Fig. 13.

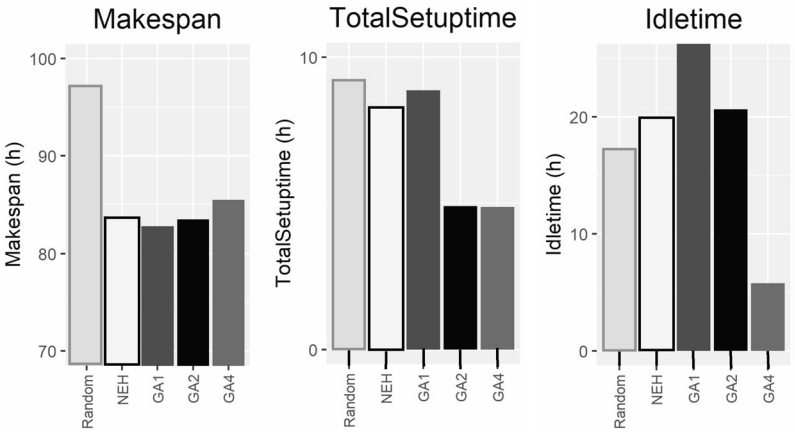

Fig. 13 Comparing the performance of GA1, GA3 and GA4 with two benchmarks (Random and NEH)

Comparing the two benchmarks, it can be seen that NEH outperforms the random solutions in terms of makespan and total setup time, however, the idle times are slightly higher. Furthermore, we observe that GA1 outperforms NEH when we only consider the makespan objective. In case that both the makespan and total setup time are considered simultaneously, GA3 outperforms both benchmarks as well as GA1. However, this comes at the cost of a higher idle time in both GA1 and GA3 compared to the random and NEH benchmarks.

Finally, we make two observations about the behaviour of the variants of the GA for different instance sizes based on the number of orders and the order sizes. First, larger instances have larger setup times and exponentially larger idle times when GA1 is used. Second, there is no significant effect of the instance size on the total setup and idle time when GA4 is used. In summary, we argue that including the penalised total setup time and idle time in the objective function results in a good trade-off between the different objectives for real world production settings of various sizes.

6 Conclusions and future research

Three approaches to solve the complex flexible flow shop scheduling problem (FSSP) have been developed to solve a real world case in a large Belgian seafood processing plant. The operations in this plant are conducted on two sequential workstations. The first workstation consists of three uniquely distinct lines (with specialised machines): a filleting line, a salmon line and a semi-finished products line. In the second workstation, seven similar machines for packaging the products from the first workstation are used in parallel. If a change in mould, gas or packaging is required, a sequence-dependent setup time (SDST) must be taken into account. A worst-case and current (easy and fast single-pass heuristic) solution applied in the plant are compared with an improved solution (without the need for large investments) and a best-case solution (requiring an investment in new algorithms/software). Different variants of a genetic algorithm (GA) that is adapted to the specifications of the seafood processing plant was developed to obtain high-quality solutions and provide management with insights on the impact of various objective functions on the operations' key performance indicators. The makespan is considered as a quality objective for perishable goods, while both setup times and idle times are included as operational efficiency objectives. These different variants of the GA were compared with two benchmarks to validate the good performance of the GA in terms of all three objectives. It was shown that GA3 and GA4 clearly outperformed the NEH and GA1.

There are two research directions that could be investigated further in the near future. First, a comparative study between different methodologies, such as Particle Swarm optimisation (PSO) or a hybridisation of PSO and GA, could be conducted on more and different real world cases. Also, the performance of a job-based operation sequence representation and restart mechanism in these metaheuristic frameworks could be considered. Second, some extensions of the flexible FSSP with SDST could be investigated. For example, the impact of the variable workforce levels in the filleting and semi-finished products lines or the impact of dynamic order arrivals.

Appendix

Appendix 1: List of acronyms

Below, we have provided an overview table with all the acronyms used throughout our research study (Table 10) :

Table 10 Acronym table

Acronym	Meaning
Problem	
FSSP	Flowshop scheduling problem
JSSP	Jobshop scheduling problem
SDST	Sequence-dependent setup times
Methods	
GA	Genetic algorithm
GA1	Genetic algorithm with makespan objective
GA2	Genetic algorithm with penalty for non-idle setup times
GA3	Genetic algorithm with penalty for total and non-idle setups
GA4	Genetic algorithm with penalty for idle times
(P)PSO	(Parallel) Particle swarm optimisation
SA	Simulated annealing
MODVOA	Multi-objective discrete virus optimisation algorithm
NEH	Nawaz-Enscore-Ham
LN	Logendran and Nudtasomboon
Operators	
MSXF	Multi-step crossover fusion operator
POX	Precedence-operation crossover
OB2X	Order-based two-point crossover
SJOX	Similar job order crossover
SJ2OX	Similar job two-point order crossover
SBOX	Similar block order crossover
SB2OX	Similar block two-point order crossover
JBX	Job-based crossover
Operational	
MAP	Modified Atmosphere Packaging
LPT	Longest processing time
FPC	Forbid parent if child is selected
FCP	Forbid child if parent is selected
FCL	Forbid clones
NON	No diversity measures activated
ALL(–)	All diversity measures activated (except FCL)

