Master´s Thesis for the Attainment of the Degree
Master in Mechanical Engineering and Management
at the Faculty of Mechanial Engineering of the Technische Universität München

# Analytically evaluating task sharing and sequencing in an unpaced flow line on the operational level using an analytical model

| | |
|---|---|
| Referent: | Prof. Dr. Martin Grunow<br>Production and Supply Chain Management<br>Technische Universität München |
| Betreuer: | M.Sc. Lena Böttcher |
| Studiengang: | Master in Mechanical Engineering and Management |
| Eingereicht von: | Javier Dolz Cifre<br>Plaza de los Santos Niños 6 , 1A<br>28801 Madrid<br>Matrikelnummer: 03709613 |
| Eingereicht am: | 1. October 2020 |

Task sharing helps to reduce imbalances in mixed-model lines, derived from blocking and starving stations. On the other hand, intelligent sequencing also gives us the possibility to face these waiting times and increase the efficiency of our line. Even though both techniques have been widely studied in the past, no attention was paid to the interaction of both of them. This thesis focuses on the influences that task sharing and sequencing have between each other when studying the makespan in a mixed model assembly line.

*Keywords* : *task sharing*, *sequencing*, *mixed−model lines*, *unpaced flow lines*

# Table of Contents

# List of Figures

# List of Tables

# 1 Introduction

The main source of imbalance in mixed model lines is different tasks and different task times for different models (Bukchin et al.,1997). Also, imbalance can be caused by the variability due to operators or machine breakdowns (Gel et al., 2002). Results of unbalanced assembly lines are, for example, starving and blocking stations. If tasks are finished at a station while other tasks are being performed at the downstream station, the upstream station gets blocked, as it is not possible to pass the product downstream. On the other hand, when the downstream station has completed its tasks before the upstream station is finished, the downstream station starves (Conway et al., 1988).

Task sharing provides an opportunity for reducing this unbalance through the assignment of shared tasks between stations, individually for each model in a sequence. To decide which tasks can be shared, this thesis will distinguish between fixed and shareable tasks. Fixed tasks are tasks that can only be performed in a particular station, as they require tools that can only be available at one of the stations. Shareable tasks are tasks that can be performed in more than one station. Shareable tasks create a degree of freedom in an assembly line that allows to decide where to perform a task in order to optimize the production in the line according to a given objective ( for example, minimizing the makespan, maximizing fairness, minimizing cost and operational complexity, etc.). Another strategy to reduce starving and blocking times is the implementation of buffers (Conway et al. 1988; McClain et al., 2000), which are included in the system investigated in this thesis.

On the other hand, we can also reduce waiting times by finding sequences that make the system more efficient (Freiheit and Li, 2017). By a proper resequencing of the products that are about to enter the production line, we can help to avoid bottlenecks and distribute the workload between stations so that we obtain a continuous flow and maximize the efficiency of the line.

Existing literature mainly focuses in paced lines (Hudson et al., 2015), where the impact of the workers on the work pace is very restricted (Gosh et al., 1989, p.652). This Master thesis will focus in an unpaced flow line with consecutive workstations that is used as a mixed-model line, where distinct models of a product are produced on the same assembly line without changeovers.

The main objective of this Master thesis is to evaluate the effects of task sharing and sequencing in an unpaced flow line on the operational level using an analytical model. In order to achieve this main objective, a research in the existing literature will be done. This literature research will focus on previous cases of task sharing, to give a general definition of the concept and to look at examples where task sharing or sequencing were suggested to solve problems in the industry, together with the limitations that other studies faced when studying task sharing. The literature research will not only derive the research gap of the thesis, but will also help formulate several research questions to be adressed within the content of this work. Thanks to the literature research, we will not only understand how have task sharing and sequencing been used independent from one another before, but also we will have a first insight on how a combination of both strategies looks like.

The optimization of the assembly line will be progressively analyzed by solving different models. Starting from a basic model that measures the makespan of the unpaced flow line, different sequencing formats and different degrees of task sharing will be combined to have a global scope on the effects of both sequencing and task sharing on the line, as well as the mutual interaction of both strategies.

The results are analysed and compared with the literature research to understand the viability of the solution and its possibilities to be directly applied in the industry, deriving managerial insights on the benefits of task sharing to improve the performance of the line for the given objectives

# 2 Review of Literature and Research

In this section, a review of the previous work in task sharing and sequencing is made to show the state-of-the-art in this field. Section 2.1 shows the definition of task sharing in literature and its classification, together with previous cases where task sharing was applied to solve problems in the past. Section 2.2 shows previous studies about sequencing in Mxed Model Assembly Lines. Section 2.3 focuses in the combination of sequencing and task sharing, concluding in the research gap and research questions raised after this literature research.

## 2.1 Definition of task sharing in literature

Task sharing is a process where two or more people collaborate and share the responsibility for the performance of their work. Task sharing groups usually represent a mix of skills and know-how so that each individual can work on different tasks. The main reasons for task sharing are the need for acceleration during a process or the need for teamwork when one of the group members cannot complete his or her tasks. To create a collaborative system, we need that the group members can clearly understand their portion of work and their allocation in the whole process. Also, task sharers need to understand and agree upon their commitments and mutual responsibility and they need to be communicated beforehand who is going to make the decisions on which tasks to be shared according to the situation, as well as supporting and coaching each other because individual failure leads to team failure.

In traditional assembly lines, work sharing is not allowed. Consequently, the cycle time is determined by the amount of work performed in the most loaded station [Anuar and Bukchin (2006)]. Task sharing refers to the ability to perform a certain task in more than one station or by more than one worker, which helps to balance the line by decreasing the load in the station with a bigger workload. To make this sharing possible, workers need to be cross-trained, because some tasks can now be performed either last in station i or first in station i+1. Task sharing was not studied until the 1990s. A review on the previous studies in task sharing is done along the following sections.

To understand how task sharing is going to be implemented in this thesis we need to describe the environment of our task sharing system. This thesis focuses on an unpaced mixed-model flow line in the manufacturing industry. In a flow line, several stations stand next to each other in series. This means that each product that enters the line follows the same path. In other words, it must go through every station consecutively [Pinedo (2010)]. When buffer stocks are placed between these stations to allow the accumulation of workpieces, the production line is described as unpaced. Each station is surrounded by one upstream buffer and one downstream buffer, except the first and the last. Therefore, a worker takes a product upstream, he performs the required tasks for this product in this station and sends the product downstream. The term mixed-model line refers to a production line where different variants of the same product can be manufactured without requiring big set up times. The different products entering the line will receive mostly the same type of tasks and will be differentiated by a fewer number of more specific tasks, which will also create a difference in their processing times.

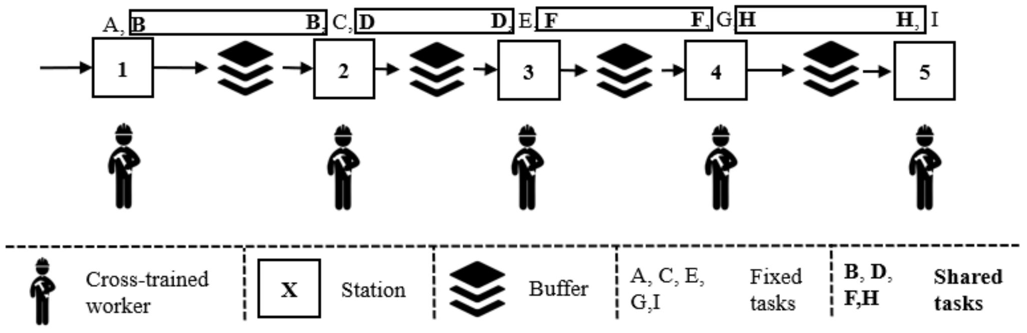


Figure 2.1: Representation of the task sharing system

As shown in figure 2.1, five workers are assigned to five work stations. For producing an incoming product, nine tasks, A, B, C, D, E, F, G, H, I are performed one after the other. In this flow line, each worker has to work in his corresponding fixed task, A for worker one, C for worker two, E for worker three, G for worker four and I for worker five. Also, these workers share the shared tasks B, D, F and H. This means that task B can be done by worker one or worker two, task D can be done by the second or third worker, task F can be done by the third or the fourth worker and task H can be performed by the fourth or the fifth worker.

### 2.1.1 Relevant classification of task sharing systems

There are different approaches in the literature regarding the sharing of tasks [Bratcu et al. (2009)]. To classify task sharing in its different approaches, several distinctions need to be made.
First, workers could be moving or they could be fixed in their station. When a worker is fixed at a work station, this worker is called stationary.
Another criterium is the staffing level, in other words, how many workers do we have per station [Xu et al. (2011)]. In this context, we classify the staffing level in three different orders. When the staffing level is below 100 per cent it means that there are fewer workers than workstations. If the staffing level is 100 per cent, the

system has as many workers as workstations, in other words, there is one worker per station. If the staffing level is above 100 per cent, the flow line has more workers than workstations.

Also, some authors allow self-balancing and collaboration between workers from different stations in their models. When collaboration takes place, several workers are allowed to work simultaneously at the same station. Self-balancing occurs in systems where there are fewer workers than stations (staffing level below 100 per cent). Therefore, workers in self-balancing systems are allowed to move along the different work stations. This type of production systems are usually ruled by a bucket brigade method that aims to balance the workload itself. A bucket brigade is a linear production line in which each worker picks up a job and processes it at each station until he gets interrupted by a downstream worker [Armbruster et al. (2007)]. Figure 2.2 has been added to support the explanation.

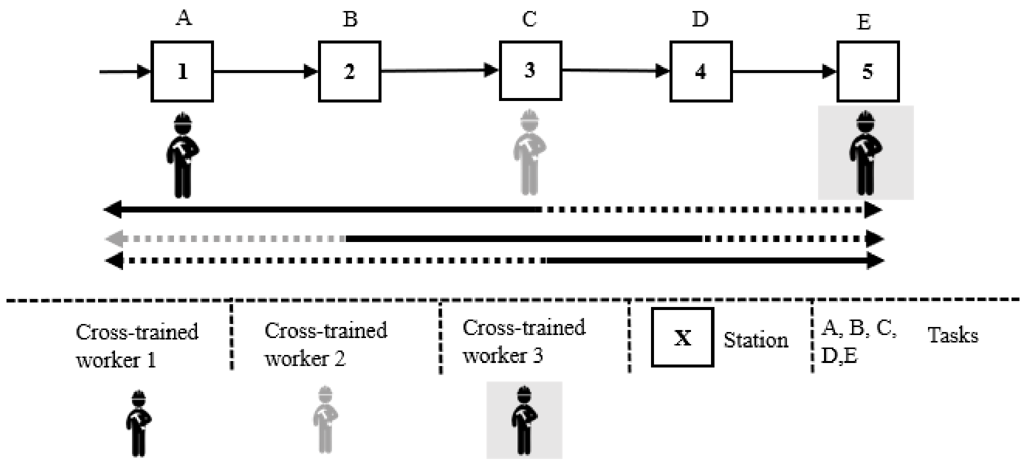


Figure 2.2: An example of a bucket brigade manufacturing system

Figure 2.2 shows a system with five work stations, where three workers move along the flow line to do tasks A to E in their correspondent station. In this example, the first worker covers stations 1 to 3, the second worker covers stations 2 to 4 and the last one covers stations 3 to 5. However, each worker is allowed to move to any station although each of them is assumed to be usually working within their designated area. A common situation in this system could be the following: the first worker finishes the task A at station one. Then he continues working on the product at station two, performing task B. However, while he is approaching station 3, he gets interrupted by worker two, who has already finished task C for a previous job and was moving backwards in the line to reach worker one and take over a new job. Worker two performs task C for the new job, while worker one is moving back to station one to start another job and moves to station 4 but he is interrupted by worker three, who was moving backwards after finishing task E for the previous job at station 5. Worker three performs tasks D and E to finish the job.