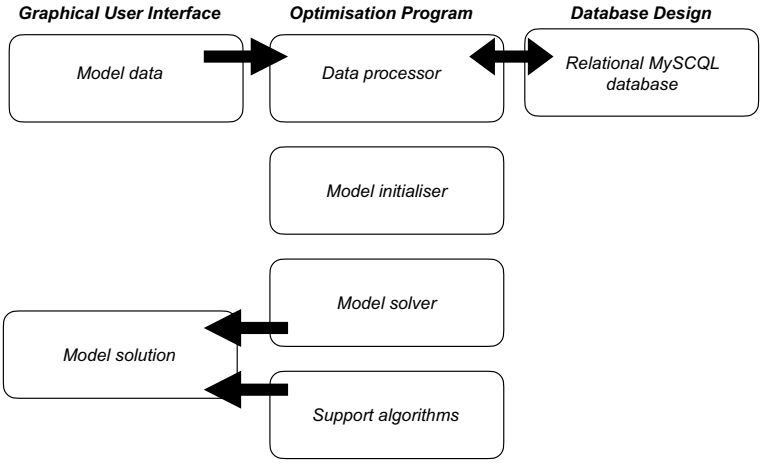

Fig. 14 Overview of implementation design at the company

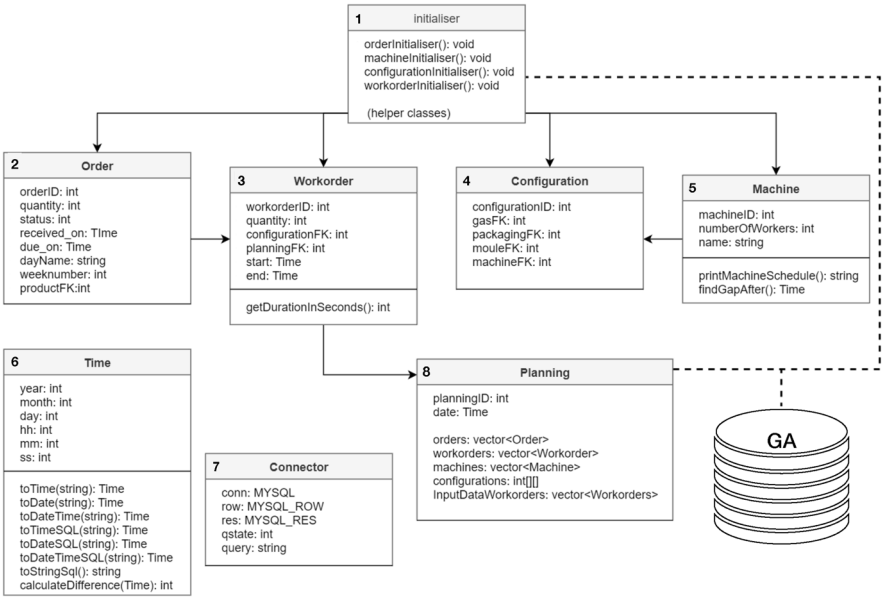

Fig. 15 Overview of implemented program at the company

Appendix 2: Implementation procedure details

Below, we have provided an overview figure of the implementation design as well as the structure of the implemented program (GA) at the company.

Acknowledgements This work was supported by the Fonds of Wetenschappelijk Onderzoek (FWO) under Grant No. 12A4222N.

Author contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Thibaut Verbanck. The first draft of the manuscript was written by Thibaut Verbanck. All authors have read and approved the final manuscript.

Funding This work was supported by the Fonds of Wetenschappelijk Onderzoek (FWO) under Grant No. 12A4222N.

Data availibility The data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

Abdelmaguid TF (2015) A neighborhood search function for flexible job shop scheduling with separable sequence-dependent setup times. Appl Math Comput 260:188–203

Amirteimoori A, Kia R (2023) Concurrent scheduling of jobs and agvs in a flexible job shop system: a parallel hybrid pso-ga meta-heuristic. Flex Serv Manuf J 35(3):727–753

Azzouz A, Ennigrou M, Said LB (2016) Flexible job-shop scheduling problem with sequence-dependent setup times using genetic algorithm. In: ICEIS (2), pp 47–53

Balin S (2011) Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Inf Sci 181(17):3551–3569

Blackstone JJR, Phillips D, Hogg G (1982) A state-of-the-art survey of dispatching rules for manufacturing job shop operations. Int J Prod Res 20:27–45

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM comput surv (CSUR) 35(3):268–308

Burmeister SC, Guericke D, Schryen G (2023) A memetic nsga-ii for the multi-objective flexible job shop scheduling problem with real-time energy tariffs. Flexible Serv Manuf J 36:1–41

Calleja G, Pastor R (2014) A dispatching algorithm for flexible job-shop scheduling with transfer batches: an industrial application. Prod Plan Control 25(2):93–109

Chaudhry IA, Khan AA (2016) A research survey: review of flexible job shop scheduling techniques. Int Trans Oper Res 23(3):551–591

Defersha FM (2011) A comprehensive mathematical model for hybrid flexible flowshop lot streaming problem. Int J Ind Eng Comput 2(2):283–294

Dorigo M (1992) Optimization, learning and natural algorithms. Ph D Thesis, Politecnico di Milano

Dueck G, Scheuer T (1990) Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 90(1):161–175

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the sixth international symposium on micro machine and human science, Ieee, pp 39–43

Feo TA, Resende MG (1995) Greedy randomized adaptive search procedures. J Global Optim 6:109–133

Fogel LJ (1962) Toward inductive inference automata. In: IFIP Congress, pp 395–400

Foo Y, Takefuji Y (1988) Stochastic neural networks for job-shop scheduling: Parts 1 and 2. In: Proceedings of the IEEE international conference on neural networks, pp 275–290
Gao K, Suganthan P, Pan Q et al (2014) Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. Inf Sci 289:76–90
Geyik F, Elibal K (2017) A linguistic approach to non-identical parallel processor scheduling with fuzzy processing times. Appl Soft Comput 55:63–71
Glover F, Greenberg HJ (1989) New approaches for heuristic search: A bilateral linkage with artificial intelligence. Eur Comput Oper Res 39(2):119–130
Glover F, Laguna M, Martí R (2000) Fundamentals of scatter search and path relinking. Control Cybern 29(3):653–684
Gong X, Deng Q, Gong G et al (2018) A memetic algorithm for multi-objective flexible job-shop problem with worker flexibility. Int J Prod Res 56(7):2506–2522
Gupta JN, Krüger K, Lauff V et al (2002) Heuristics for hybrid flow shops with controllable processing times and assignable due dates. Comput Oper Res 29(10):1417–1439
Hansen P, Mladenović N (2001) Variable neighborhood search: Principles and applications. Eur Comput Oper Res 130(3):449–467
Hasani A, Hosseini SMH, Sana SS (2022) Scheduling in a flexible flow shop with unrelated parallel machines and machine-dependent process stages: Trade-off between makespan and production costs. Sustain Anal Model 2:100010
Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press
Hong TP, Huang CM, Yu KM (1998) Lpt scheduling for fuzzy tasks. Fuzzy Sets Syst 97(3):277–286
Hopp WJ, Spearman ML (2011) Factory physics. Waveland Press
Huang PY, Hong TP, Kao CY (2006) Combining single-pass and multiple-pass heuristics for group flexible flow-shop scheduling problems. In: 2006 IEEE international conference on systems, man and cybernetics, pp 3901–3906
Jayamohan M, Rajendran C (2000) A comparative analysis of two different approaches to scheduling in flexible flow shops. Prod Plan Control 11(6):572–580
Kato ERR, de Aguiar Aranha GD, Tsunaki RH (2018) A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and random-restart hill climbing. Comput Ind Eng 125:178–189
Kirkpatrick S, Gelatt CD Jr, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
Kurdi M (2017) An improved island model memetic algorithm with a new cooperation phase for multi-objective job shop scheduling problem. Comp Ind Eng 111:183–201
Li X, Gao L (2016) An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. Int J Prod Econ 174:93–110
Li ZT, Chen QX, Mao N et al (2013) Scheduling rules for two-stage flexible flow shop scheduling problem subject to tail group constraint. Int J Prod Econ 146(2):667–678
Logendran R, Nudtasomboon N (1991) Minimizing the makespan of a group scheduling problem: a new heuristic. Int J Prod Econ 22(3):217–230
Lu C, Li X, Gao L et al (2017) An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times. Comput Ind Eng 104:156–174
Mokhtari H, Kazemzadeh R, Salmasnia A (2011) Time-cost tradeoff analysis in project management: an ant system approach. IEEE Trans Eng Manage 58:36–43
Murata T, Ishibuchi H, Tanaka H (1996) Genetic algorithms for flowshop scheduling problems. Comput Ind Eng 30:1061–1071
Nawaz M, Enscore EE Jr, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11(1):91–95
Neufeld JS, Gupta JN, Buscher U (2016) A comprehensive review of flowshop group scheduling literature. Comput Oper Res 70:56–74
Nouiri M, Bekrar A, Jemai A et al (2017) Two stage particle swarm optimization to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns. Comput Ind Eng 112:595–606
Petrov VA (1966) Flowline group production planning. Business Publications, London
Pinedo M (2008) Scheduling - Theory, Algorithms, and Systems, 3rd edn. Springer Publishing Company, Cham