The bucket brigade system is a great example to show the importance of the worker know-how as he can perform very different tasks and the importance of his speed. It has been studied by Bartholdi et al. (1999) that if workers can be sequenced from slowest to fastest so that each worker is faster than his predecessor at every point along the line, the bucket brigade system helps the line balance itself and this will yield optimal throughput. Based on his work together with the contribution of Zavadlav et al. (1996), it is understood that, even though work sharing lead to benefits in this system, this only happens with a calculated overlap which leads to sharing less than the half of tasks. An excess of shared tasks leads to a decrease in efficiency. Further research on this system has been done by Buzacott (2002). His work showed that optimality of bucket brigades systems depends on the preemption of tasks (the concept of preemption and is explained at the end of this section together with some examples). Without preemption, the full theoretical advantages of dynamic task sharing would be hard to achieve in practice. Further research was done by Armbruster et al. (2007) who studied the influence that a new untrained worker would have in a bucket brigade system and Bratcu et al. (2009) presented their Normative Model where they did not add the restriction regarding the infinite backward velocity of workers and they developed a simulation model in Matlab/Simulink. The most recent work on this topic was done by Pratama et al. (2018). They stressed the importance of collaboration between workers in the bucket brigade system. Collaboration means that several workers are allowed to work at the same station at the same time. Collaboration makes sense with staffing levels of at least 100 per cent and it appears in every system where the staffing level is above 100 per cent. Buzacott (2004) created models that suggest that the advantage of teams lies specially in the structure that enables faster and better workers to help out others. Hopp et al. (2004) also suggested that allowing only one worker to perform tasks at a station at any given time in Bucket Brigade systems tends to cause blocking. Sennott et al. (2006) included the figure of floating worker in their research. This is an experienced and cross-trained worker that is ready to help in any station that is slowing the whole line down at a particular moment. Ahn et al. (2006) concluded in their research in dynamic load balancing with flexible workers that under general conditions there is a tendency for workers to work together under the optimal policy. Therefore, the collaboration between workers is helpful. The paper also reminds us of the importance of establishing sharing rules between the collaborative workers so that they know who should be doing each task or who must lead the action. An interesting research about collaboration in Tandem Lines was performed by Andradottir et al. (2001), Andradottir et al. (2005) and Andradottir et al. (2007) with a total of three papers where they examined flow lines in different types of systems with different staffing levels and they ended up comparing the performance of these collaborative systems with systems that have extra flexibility obtaining similar results for both cases.

On the other hand, task sharing approaches can be grouped by its characteristics in

online task sharing, offline task sharing and preemptive task sharing. Online task sharing systems refer to those where the complete sequence is not known in advance and decisions about the assignment of tasks to stations are made while the production is running. Most of the papers in this field reference the work of Ostalaza et al. (1990) and McClain et al. (1992) as first approaches. Online task sharing was then studied by Gel et al. (2002), who solved a task sharing problem using a Markov decision process formulation in a flow line with two stations and one shareable operation between both. The same procedure with Markov decision formulation was used by Gel et al. (2007), where only one of the workers in the line was cross-trained to perform different tasks. New task sharing decision rules involving the starvation of the line and design aspects of the system was introduced by Askin et al. (2006) to test the performance with different amounts of cross-trained workers.
Offline task sharing refers to systems where the decisions are not made in real-time and the solution is more related to the intelligent sequencing of the line. Offline task sharing has mainly been modelled in flow lines with two stations. The first example of this is the research done by Gupta et al. (2004), where they faced the classical problem of minimizing the makespan in a two-machine flow shop including a shareable task between both machines. A two-station flow line was also modelled by Gultekin (2012). He examined two stations with different processing times, in contrast to the model from Gupta et al. (2004) and he concluded that most of the benefits respect to reducing the makespan are achieved with a relatively small level of flexibility. His work can be extended by increasing the number of stations of the flow line or considering other objective functions other than the makespan. Afterwards, Uruk et al. (2013) investigated the two-station flow line with the objectives of minimizing both the makespan and the manufacturing costs, allowing worker collaboration and considering the controllability of the processing times of each operation on these machines. Lin et al. (2017) showed the last approach, where multiple objectives were pursued by dynamic programming algorithms subject to a fixed job sequence, for example, they proved the NP-hardness of the problem under study.
Preemptive task sharing systems are those where a task can be started in a machine and finished in the next one, allowing more flexibility to the solution. Burdett and Kozan (2001) examined them, creating a general model suitable for modelling any task sharing scenario in a flow line and leaving their work open for future research in the direction of assembly line balancing. Their research was followed by Bultmann et al. (2018a), who focused on flow shop problems where processing times where not fixed in advance but chosen with flexibility. They introduced a general model in which processing times can be distributed freely if the machines have some predefined boundaries in their possible processing times. The paper proposes different variants of flow shop models such as no-wait flow shops, blocking flow shops and synchronous flow shops. In the same year, Bultmann et al. (2018b) focused their research on synchronous flow shop scheduling problems for flow shops with pliable

jobs and showed that, while minimizing the makespan in standard synchronous two-machine flow shops can be done in polynomial time, the problem becomes NP-hard whenever pliable jobs appear.
The following tables are a summary of the information exposed above.

Table 2.1: Relevant classification of task sharing systems

| Authors | Workers | Staffing level | Task sharing |
|---|---|---|---|
| *Bartholdi et al.* (1999) | Moving | <100 | Self-balancing |
| *Zavadlavet al.* (1996) | Moving | <100 | Self-balancing |
| *Buzacott* (2002) | Moving | <100 | Self-balancing |
| *Armbruster et al.* (2007) | Moving | <100 | Self-balancing |
| *Bratcu et al.* (2009) | Moving | <100 | Self-balancing |
| *Pratama et al.* (2018) | Moving | <100 | Self-bal./Collab. |
| *Andradottir et al.* (2001) | Moving | 100 | Collaboration |
| *Buzacott* (2004) | Moving | 100 | Collaboration |
| *Hopp et al.* (2004) | Moving | 100 | Collaboration |
| *Andradottir et al.* (2005) | Moving | >100 | Collaboration |
| *Ahn et al.* (2006) | Moving | 100 | Collaboration |
| *Sennott et al.* (2006) | Moving | >100 | Collaboration |
| *Andradottir et al.* (2007) | Moving | >100 | Collaboration |
| *Ostalaza et al.*(1990) | Stationary | 100 | Online |
| *McClain et al.*(1992) | Stationary | 100 | Online |
| *Gel et al.* (2002) | Stationary | 100 | Online |
| *Askin et al.* (2006) | Stationary | 100 | Online |
| *Gel et al.* (2007) | Stationary | 100 | Online |
| *Gupta et al.* (2004) | Stationary | 100 | Offline |
| *Gultekin* (2012) | Stationary | 100 | Offline |
| *Uruk et al.* (2013) | Moving | >100 | Offline/Collaboration |
| *Lin et al.* (2017) | Stationary | 100 | Offline |

Table 2.2: Different task sharing approaches

| Authors | Objectives | Methodology | Decisions |
|---|---|---|---|
| *Ostalaza et al.*(1990) | max TH | Simulation | Share task or not |
| *McClain et al.*(1992) | max TH | Simulation | Share task or not |
| *Gel et al.* (2002) | max TH | MDP | Share task or not |
| *Askin et al.* (2006) | max TH | Simulation | Share task or not |
| *Gupta et al.* (2004) | min Makespan | MILP | Flexible task assignment |
| *Gultekin* (2012) | min Makespan | MIP | Flexible task assignment |
| *Uruk et al.* (2013) | min Makespan | MIP | Flexible task assignment |
| *Lin et al.* (2017) | various | DP | Flexible task assignment |

Table 2.3: Different task sharing lines

| Authors | Stations | Single-model line |
|---|---|---|
| *Ostalaza et al.*(1990) | 2 to 5 | Yes |
| *McClain et al.*(1992) | 2 to 5 | Yes |
| *Gel et al.* (2002) | 2 | Yes |
| *Askin et al.* (2006) | 2 | Yes |
| *Gupta et al.* (2004) | 2 | |
| *Gultekin* (2012) | 2 | Yes |
| *Uruk et al.* (2013) | 2 | Yes |
| *Lin et al.* (2017) | 2 | |

## 2.2 Sequencing on Mixed Model Assembly lines

In some production processes like the assembling of cars, where there is a variety of products coming into the line, Mixed Model Assembly Lines are used to keep a high throughput and reduce idle times. Mixed Model Assembly Lines create a problem of sequencing, where the sequence of the next few products entering the production line is to be determined.
Sequencing has been a matter of study for many years, we can find some pioneering work in the studies done by Dar-el and Cother (1975) or Nick T. Thomopoulos (1967)), where they already focused in different objectives such as minimizing cost or reducing the blocking in the line. The literature review about sequencing is extensive and shows an immense variety of approaches. However, inside the Just-in-time context two main objectives take place [Bard et al (2007)]:
Work overload: an increase in the variety of products usually implies an increase in the number of different processing times, for example, cars with a sunroof require higher processing times than cars without one. If several cars with a sunroof enter the flow line consecutively waiting times probably increase as the work overload has suddenly increased drastically. If this happens, the operations on one of the cars may not be finished within their corresponding station boundaries and compensation is required (for example, a collaboration between workers or stopping the line). Therefore, work overload objectives pursue to find the right sequence that alternates products requiring higher and lower processing times to avoid this blocking.
Just-in-Time objectives: They focus on deviating material requirements. A great variety of models require a proper kitting of the working stations with the tools needed for processing the tasks required for each product. The Just-in-Time objectives focus on evenly smoothing the material requirements according to the production sequence over the planning horizon. For this reason, we need a sequence that adjusts the actual consumption rates of materials as much as possible to the target rates.
Both objectives were taken up by three alternative sequencing approaches Boysen et al. (2009b). Concerning work overload, in literature, we can find information on

Mixed-model sequencing and Car sequencing.
Mixed-Model sequencing: The objective is to minimize the total work overload based on a detailed schedule that takes the operation times, the movement of workers, the boundaries between stations and other operational characteristics of the flow line [Bolat (1997), Celano et al. (2004); Scholl (1999)]. The balancing of an MMAL is usually determined by the precedence graphics of the line, where the processing times that diverge from the different models are averaged [Boysen et al. (2009a); van Zante-de Fokkert and de Kok (1997)] and, to avoid an excess in capacity, the cycle time is also established as an average of the different cycle times for the different models. The consequence of is that some models require higher cycle times tan the average, while others require lower cycle times. If several models with high cycle times appear consecutively, the work overload problem mentioned above takes place, as the worker moves out of the boundaries of his station and he will not have time to move to the beginning of his station before the new model arrives at his position. To counterattack this overload, different compensation actions can take place [Scholl (1999); Wild (1972)]. For example, the flow line can be stopped until every worker has finished their actual job, a collaboration between workers can take place by the use of utility workers that help the worker who is suffering a work overload or the unfinished jobs are finished offline after going through the whole line. These actions are costly and should be avoided with a correct sequencing of the mixed models alternating between high intensive and less intensive jobs in each station. Therefore, it is needed to represent in the modelling the different stations, cycle times, processing times, movement of workers and station boundaries.
Car Sequencing Problem CSP: The car sequencing problem focuses on minimizing the number of rule violations in an assembly line. These rules are the so-called Ho: No sequencing rules, which imply that from the N jobs happening next in the sequence only Ho occurrences of a certain option o are allowed. When a sequence that does not violate the rules is found, then we can avoid the work overload or if at least the violation of rules is as low as possible the work overload can still be decreased. [ Gagné et al. (2006); Parrello et al. (1986); Smith et al. (1996)]. Car sequencing is not based on a detailed schedule as in Mixed Model Sequencing, but it considers and controls the successive models that include highly intensive options, for example, the cars with a car roof mentioned above, to minimize the work overload [Meyr (2004); Pil and Holweg (2004); Röder and Tibken (2006)]. Sequencing rules are usually determined by approaches based on the operational characteristics of the assembly line or by the experience coming from previous practical applications of the CSP in the car industry, that derive in the so-called rules of thumb, for example, from every three consecutive models, only one of them can be a model that requires a sunroof. Concerning the Just-In-Time objectives, in literature, we can find information on Level Scheduling. Level Scheduling: to follow the JIT philosophy, target consumption rates are established and a sequence is created to minimize the deviation between the actual and the target consumption rates [Aigbedo (2004); Bautista

et al. (1996)]. Level scheduling has been widely studied as part of the well-known Toyota Production System [Dhamala (2015); Kubiak (1993)], which is focused in the avoidance of waste and reducing the inventory as much as possible.

### 2.2.1 Sequencing with limited buffers

In the previously mentioned approaches the following have been assumed (Boysen et al. (2009b)):

- There are no buffers between stations. This means that the production sequence is determined before starting the production.
- The workpieces have a fixed position on the transportation system and only their orientation can be changed
- The model-mix in the planning horizon is already known and no changes are possible (static problem), so that rush orders are not possible.
- Multiple models are made of different materials and they require different tasks with individual processing times. This means that the material requirement and the capacity utilization on every station can change from one model to the other
- Resequencing is not allowed as it is supposed that there are no disturbances, like machine breakdowns or material stock-outs [Ding and Sun (2004); Inman (2003)].

As a consequence of the restrictions above mentioned there are some aspects of sequencing that are not mentioned in literature and, therefore, limit the resolution of real-life problems in assembly lines.

Focusing on the first restriction, most of the literature in sequencing assumes that the buffering capacity is infinite between workstations. However, in the real production systems, there is always a limited capacity due to the physical necessity that the jobs need to wait before being processed in a station every time this station is already full with a previous job being processed on it. This limitation becomes more relevant as the volume of jobs increase or the size of the workpieces or lots processed increases. The existence of buffers with a limited capacity between stations creates a whole new sequencing problem. For example, the Johnson algorithm is not a guarantee of optimality anymore and it is known that the starting time of the different jobs will always be equal or higher than when considering unlimited capacity between stations.
Leisten (1990) shows that sequencing problems can be described in four different types regarding their capacity: problems with unlimited capacity, problems with a limited capacity greater than cero, problems with no capacity (blocking scheduling) and no-wait scheduling problems.

Blocking scheduling takes place when it is considered that there is no storage capacity between stations. Therefore, if a job has finished his operation in one station, it can only move to the next station if the next one is free. Otherwise, blocking takes place and the job needs to wait until the next station is free. On the other hand, no - wait scheduling problems create an additional constraint for each job because the completion time of any job at one machine must coincide with the starting time of the same job at the next machine so that no job needs to wait while moving through the flow shop. Previous examples of these sequencing problem types can be found in the literature. Pioneering work in flow shops with a limited capacity is found in the paper of Dutta and Cunningham (1975), where a two-machine flow shop problem has been solved by dynamic programming. Afterwards, Papadimitriou and Kanellakis (1980) proved that the problem with limited buffer capacity was NP-complete. This was a reference point for a growing interest in this research field. These authors created a heuristic to solve the two-machine flow shop problem. Leisten (1990) was the first to create a general model for the flow shop with limited capacity scheduling problem and his work was followed by Smutnicki (1998) and Nowicki (1999). Their work is based on a flow line where only sequences based on permutation between jobs are allowed, where the processing times between machines can be denoted as $p_{ij} > 0$. Some basic assumptions were considered for this problem, for example, jobs can not be interrupted, every machine processes only one job at a time or a job can only be processed at one machine at a time. Between every two machines j-1 and j there is a buffer Bj with size b(j) where a FIFO discipline is followed. Every job goes through the buffer Bj when finishing in machine j-1 and moving to machine j. This means that, once a job is finished in machine j-1, it can only move to the buffer Bj if it is not full. In case Bj was full, the job needs to wait in the machine until there is a free spot in the buffer and, until this happens, machine j-1 is blocked and no other job can be processed on it. The starting times of every job i in machine j are described as $S_{ij}$ and the completion times are described as $C_{ij}$. Therefore, a constraint for starting every job in the next machine can be defined as:

$$S_{ij} >= S_{i\,j\text{-}1} + p_{i\,j\text{-}1} = C_{i\,j\text{-}1}$$

This means that the starting job for job i in machine j needs to be higher or equal than the starting time of the same job in the previous machine plus the processing time for this job in that machine, what is equal to the completion time of job i in machine j-1. Nowicki (1999) denoted in his formalization of the problem with using a graph as type V or Vertical relationships. On the other hand, it is also needed a constraint that ensures that the starting time of job i in machine j is greater or equal than the starting job of the previous job in the same machine plus the processing time of that job in this machine, what is equal to the completion time of job i-1 in machine j.

$S_{ij} >= S_{i\text{-}1\ j} + p_{i\text{-}1\ j} = C_{i\text{-}1}$

This type of relationships are called H type or Horizontal relationships. However, when there are capacity limitations in the existing buffers between stations a new type of relationship appears. The starting time of job i in machine j must only take place if there is a free spot for this job in the buffer b(j+1). This is represented in the constraint:

$S_{ij} >= S_{[i\text{-}b(j+1)\text{-}1]\ j+1}$

Therefore, a job can not start in a machine if job [i-b(j+1)*1] has not started in the next machine. Nowicki (1999) denotes this type of relationships as type D or Diagonals. Thanks to the three previous relationships it is possible to compute the value of $S_{ij}$ and the problem gets reduced to solving the sequence of the permutation of i so that the makespan $C_{max}$ is minimized, being $C_{max}= S_{N\ M} + p_{N\ M}$ ( starting plus processing of the last job in the last machine). Also, To represent the time in the buffer or blocking times, Pinedo (2010) defines $D_{ij}$ like the moment when job i finally leaves machine j. In this sense $D_{ij} = C_{ij}$ in lines with unlimited buffer capacity but $D_{ij} >= C_{ij}$ in Flow shops with limited buffers between stations, because of the blocking effect. This is represented in the following equation:

$D_{ij} = \max ( C_{ij}, S_{[i\text{-}b(j+1)]\ j+1})$

In the article from Leisten (1990), a study where eleven different heuristic rules are tested, one of the rules being extracted from the work of Papadimitriou and Kanellakis (1980) or the NEH rule of Nawaz et al. (1983), together with other rules are applied in direct and inverse order of the machines. Leisten (1990) concludes that NEH was the best heuristic rule exposed. In the papers from Smutnicki (1998) and Nowicki (1999), a graph is used to solve the sequencing problem, where the nodes j and i , corresponding to machine and job, represent the starting time of each job i in machine j a have the same weight as the processing time $p_{ij}$. Every arch has value cero except those that represent the relationships coming from the buffers that have a negative value for the processing time $P_{[i\text{-}b(j+1)\text{-}1]j+1]}$. Thanks to their graphs, a quick implementation was done to compute the makespan of a determined sequence. Then, they develop a Tabu search algorithm based on the relationships mentioned above to find the best permutations. The previous model was extended in Brucker et al. (2003) for general flow shops which do not impose that the permutation must be the same for every machine. A Tabu Search Algorithm is also used here. The same sequencing problem is addressed by Wang et al. (2006). However, a genetic algorithm is implemented here to study the effects of the buffer sizes in the makespan. Other papers using metaheuristics for this problem are Norman (1999), who solves the problem by Tabu Search and considers set up times dependant from the sequence. Liu et al. (2006) who used Particle Swarm Optimization. Hsieh et al. (2009) with an immune based approach. Qian et al. (2009) with a differential evolution algorithm. Pan et al. (2011) with a chaotic algorithm of harmonic search. Rossi and Lanzetta (2013) with an ant colony optimization or Abdollahpour and Rezaeian (2015) with an algorithm based in immune systems.

Table 2.4: Summary of the authors and solving processes

| Authors | Solving Process |
|---|---|
| Dutta and Cunningham (1975) | Dynamic Prog. |
| Papadimitriou and Kanellakis (1980) | Heuristic |
| Leisten (1990) | Heuristics |
| Smutnicki (1998) | Tabu Search |
| Nowicki (1999) | Tabu Search |
| Norman (1999) | Tabu Search |
| Brucker et al. (2003) | Tabu Search |
| Wang et al. (2006) | Genetic Algor. |
| Liu et al. (2006) | Particle Swarm Optim. |
| Hsieh et al. (2009) | Immune Based Approach |
| Qian et al. (2009) | Differential Evolution Algor. |
| Pan et al. (2011) | Chaotic Algor. Harmonic Search |
| Rossi and Lanzetta (2013) | Ant Colony Optim. |
| Abdollahpour and Rezaeian (2015) | Immune System |

### 2.2.2 Grouping and spacing

The solution methodologies have been generally divided into two main methodologies known as priority rule-based approaches and analytical approaches. Analytical are approaches are then divided into exact and approximate approaches. [Blackstone et al. (1982)].

**Approximate approaches** have been shown in the table that closes section 2.2.1. These are methods that do not reach optimal but reach very good solutions in affordable run time [Blazewicz et al. (2009)]. They can be split into heuristics if they face a particular decision problem, or metaheuristics if the objective is to create a generic procedure applicable in different contexts.

**Exact approaches** focus on finding the optimal solution of the respective objective function. Some examples are the branch-and-bound algorithm for integer problems or dynamic and mixed-integer programming [Blazewicz (2009) et al.]. While approximate approaches are generally seen in industrial approaches, where the mathematical modelling is softened to get a faster solution, exact approaches are usually seen in small-sized problems to avoid NP-hardness where the objective is to find the best solution possible. It is common to read in literature papers where the problem is faced at first as an exact approach and it becomes approximate afterwards to be implemented in the industry.

**Priority rule-based approaches** are those where several rules are created to build a sequence according to the priority value that every job receives. They can be used for building a sequence before the production has started or during the production as dispatching rules, deciding which job comes next in the line. Some examples of priority rules are the shortest processing time (SPT), the longer processing time (LPT), the first-in-first-out rule (FIFO) or the earliest due date (EDD). More examples are listed on the work of [Blackstone et al. (1982), Haupt et al. (1989), Jayamohan et al. (2000)]. We can also find in these papers a classification of the different rules to cluster them according to the properties of the rule. Rules involving processing time, Rules involving due dates, Rules involving shop or job characteristics, etc. Priority rules are used daily and take part of the most of the sequencing literature even if they are not explicitly mentioned, as random sequences are usually a vague approach that may lead to confusion in strongly sequent-dependent flow or job shops.