Ponnambalam SG, Aravindan P, Chandrasekaran S (2001) Constructive and improvement flow shop scheduling heuristics: an extensive evaluation. Prod Plan Control 12(1):335–344
Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution, frommann–holzboog. Stuttgart-Bad Cannstatt 47
Reeves C (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22:5–13
Reeves CR (1995) A genetic algorithm for flowshop sequencing. Comput Oper Res 22(1):5–13
Reeves CR, Yamada T (1998) Genetic algorithms, path relinking, and the flowshop sequencing problem. Evol Comput 6(1):45–60
Rossi A, Dini G (2007) Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method. Robot Comput Integrated Manuf 23(5):503–516
Rossi A, Pandolfi A, Lanzetta M (2014) Dynamic set-up rules for hybrid flow shop scheduling with parallel batching machines. Int J Prod Res 52(13):3842–3857
Ruiz R, Maroto C (2006) A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. Eur J Oper Res 169(3):781–800
Ruiz R, Vazquez-Rodriguez J (2010) The hybrid flow shop scheduling problem. Eur J Oper Res 205:1–18
Ruiz R, Maroto C, Alcaraz J (2005) Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur J Oper Res 165(1):34–54
Saqlain M, Ali S, Lee J (2023) A monte-carlo tree search algorithm for the flexible job-shop scheduling in manufacturing systems. Flex Serv Manuf J 35(2):548–571
Singh MR, Mahapatra SS (2014) A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks. Int J Ad Manuf Technol 62(1):267–277
Stützle T (2006) Iterated local search for the quadratic assignment problem. Eur J Oper Res 174(3):1519–1539
Zandieh M, Dorri B, Khamseh AR (2009) Robust metaheuristics for group scheduling with sequence-dependent setup times in hybrid flexible flow shops. Int J Adv Manuf Technol 43(7):767–778
Zhang Y, Wang J, Liu Y (2017) Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact. J Clean Prod 167:665–679
Zheng DZ, Wang L (2003) An effective hybrid heuristic for flow shop scheduling. Int J Adv Manuf Tech 21(1):38–44
Zobolas G, Tarantilis CD, Ioannou G (2008) Exact, heuristic and meta-heuristic algorithms for solving shop scheduling problems. In: Metaheuristics for Scheduling in Industrial and Manufacturing Applications. Springer, p 1–40

Tom Servranckx is an Assistant Professor at Ghent University. His major research interests include the optimization of project baseline schedules, and more precisely, the incorporation of alternative technologies.

Thibaut Verbanck is a Business Architect at Base2Build. His major interests include the optimization of project baseline schedules and the integration with risk analysis and project control.

Jie Song is an Assistant Professor at the Dalian Univeristy of Technology. Her research interests include advanced methods for project control.

Mario Vanhoucke is a Full Professor at Ghent University. His research interests include the integration of project scheduling, risk management and project control using algorithmic modelling and optimization.

Authors and Affiliations

Tom Servranckx[1] · Thibault Verbanck[1] · Jie Song[2] · Mario Vanhoucke[1,3,4]

✉ Mario Vanhoucke
mario.vanhoucke@ugent.be; mario.vanhoucke@vlerick.com; m.vanhoucke@ucl.ac.uk

Tom Servranckx
tom.servranckx@ugent.be

Thibault Verbanck
thibaut.Verbanck@UGent.be

Jie Song
jieson.song@ugent.be

[1] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Ghent, Belgium

[2] School of Economics and Management, Dalian University of Technology, No.2 Linggong Road, Dalian, China

[3] Operations and Technology Centre, Vlerick Business School, Reep 1, 9000 Ghent, Belgium

[4] UCL School of Management, University College London, 1 Canada Square, London E14 5AA, UK