## 2.3 Task sharing at the operational level with offline sequencing

At the operational level, scheduling decisions are necessary for a production system. The scheduling problem becomes the sequencing in a permutation flow shop as long as we are not studying a no-wait system. The sequencing decisions needs to be decoupled in time between production and customer demands. In other words, we need to allow that the production of the different jobs deviates in time from the customer orders. In case we do not, the sequence would be explicitly determined by the products due dates and lead times of these orders as in a traditional make-to-order production scheduling [Yu et al.(2015)]. Section 2.2 provides a foundation for the more efficient sequencing in the assembly operations associated with a MMAL. The basis of them is the reduction of the idle time inefficiencies that arise as a result of upstream and downstream blocking of stations and the utilisation of the available multi-skilled workforce Burdett and Kozan (2001). These inefficiencies can have a huge effect on the makespan even if a good sequence is found. However, as mentioned in Section 2.1, with multi-skilled workers it is possible to reduce idle times and consequently the makespan by sharing tasks between them. It is then assumed that workers know how to perform tasks for adjacent stations or that they will be trained for it. Also, they need to have the tools needed to perform the shareable tasks in their workstation. After reviewing the literature in sequencing along section 2.2, it is seen that in the previous studies to improve the efficiency of the assembly line, the utilisation and modelling of human labour interaction has not been usually addressed. Stationary workers frequently were expected to work in closed stations and to perform several tasks that are fixed to their station. Still, most of the papers in sequencing have been done for paced flow lines, where the worker moves with the conveyor assembly line. However, as exposed in [Buzacott et al.(1990)], paced lines have been abandoned in the automobile assembly systems where most of the tasks are performed by human operators. Burdett and Kozan (2001) investigated the effects of applying the task redistribution approach for the sequencing problem by applying two solution approaches, the first by a two-step NEH [Nawaz et al. (1983)] - Simulated Annealing approach, solving first the sequence and then the task redistribution. Second, a two-step Simulated Annealing approach starting with the task redistribution and solving the sequencing in the second step. There is very few literature that focused on the combination of task sharing and sequencing and the mutual benefits both techniques could reach. One possibility is that the benefits of task sharing could have implications on sequencing rules. This means that task sharing could help to reduce the restrictions established in a Car Sequencing Problem, by a lower Ho: No ratio. The importance of workloads in the spacing approaches has been introduced before by Lesert et al. (2011), as the Ho: No are usually created after considering the variants in workload for different models. Therefore, we can distinguish between high work-intensive and low work-intensive model variants. Thanks to task sharing and the possibility to shift tasks between adjacent stations, the workload can be smoothed and this could allow a lower Ho: No ratio. Also, the Ho: No ratio is no longer fixed but variable depending on the task sharing case achieving more flexibility regarding sequence building. Car sequencing constraints are differentiated between soft and hard [Solnon et al. 2008]. Hard constraints must be satisfied to avoid a critical stoppage of the production and soft constraints can be violated for a certain cost. The introduction of task sharing might enable some of these hard constraints so be changed to soft constraints.

### 2.3.1 Research gap and research questions

**Research gap**: The main focus of this thesis is the analysis and evaluation of task sharing and sequencing in an unpaced mixed model flow line at the operational level using a deterministic model. In general terms, literature has focused on single-model lines with two work stations and authors have used simulation as a research approach with stochastic processing times and have used state-dependent task sharing decision rules. This thesis studies a mixed- model line with five consecutive workstations to show the full effect of task sharing. Also, attention is paid to finding the best sequence of jobs entering the flow line. Very few research has been conducted combining both task sharing and sequencing and studying the effects of both approaches together to maximize the efficiency of the line. This thesis focuses in the interaction that sequencing and task sharing have between each other.

**Research questions**

1. How can an unpaced flow line with limited buffer capacity be modelled to include task sharing?

Can we reduce the makespan? How much?

How does the quantity of task shared affect the results?

2. How can the sequencing problem be included in an unpaced flow line where task sharing is allowed?

   How much does the sequence given affect the benefits of task sharing?

   Can a good sequence neglect the benefits of task sharing?

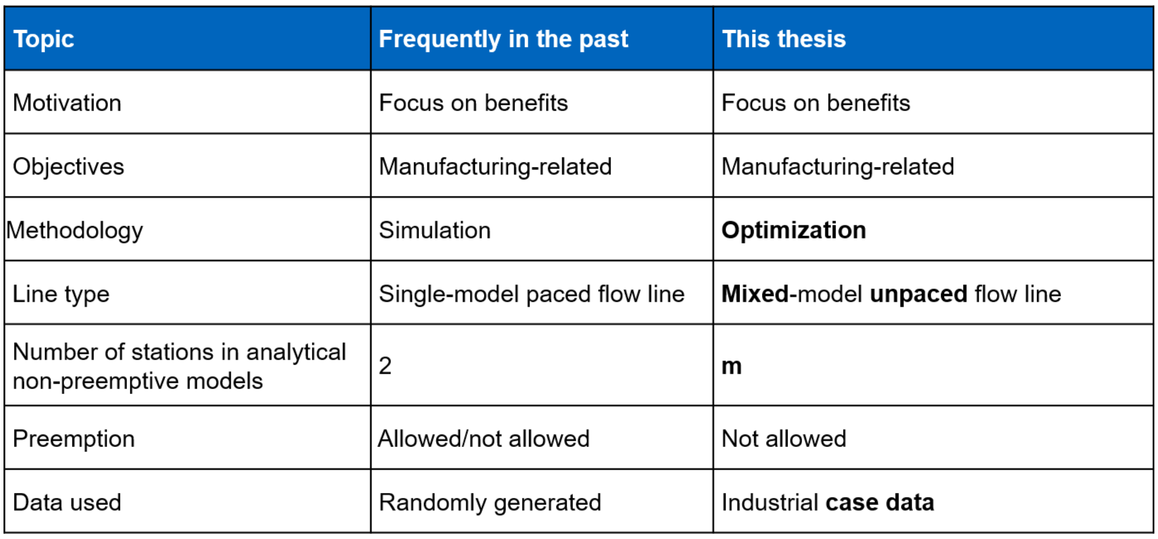
   How does the quantity of task shared affect the results?

3. How does the interaction of task sharing and intelligent sequencing look like?

   How much is the makespan reduced by intelligent sequencing compared with the initial model without task sharing?

   Does the combination of both deliver better results for different sequences? What is the percentual gain?

   Are these benefits ( in case there are any) worth assuming both costs?



| Topic | Frequently in the past | This thesis |
|---|---|---|
| Motivation | Focus on benefits | Focus on benefits |
| Objectives | Manufacturing-related | Manufacturing-related |
| Methodology | Simulation | **Optimization** |
| Line type | Single-model paced flow line | **Mixed**-model **unpaced** flow line |
| Number of stations in analytical non-preemptive models | 2 | **m** |
| Preemption | Allowed/not allowed | Not allowed |
| Data used | Randomly generated | Industrial **case data** |

Figure 2.3: This thesis setting

# 3 Methodology

The methodology in this thesis is presented in the following flow chart:

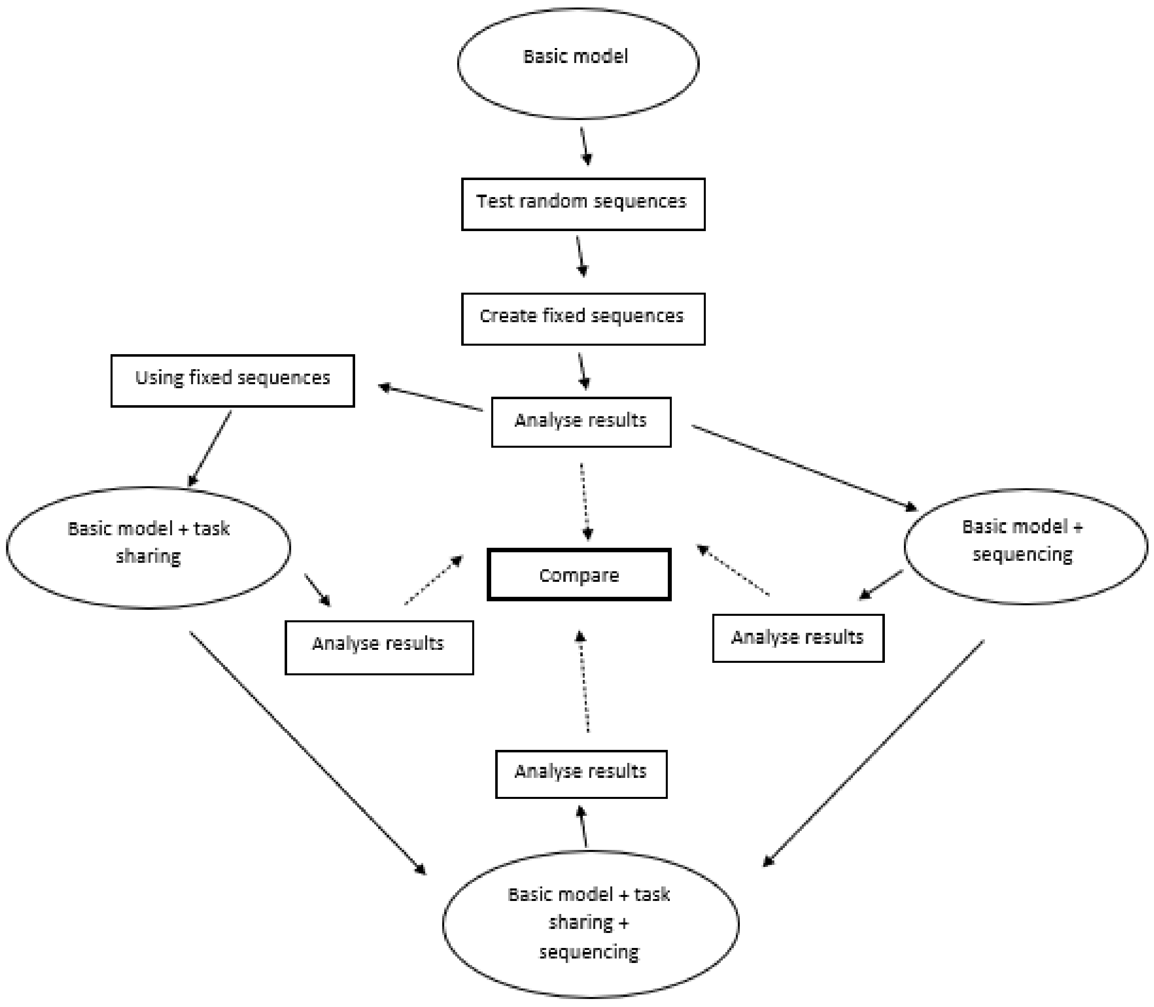


Figure 3.1: Methodology Flow Chart

**Step 1**: Basic model with random sequences.

The first step is to create a generic sequencing model for unpaced flow lines. The model can then be tested with random sequences to see the effect of task sharing in the efficiency of the line for a given objective and it will serve as base case for every comparison with the future models.

**Step 2**: Create fixed sequences.

Once the model has been tested with random sequences to prove its functionality as an example of a case scenario where we can not choose the sequence, a sequencing logic is built. In this new scenario, the line has a resequencing buffer available before the first workstation so that our sequence is not totally dependant on the previous line but we have the possibility to build more efficient sequences by applying a simple and generic logic that will be explained in detail. Creating good sequences will be crucial to understand the effects of task sharing and intelligent sequencing in the following steps. If we would like to compare different scenarios, it would be unfair to use random sequences as we can never be sure if the benefits obtained in our models are a consequence of either task sharing or intelligent sequencing or a consequence of choosing a random sequence that provokes a loss of eficiency in the base case.

**Step 3**: Basic model + task sharing.

We have obtained the first results for our model after the first two steps, building the current base case for the new solutions that this thesis proposes. It is now time to evaluate the effect of task sharing for the created sequences and to analyze the variations in the makespan.

**Step 4**: Basic model + sequencing.

This step will help us understand the difference between using a generic sequencing logic and adding sequencing to your model to find the optimal sequence for your specific conditions. Step 4 represents a crucial moment in this thesis, as it allows us to compare the benefits of task sharing, obtained in Step 3, with the benefits of intelligent sequencing. If we would like to compare this two approaches, it would not be reasonable to directly compare the makespans obtained in both experiments. As we have mentioned before, the sequence chosen plays a great role in this result. However, it is reasonable to compare each of this scenarios with the base case. If we measure the efficiency increase to the base case in both cases separatedly, we are collecting data that may help us take the decision on choosing either to allow task sharing in our system or to focus on building optimal sequences in the resequencing buffer

**Step 5**: Basic model + task sharing + sequencing

The last step explores the scenario where both systems are included. We obtain optimal sequences while task sharing between stations is allowed. This scenario is once again compared to the base case to measure improvement and to be compared with previous steps.

## 3.1 Building a basic model

### 3.1.1 Sets and Parameters

The research approach of this thesis begins with the need for a mathematical model that simplifies the real system that is studied in this work. This simplification is needed because the real system of an unpaced flow line working as a mixed model assembly line is too complex in reality to represent it precisely with all its components and possibilities. The thesis focuses on finding optimal solutions to different scenarios where task sharing and sequencing are used to be able to compare them and understand the benefits of including both solutions in assembly lines. As the focus lies on evaluating the effect of these techniques in simplified environments once they reach optimality and keeping the validation effort low, an analytical/deterministic model is chosen. Then, an optimization programming language is chosen to implement the model.

Before going into the mathematical formulation, the Sets and parameters that will be part of the different models are presented in the following figure.

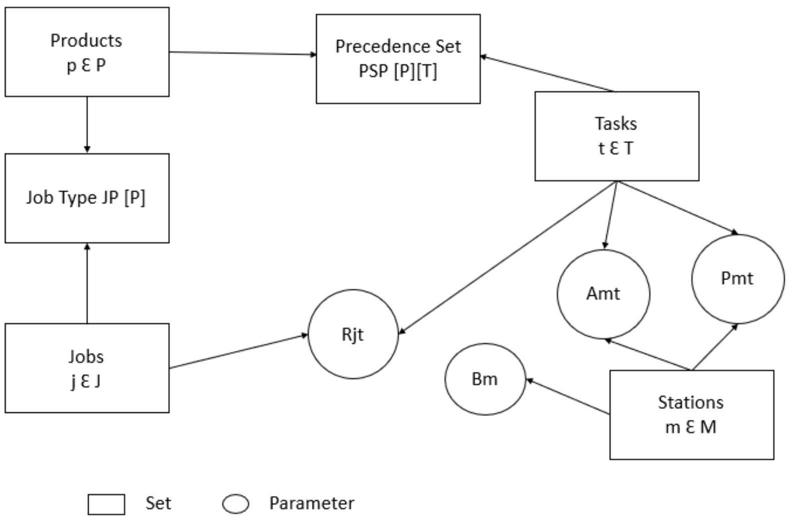


Figure 3.2: Sets and parameters

## Sets

$J$ - Jobs (1,2,..J) Every item that enters the assembly line to be processed is considered a job. This labelling is independent of the type of product entering the line. In other words, the next job entering the line after job j will always be j+1.

$T$ - Tasks (1,2,..T) Every job receives different tasks to be processed throughout the station, tasks are classified in this problem as fixed tasks or flexible tasks depending on their availability. If task t is a fixed or a flexible task this is represented by the values of its parameter Pjt. Once a task is started in one station, the worker must finish this task before passing the job to the next station.

$M$ - Stations (1,2,..M) The flow line is composed by M consecutive stations, where every station contains one worker that can work in only one job at the same time and every job goes through every station (Even if the job does not receive any task t in station m). In that case, the job is considered to go through the station at infinite velocity. A worker can only start a new job once he has sent the previous job to the buffer located right after his station.

$P$ - Products (1,2,..P) In a mixed model assembly lines, different types of products are processed. While all jobs entering the station receive similar tasks, depending on the type of product they are the job will receive some particular tasks associated with its product type.

$JP_{\text{p}}$ - Jobs Set of type Product p in P This set represents the connection between every job that enters the station and its product type. This is done by studying for every type of product, which of the incoming jobs are of his product type. For example, if jobs two and four are of type product one $JP_1$ (2,4)

$PSP_{\text{pt}}$ - Precedence Set for all t in T for Product p in P This set takes into account the precedence of tasks concerning the type of product that is travelling the flow line. For example, if task one needs to be performed before tasks three and five for product one: $PSP_{1\,1}$ (3,5). Tasks three and five are then direct successors of task 1

**Parameters**

$A_{\text{mt}}$ = 1 if task t can be done at station m
0 otherwise

Not every task can be done at every station as it makes no sense to provide every station with all the tools and materials that every type of task requires. The parameter Amt gives information about which tasks could be performed in which stations.

$P_{\text{mt}}$ - Processing time of task t for station m

Every task needs time to be processed. Some of the tasks will be labelled as fixed and others as flexible, but every task requires a processing time.

$R_{\text{jt}}$ = 1 if task t is required for job j 0 otherwise

Depending on the type of product that job j is, the required tasks for this job may vary. This parameter is obtained after a pre-processing of the manufacturer's data, where information of the tasks required for every product p is received, i.e. Rpt if task t is required for product p.

$B_{\text{m}}$ - Buffer Size in Station m

There is a buffer located after every station m. However, there is no buffer after the last station as the finished products are then moved to another station. Therefore, the buffer size in the last station has an unlimited capacity. To reduce computational effort, the last buffer can be considered of a size equal to the total number of jobs that are processed in this line for a particular case study.

### 3.1.2 Modelling an unpaced flow line

The first step in our model will be to model a traditional unpaced flow line. In this initial model, every task is fixed to one station and the sequence is given as a parameter. To model this line, two decision variables are included:

$S_{\text{mj}}$ = starting of job j at station m

This decision variable will measure the time at which every job starts at each station. It is important to remember here that in this flow line every job will necessarily go through every station. In the implementation, it is considered by default that $S_{1\,1}$=0, there will be no need to impose this constraint.

$C_{\text{mj}}$ = Completion time of job j at station m

This is the time when each one of the jobs is completed at the different machines and is moved to the buffer. Note that the completion time of job j at station m is not necessarily equal to the starting time of job j+1 at station m because the job may need to wait in the buffer until the next machine is available.

Next, the objective of the model needs to be decided. In this case, we are aiming to optimize the assembly flow line studied. In other words, we want the jobs going through the line to be assembled as fast as possible, this can be understood as a maximization of the throughput or minimization of the makespan. The makespan is the time the last job leaves the system. Therefore, we would like to minimize the completion time of the last job in the last machine.

In our model:

Min $C_{\text{max}}$ = Min $C_{\text{MJ}}$

Finally, the constraints that model this system are enumerated. Some of them will sound familiar as a model for this type of system has been exposed in section 2.2.1 of the literature review. Therefore, similarities between these constraints and the once found in the work of Nowicki (1999) are naturally given.

$$S_{\text{mj}} \geq C_{\text{m j-1}} \qquad \forall m \in M, j > 1 \in J$$

A job can only be started at one machine if the previous job in the sequence has already been processed. Job j can never overtake job j-1 in a flow line.

$$S_{\text{mj}} \geq C_{\text{m-1 j}} \qquad \forall m > 1 \in M, j \in J$$

A job can only start at one machine after it has been completed in the previous machine. As mentioned above, every job goes consecutively through every station.

$$S_{\text{mj}} \geq S_{\text{m+1 j-1-B}_{\text{m}}} \qquad \forall m \in M(\neq M), j > 1 + B_{\text{m}} \in J$$

A job can only start at one machine if this machine is available. We know that in this system, there is only one worker at every station working on a particular job. This worker needs to complete a job to take a new one from the upstream buffer, as exposed in constraint 1. However, this is not a sufficient condition to start a new job in the same station. As we have mentioned before, the worker moves the job to the downstream buffer after finishing with its processing. Therefore, there must be enough space in the buffer to the worker to do so or the job will remain to wait in the station and the station will be blocked and no new job will be able to be started. We know that if the job (j-1-Bm) has started in the next machine, there is for sure a spot for the job (j-1) in the buffer of station m and station m will be ready to receive job j ( it will not be blocked anymore). As an example, considering a buffer size of one unit this would mean that job three can not start in machine one until job one has already started in machine two. Otherwise, job one would still be in the buffer after the first machine and job two would be in station one waiting to be moved to the buffer while blocking the station.

To sum up, the starting time of every job at every station will be

$S_{\text{mj}} = max(C_{\text{m j-1}}, C_{\text{m-1 j}}, S_{\text{m+1 j-1-B}_\text{m}})$

The other decision variable described was the completion time

$C_{\text{mj}} = S_{\text{mj}} + \sum_t^T P_{\text{mt}} * R_{\text{jt}} \qquad \forall m \in M, j \in J$

The completion time of every job at every station is equal to the starting time plus the processing time of all the required tasks at this station. Every task in this initial model is a fixed task and the parameter Rjt ensures that only the required tasks for each job are processed.

## 3.2 Create fixed sequences

This section focus on building a sequencing logic that will help us find good sequences to our model. Having good sequences and following the same sequencing logic every time reduces the influence that the sequence has in our objective function. The sequencing logic attempts to find good sequences by focusing only on the different products that come into the line. This keeps the computational effort really low and creates sequences that, despite not being optimal, provide better results than random sequences and are independent of the number of stations, number of tasks or any other specific requirements of the line.

What is the sequencing logic followed in this thesis? As our objective is to minimize the makespan and we are talking about unpaced flow lines, we already know that a reduction in the makespan comes along with avoiding or reducing starving and blocking times. Blocking and starving are also known as line imbalances. These are moments when the flow of products through the line is interrupted and productivity is lost, as we will have some workers waiting downstream to receive a job (starving) or waiting to find a free space in the buffer after finishing with one job to be able to start with the next one (blocking). For this reason, this section searches for sequences with a focus in workload balancing. This means dividing the line in several intervals and focusing on balancing the workload in each interval, as we will explain now in detail by showing the model.

### 3.2.1 Creating a model to balance the workload

#### 3.2.1.1 Sets and Parameters

Following the same procedure as in the previous section, we will start the model by defining any new sets and parameters that are needed for this model.

**Sets**

$N$ - Interval (1,2...N) We are going to evaluate the workload in several intervals, so that the the whole sequence gets balanced.

**Parameters**

$D_{\text{p}}$ - Demand in units for product p.
After choosing one Product mix, we will have a resulting demand for every type of product that will determine the assignment of products to different jobs.
$PRODTIME_{\text{p}}$ - Processing time of product p.
When multiplying, in a preprocessing phase, the Processing time for each task in each station by the Required tasks per each type of product, we obtain the processing times of the task that are performed for each product. By adding them up, we compute the processing time for each product in every station m. The average of these m processing times lead to this parameter, which shows the average processing time per station for every product p.
$w$ - Length of the interval
$AVGW$ - Average workload of interval of size w. The value for the average workload of 1 interval is preprocessed in Excel.

#### 3.2.1.2 Constraints and objective function

Constraints

$\sum_p^P Z_{\text{pj}} = 1 \qquad \forall j \in J$

Only one product can be assigned to every job j.

$\sum_j^J Z_{\text{pj}} = D_{\text{p}} \qquad \forall p \in P$

Each product must fulfill its demand.

$Y_{\text{n}} = \sum_{k>=n}^{k<=n+w-1} \sum_p^P PRODTIME_{\text{p}} * Z_{\text{kp}} \qquad \forall k \in J \forall n \in N$

For every interval, we compute the value of the sum of processing times of every product belonging to the interval.

$Y_{\text{n}} = AVGW + W1_{\text{n}} - W2_{\text{n}} \qquad \forall n \in N$

We compare the value obtained in every interval with the average workload, so that we obtain either a positive or a negative deviation.

Objective

Min $\sum_{n}^{N} W1_{\text{n}} + W2_{\text{n}}$

The objective is to minimize the total deviations to the average workload, so that every interval remains close to the average as we manage to balance the whole line.

#### 3.2.1.3 Model

Let us have a look at the whole model:

Objective

Min $\sum_{n}^{N} W1_{\text{n}} + W2_{\text{n}}$

Decision Variables

$Z_{\text{pj}}$ = 1 if product p is in position of job j
0 Otherwise

$Y_{\text{n}}$ = Sum of processing times in position n

$W1_{\text{n}}$ = negative deviation from the average for position n

$W2_{\text{n}}$ = positive deviation from the average for position n

Sets

$J$ - Jobs (1,2,..J)

$P$ - Products (1,2,..P)

$N$ - Position in the sequence

Parameters

$AVGW$ = Average workload of interval of size w

$D_{\text{p}}$ - Demand in units for product p

$PRODTIME_{\text{p}}$ = Processing time of product p

$w_{\text{m}}$ - Length of the interval

Constraints

$\sum_{p}^{P} Z_{\text{pj}} = 1 \qquad \forall j \in J$

$\sum_{j}^{J} Z_{\text{pj}} = D_{\text{p}} \qquad \forall p \in P$

$Y_{\text{n}} = \sum_{k>=n}^{k<=n+w-1} \sum_{p}^{P} PRODTIME_{\text{p}} * Z_{\text{kp}} \qquad \forall k \in J \forall n \in N$

$Y_{\text{n}} = AVGW + W1_{\text{n}} - W2_{\text{n}} \qquad \forall n \in N$

## 3.3 Adding task sharing to the basic model

It is at this point where task sharing is included in the system by the allowance of flexible tasks. We need a new decision variable:

$X_{\text{mtj}}$ = 1 if job j receives task t at station m 0 otherwise

This variable is responsible for the assignment of flexible tasks to a station to make the best decisions to fulfil the objective.

Also, new constraints need to be written to include this flexibility to the model

$$X_{\text{mtj}} \leq A_{\text{mt}} \qquad \forall m \in M, t \in T, j \in J$$

While the availability of the resources to perform the different tasks was included in the parameter containing the processing times for the previous model, the flexibility that task sharing brings in needs to be constrained by the parameter Amt. This parameter will then give input to the model on which tasks can be shared between what stations and which tasks need to remain fixed. Due to this constraint, the parameter Amt will have a huge influence on the effect of task sharing, as this parameter restricts the flexibility that the system may have. Depending on this parameter the system can allow only upstream task sharing, downstream task sharing, task sharing in both directions, etc. for every station.

$$\sum_{m}^{M} X_{\text{mtj}} = R_{\text{jt}} \qquad \forall t \in T, j \in J$$

Every task in the set is unique. This means that it can only be assigned at most once to a job throughout the different stations. However, if this task is not required it can never be assigned to the job throughout the stations.

$$C_{\text{mj}} = S_{\text{mj}} + \sum_{t}^{T} P_{\text{mt}} * X_{\text{mtj}} \qquad \forall m \in M, j \in J$$

With the addition of task sharing the constraint to compute the completion time is changed. The tasks are not anymore just characterized by their fixed processing time. Now they can either be considered as fixed or flexible and they depend on their assignment done by the decision variable Xmtj and their processing time will only contribute to the completion time of a job if the decision to perform this task in this particular station is done ( the task could also be performed in an adjacent station as it is a flexible task).

$$\sum_{m}^{M} m * X_{\text{m t1 j}} \leq \sum_{m}^{M} m * X_{\text{m t2 j}} \qquad \forall t1 \in T, t2 \in PSP[p][t1], j \in JP[p]$$

This constraint is created to control the precedence of tasks. It ensures that if t2 is a direct successor of t1, it can never be performed before t1 along the process. The constraint is formed for the different job sets because depending on the type of product the precedence set may be different.

To sum up, How can an unpaced flow line with limited buffer capacity be modelled to include task sharing?

Moreover, the model described below works both as a replica for the basic model and as the basic model + task sharing. The decision to solve one model or the other is made by the input data introduced for the parameter $A_{\text{mt}}$. When we only allow tasks to be performed at their original station, we will be solving the basic model. However, when we allow tasks to be solved in more than one station, we are introducing task sharing to the degree we decide to do so. We will explore this in detail in the next chapter (Results).

Objective

Min $C_{\text{max}}$ = Min $C_{\text{MJ}}$

Decision Variables

$S_{\text{mj}}$ = starting of job j at station m

$C_{\text{mj}}$ = Completion time of job j at station m

$X_{\text{mtj}}$ = 1 if job j receives task t at station m 0 otherwise

Sets

$J$ - Jobs (1,2,..J)

$T$ - Tasks (1,2,..T)

$P$ - Products (1,2,..P)

$JP_{\text{p}}$ - Jobs Set of type Product p in P

$PSP_{\text{pt}}$ - Precedence Set for all t in T for Product p in P

Parameters

$A_{\text{mt}}$ = 1 if task t can be done at station m 0 otherwise

$P_{\text{jt}}$ - Processing time of task t for station m

$R_{\text{jt}}$ = 1 if task t is required for job j 0 otherwise

$B_{\text{m}}$ - Buffer Size in Station m

Constraints

$$X_{\text{mtj}} \leq A_{\text{mt}} \qquad \forall m \in M, t \in T, j \in J$$

$$\textstyle\sum_{m}^{M} X_{\text{mtj}} = R_{\text{jt}} \qquad \forall t \in T, j \in J$$

$$S_{\text{mj}} \geq C_{\text{m j-1}} \qquad \forall m \in M, j > 1 \in J$$

$$S_{\text{mj}} \geq C_{\text{m-1 j}} \qquad \forall m > 1 \in M, j \in J$$

$$S_{\text{mj}} \geq S_{\text{m+1 j-1-B}_{\text{m}}} \qquad \forall m \in M(\neq M), j > 1 + B_{\text{m}} \in J$$

$$C_{\text{mj}} = S_{\text{mj}} + \sum_t^T P_{\text{mt}} * X_{\text{mtj}} \qquad \forall m \in M, j \in J$$

$$\sum_m^M m * X_{\text{m t1 j}} \leq \sum_m^M m * X_{\text{m t2 j}} \qquad \forall t1 \in T, t2 \in PSP[p][t1], j \in JP[p]$$

## 3.4 Adding sequencing to the basic model

As mentioned in the introduction to the chapter, this section aims to find the best sequence possible for the incoming jobs. This time, we will directly focus in the basic model to include the sequencing in it so that we obtain the best solution possible to our specific parameters. It is important to remark here that, from an operational point of view, this is a more complicated solution. While the workload balancing solution results generally in a good sequence for most types of lines, the optimal sequence obtained here will not necessarily be an optimal sequence for different types of lines, but just for this one. This means that the resequencing buffer must be right before the line ( if you have enough physical space for this) and we may need to resequence again for following lines. On the other hand, we aim to find better solutions with this model than with the workload balancing example. Let us take a look at the new sets, parameters, decision variables and constraints for this new model, which shares some of the nomenclature with the workload balancing model:

New sets

There are not new sets for this model. However $JP_{\text{p}}$ does not exist anymore. We will assign the type of product to each job position when solving the model.

New parameters

$D_{\text{p}}$ - Demand in units for product p.
This parameter was also needed for the workload balancing model.

$M$ - Big number

$R_{\text{pt}}$ = 1 if task t is required for product p 0 otherwise

We do not know the assignment of tasks to jobs anymore because the Job set is not a parameter now, but part of the solution. However, we still know the nature of each product and which tasks t will be required for product p.

New decision variables

$Z_{\text{pj}}$ = 1 if product p is assigned to job j. 0 otherwise This is the same decision variable we used for the workload balancing model.

New constraints

$\sum_m^M X_{\text{mtj}} = \sum_p^P R_{\text{pt}} * \text{Z}_{\text{pj}} \qquad \forall t \in T, j \in J$

We need to adapt the constraint to the new parameter to have the same effect as we did before. If a product p is assigned to a job j and task t is required for job j, then we can obtain once again the information about which tasks t are required for each job j.

$\sum_m^M m * X_{\text{m t1 j}} \leq \sum_m^M m * X_{\text{m t2 j}} + (1 - \sum_p^P Z_{\text{pj}} * R_{\text{pt1}} * R_{\text{p t2}}) * M \qquad \forall t1 \in T, t2 \in PSP[p][t1], j \in J$

We need to adapt this constraint. As we have deleted the Job Set, we do not want the constraint to be binding for every job, but only for those situations where tasks t1 and t2 are both required. Therefore, we use the Big M formulation.

$\sum_p^P Z_{\text{pj}} = 1 \qquad \forall j \in J$

Only one product can be assigned to every job j. This contraint was not needed before because the sequence was included in the Job Set.

$\sum_j^J Z_{\text{pj}} >= D_{\text{p}} \qquad \forall p \in P$

Each product must fulfill its demand. The demand was included in the Job set in previous models.

Let us have a look at the whole new model

Objective

Min $C_{\text{max}}$ = Min $C_{\text{MJ}}$

Decision Variables

$S_{\text{mj}}$ = starting of job j at station m

$C_{\text{mj}}$ = Completion time of job j at station m

$X_{\text{mtj}}$ = 1 if job j receives task t at station m 0 otherwise

$Z_{\text{pj}}$ = 1 if product p is assigned to job j. 0 otherwise

Sets

$J$ - Jobs (1,2,..J)

$T$ - Tasks (1,2,..T)

$P$ - Products (1,2,..P)

$PSP_{\text{pt}}$ - Precedence Set for all t in T for Product p in P

Parameters

$A_{\text{mt}}$ = 1 if task t can be done at station m 0 otherwise

$P_{\text{jt}}$ - Processing time of task t for station m

$B_{\text{m}}$ - Buffer Size in Station m

$D_{\text{p}}$ - Demand in units for product p.
This parameter was also needed for the workload balancing model.

$M$ - Big number

$R_{\text{pt}}$ = 1 if task t is required for product p 0 otherwise

Constraints

$X_{\text{mtj}} \leq A_{\text{mt}} \qquad \forall m \in M, t \in T, j \in J$

$\sum_m^M X_{\text{mtj}} = \sum_p^P R_{\text{pt}} * \mathbf{Z}_{\text{pj}} \qquad \forall t \in T, j \in J$

$S_{\text{mj}} \geq C_{\text{m j-1}} \qquad \forall m \in M, j > 1 \in J$

$S_{\text{mj}} \geq C_{\text{m-1 j}} \qquad \forall m > 1 \in M, j \in J$

$S_{\text{mj}} \geq S_{\text{m+1 j-1-B}_\text{m}} \qquad \forall m \in M(\neq M), j > 1 + B_{\text{m}} \in J$

$C_{\text{mj}} = S_{\text{mj}} + \sum_t^T P_{\text{mt}} * X_{\text{mtj}} \qquad \forall m \in M, j \in J$

$\sum_m^M m * X_{\text{m t1 j}} \leq \sum_m^M m * X_{\text{m t2 j}} + (1 - \sum_p^P Z_{\text{pj}} * R_{\text{pt1}} * R_{\text{p t2}}) * M \qquad \forall t1 \in T, t2 \in PSP[p][t1], j \in J$

$\sum_p^P Z_{\text{pj}} = 1 \qquad \forall j \in J$

$\sum_j^J Z_{\text{pj}} >= D_{\text{p}} \qquad \forall p \in P$

This model leads to a similar situation as the basic model + task sharing, meaning that we can find two different models in the same structure and the decision remains in the use of the parameter $A_{\text{mt}}$. We have explained before that the previous model would solve both the basic model and the basic model + task sharing when modyfing the parameter. In this case, this new model works both as the basic model + sequencing ( when tasks are only allowed to be performed in the original station) and as the basic model + task sharing + sequencing ( when tasks are allowed to be performed in more than one station).

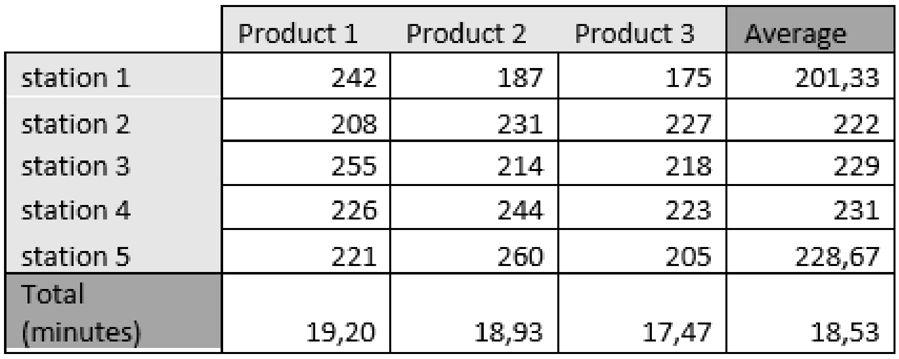
# 4 Results

## 4.1 Case study

There is currently little literature on task sharing that uses real industry data in their work. Theoretical data can lead to unrealistic results. However, this thesis bases its numerical testing in real data from a production system of an engine supplier. The study is based in a unidirectional unpaced mixed-model permutation flow line, consisting of 5 stations near the end of the outbound line. This line looks appropriate for studying task sharing as it is composed of stations that involve many manual tasks, such as screwing, pinning or clipping. These are tasks that can be learned after little training and that do not require complex or really specific materials to be processed. Therefore, it is a good opportunity to figure out the possibility to share tasks between adjacent stations. Each of the 5 stations is filled by exactly one worker, this means that there is a 100 per cent staffing level of workers fixed to their stations. These 5 stations are part of a bigger line but other stations are outside of the scope of this study. For this reason, we will consider that before the first station we have got a source of products (every time station 1 is available, 1 new job will enter the system) and we have got a sink at the end ( working as a buffer with unlimited capacity). In this flow line, three different products can be produced. Products 1 and 2 are very similar in their total processing times and the tasks that they require. However, product 3 has different characteristics in those terms. A table with the diffent tasks and processing times, together with other data, can be found in the Appendix.

62 different tasks are to be processed in this flow line (5.1). Although not every product goes through every task as explained in the previous chapter. Table 5.1 was obtained from the thesis by Hieronymus (2020), who used this same data, and shows the different types of task, processing times in seconds, the product that requires each task and the home station for every task when no task sharing is allowed. It can also be seen in this table that some of the tasks are fixed tasks (marked with F in the table) that cannot turn shareable as they represent core tasks that must be done only in their home station.

Product 1 has the highest workload with 19,2 minutes and product 3 has the lowest workload with 17,47 minutes. About the stations, station 4 has the highest workload with 231 seconds average and station 1 has the lowest with 201,33 seconds. However, the workload in a station is mainly influenced by the product volume mix. For example, station 2 has a lower average workload than station 3, but station 2 has a higher average for product 2. This means that in a product mix where product 2 has the highest volume, we may expect that the workload average for station 2 becomes higher than for station 3. The data from the engine supplier company also provides with difference precedence sets, taken from the thesis by Hieronymus (2020) that can be seen in the Appendix (5.2, 5.3, 5.4) and that will be taken into account, as we have already seen in the models provided in the methodology.



| | Product 1 | Product 2 | Product 3 | Average |
|---|---|---|---|---|
| station 1 | 242 | 187 | 175 | 201,33 |
| station 2 | 208 | 231 | 227 | 222 |
| station 3 | 255 | 214 | 218 | 229 |
| station 4 | 226 | 244 | 223 | 231 |
| station 5 | 221 | 260 | 205 | 228,67 |
| Total (minutes) | 19,20 | 18,93 | 17,47 | 18,53 |

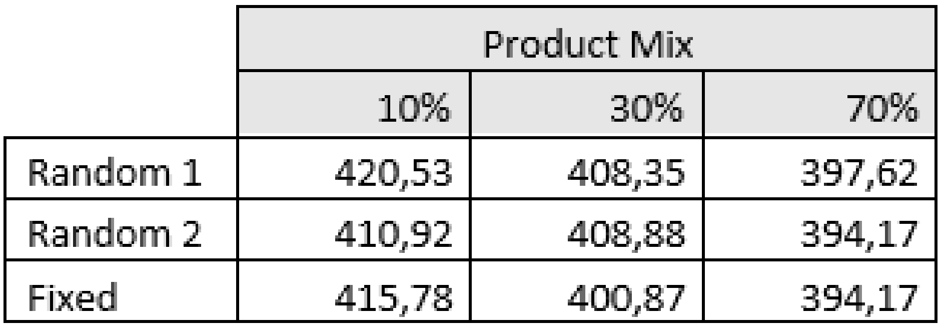
Figure 4.1: Workloads per station and product

## 4.2 Parameters used in the study

We will use sequences consisting of 100 jobs, which approximately simulates one shift at the engine supplier company. There is a buffer for only one job between every two stations, where station 1 is never starving, as there is no station before the line in the model, and station 5 never gets blocked, as there is no station behind. As explained in the previous section, three different types of products will be processed in the line, with products 1 and 2 having similar characteristics and product 3 being less workload demanding. We will conduct a study of the makespan in the system for three different levels of product mixes. The three different levels are 10 per cent, 30 per cent and 70 per cent. The percentage refers to the share of product 3 in the total volume. Product 1 and 2 divide the rest of the share equally (5.5).

## 4.3 Basic model

As explained in the Methodology chapter. We will start showing the results obtained in the thesis by calculating a base case that will help us understand the benefits of task sharing and sequencing in the upcoming sections. For every model in this thesis, Microsoft Excel has been used as a data source, as well as a tool for pre-processing the data to make it accurate for the models. All the models have been programmed and computed in CPLEX Studio. Let us now compute the makespan for the three different product mixes that are evaluated in this thesis. For every mix, the makespan was calculated for two randomly generated sequences in Excel + the fixed sequence obtained with the workload balancing model. All the calculations were done in seconds. However, results will be presented in minutes for a clearer understanding.



| | Product Mix | | |
|---|---|---|---|
| | 10% | 30% | 70% |
| Random 1 | 420,53 | 408,35 | 397,62 |
| Random 2 | 410,92 | 408,88 | 394,17 |
| Fixed | 415,78 | 400,87 | 394,17 |

Figure 4.2: Makespan for different sequences and product mixes

As explained in the Methodology chapter. We can observe in these results, that the workload levelling model provides sequences that are expected to be good, as they help us balance the flow line. However, they are not necessarily optimal sequences. This has been proven for the product mix 10 per cent, where we could find a random sequence that provided a better makespan than the fixed sequence. In a sequence of jobs with very few type 3 products, the model focuses on alternating some of these products with type 1 products, as type 1 products have a higher workload. For this reason, type 2 products are left together in this model. What the workload model does not include is the fact that products of the same type require the same type of tasks. Therefore, if we do not alternate them blocking times can increase and that is exactly what happened in this example. Of course, there are lots of random sequences that provide worse results, such as Random 1, but it was interesting to show here an example where the generic sequencing model provides a higher makespan than some random sequence. The effectiveness of the workload levelling model increases with more balanced product mixes, as shown for 30 per cent, where the fixed sequence provides a better result than most of the random sequences tested. It is important to note here that one length of interval w must be chosen for the model. The optimal length could be understood as the shorter possible (the more intervals we create, the more accurate will the final result be). However, each interval needs space for at least the 3 types of products so that the overall line gets balanced. Therefore, the length of the interval of 5 jobs was chosen for doing these calculations. Similar lengths could have also been studied to see their effect.

Every result obtained in the following models will be compared to the makespan obtained with the fixed sequencing.

## 4.4 Adding task sharing to the basic model

Once we have a reference value for our initial solution. We are going to evaluate the effect of task sharing for the different product mixes. The effect on the makespan of task sharing will depend on the number of shared tasks. For this reason, 8 different situations will be examined, starting from 5 per cent of the total processing time up to 40 per cent. As mentioned in the literature research, it has been studied that the effect of task sharing reaches optimality below the 40 per cent threshold. Out of the 62 tasks that belong to the flow line, most of them could be shared. Therefore we need a selection logic to choose which tasks should be shared for every task sharing degree. It is important to remark that in this thesis we have decided to only allow downstream task sharing. This means that shareable tasks can be performed in either station m or station m+1, but not on station m-1. This decision was made as this is a realistic scenario to work with and it is easier to communicate with the workers and to create standard procedures. Work from other authors with upstream task sharing or total flexibility can be found in the literature.
A selection logic was created and follows this scheme:

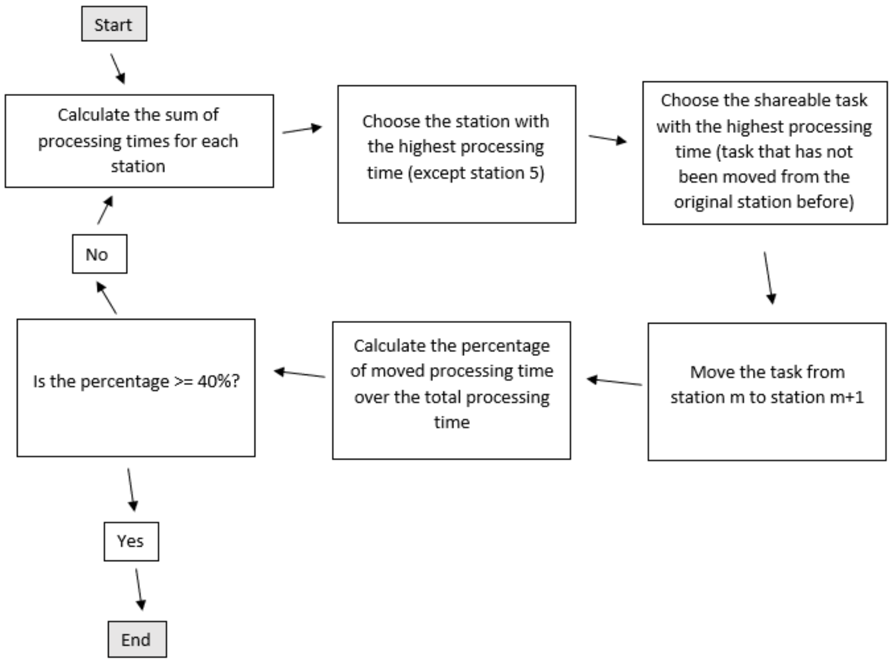

Start
Calculate the sum of processing times for each station
Choose the station with the highest processing time (except station 5)
Choose the shareable task with the highest processing time (task that has not been moved from the original station before)
Move the task from station m to station m+1
Calculate the percentage of moved processing time over the total processing time
Is the percentage >= 40%?
No
Yes
End

Figure 4.3: Selection procedure for choosing shareable tasks

We start by calculating the sum of the processing times for every station so that we can check the potential bottleneck station when every task is processed in its original station. Note that we call it potential because this will depend on the product mix and the required tasks per type of product. Then, we choose the potential bottleneck station and find the shareable task with the highest processing time. This task should never have been moved before. Otherwise, we will end up moving tasks from station m to m+2 or higher and this does not make sense in our flow line. Once the shareable task is chosen, we move it from station m to m+1 and compute the cumulative sum of all the moving processing time to see if we have already reached the 40 per cent objective, if we have not reached it yet, we go back to step 1. While iterating, we will also reach 5, 10, 15, 20, 25, 30 and 35 per cent and we will know at every stage, how many tasks have been moved. These moved tasks will represent the shareable tasks in the original model, the tasks that will be allowed to be performed in either station m or m+1 when calculating the makespan. After 22 iterations, 40 per cent was reached and the process ended.

Now, we will compute the makespan for the different product mixes and degrees of shared tasks, to see the evolution on the makespan if we invest in task sharing. It is important to understand that, as we are choosing the task in every station because of its high processing time, there is little difference in the number of tasks between 5 per cent and 10 per cent, while the difference in the number of tasks between 35 per cent and 40 per cent is much higher and will involve a greater effort to implement this extra 5 per cent.
Let us begin with the most balanced case, with 30 per cent Product 3.

Product Mix 30 per cent. Makespan in minutes and shared tasks in percentage.

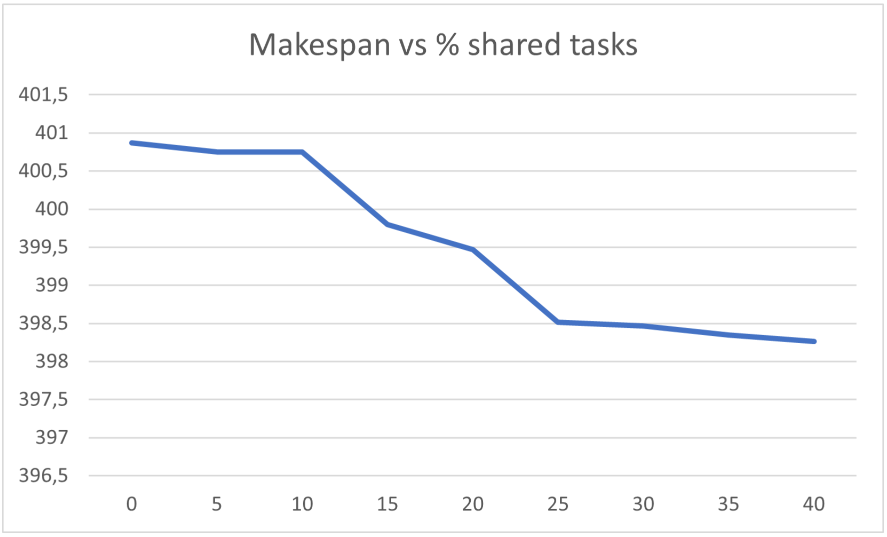


Figure 4.4: Product Mix 30 per cent with task sharing. Makespan in minutes and shared tasks in percentage

In this graphic, the 0 per cent value corresponds to the base case for this product mix (400,87 minutes) and we observe two intervals where the increase in the degree of shared tasks provokes a higher reduction in the makespan. Between 10 and 15 per cent and between 20 and 25 per cent. After 25 per cent, the reduction in the makespan becomes much smaller even when sharing a higher number of tasks. The two optimal degrees of task sharing in this case would be 15 per cent (399,80 minutes; 0,27 per cent improvement) and 25 per cent (398,51 minutes; 0,59 per cent improvement), the first one is achieved with a fewer number of shared tasks, so it is easier and cheaper to implement, while the second one nearly reaches the lowest makespan for this case.

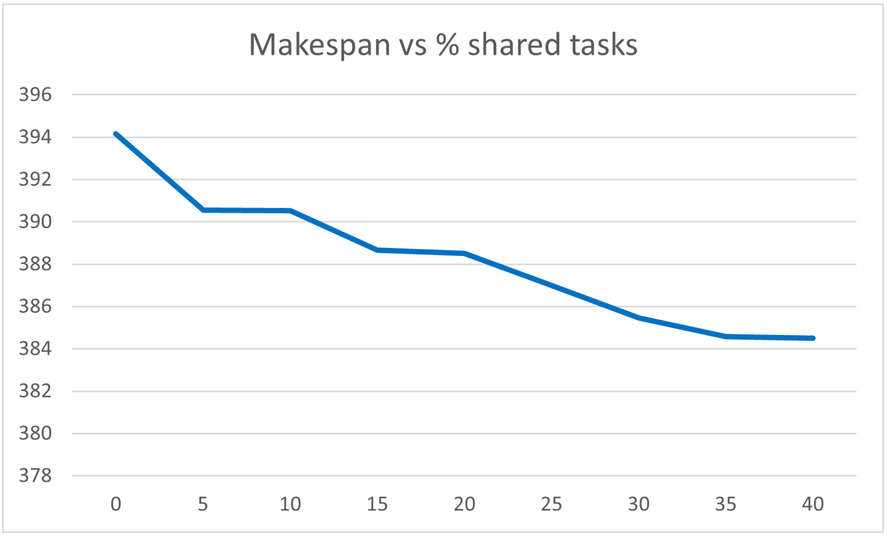


Figure 4.5: Product Mix 70 per cent with task sharing. Makespan in minutes and shared tasks in percentage

For this product mix, a 0,92 per cent improvement in the makespan is observed after sharing just 5 per cent of the tasks. The first tasks shared where selected from station 4, where product 3 requires several tasks to be performed. Therefore, it makes sense that the effect of task sharing is immediate for this product mix. Between 10 and 15 per cent, we also observe a reduction in the makespan, as we did for the 30 per cent product mix, in this case reaching 388,65 minutes ( 1,40 per cent improvement). The highest reduction lies in the 35 per cent (384,58 minutes; 2,43 per cent improvement) for this mix.

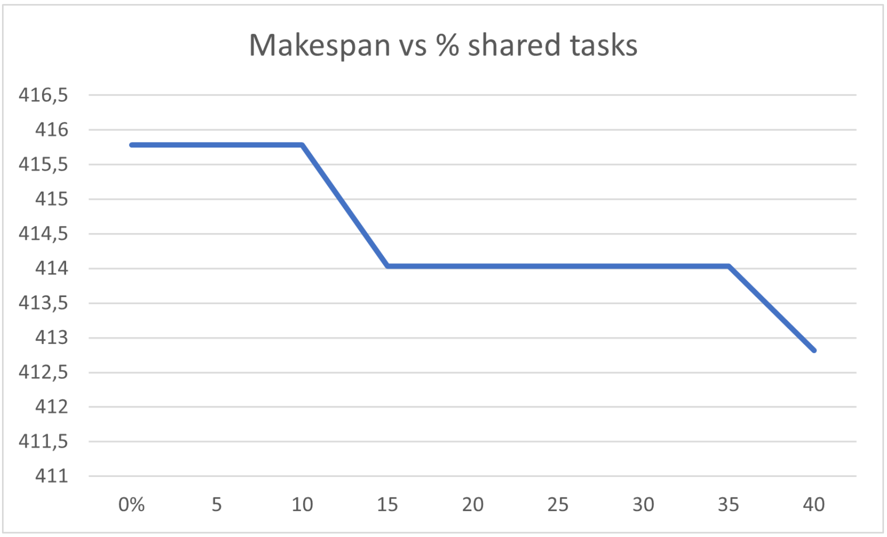


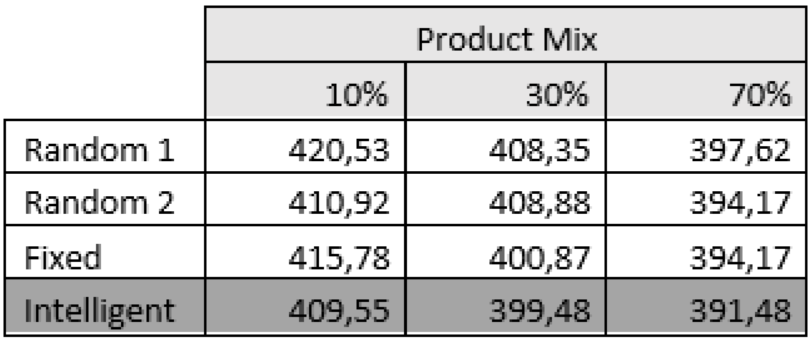
Figure 4.6: Product Mix 10 per cent with task sharing. Makespan in minutes and shared tasks in percentage

For the 10 per cent product mix, we also observe a decrease in the makespan in the interval 10 to 15 per cent, reaching 414,03 minutes (0,42 per cent improvement) .

The next reduction only takes place with a much higher number of shared tasks, between 35 and 40 per cent, reaching 412,82 minutes ( 0,71 per cent improvement).

To sum up, a degree of shared tasks of 15 per cent would be beneficial for this flow line for every product mix. A trade-off cost analysis should be carried to proof if task sharing is worth value for money in this system. As shown in the graphics, the makespan can be reduced up to 1,40 per cent (70 per cent mix) by applying a 15 per cent task sharing degree in the system.
With these results, we verified that lower makespans are obtained when product 3 is predominant. This makes sense as this is the product with the lowest work-load. However, we also showed that, for the same sequencing logic, the selection of shared tasks used in this thesis benefits product mixes with higher values of product type 3. For this reason, we expect to obtain better results on the makespan if we apply workload balancing and this task sharing logic to job sets with a predominance of products type 3.

## 4.5 Adding sequencing to the basic model



| | Product Mix | | |
|---|---|---|---|
| | 10% | 30% | 70% |
| Random 1 | 420,53 | 408,35 | 397,62 |
| Random 2 | 410,92 | 408,88 | 394,17 |
| Fixed | 415,78 | 400,87 | 394,17 |
| Intelligent | 409,55 | 399,48 | 391,48 |

Figure 4.7: Makespan table including intelligent sequencing

By including intelligent sequencing in the model, we observe that a lower makespan is guaranteed for every product mix. Having less impact in the mixes that had a good balance from the workload levelling model, such as 30 per cent or 70 per cent, and a greater impact in the 10 per cent Mix, where the previous model was proven not to be as effective. Moreover, the solution obtained for the 10 per cent product mix is already better than the optimal solution with task sharing for this mix.

## 4.6 Basic model + sequencing + task sharing

We have asked ourselves when introducing this thesis if the combination of both task sharing and intelligent sequencing would ensure the best solution possible.

Let us have a look at the results obtained:

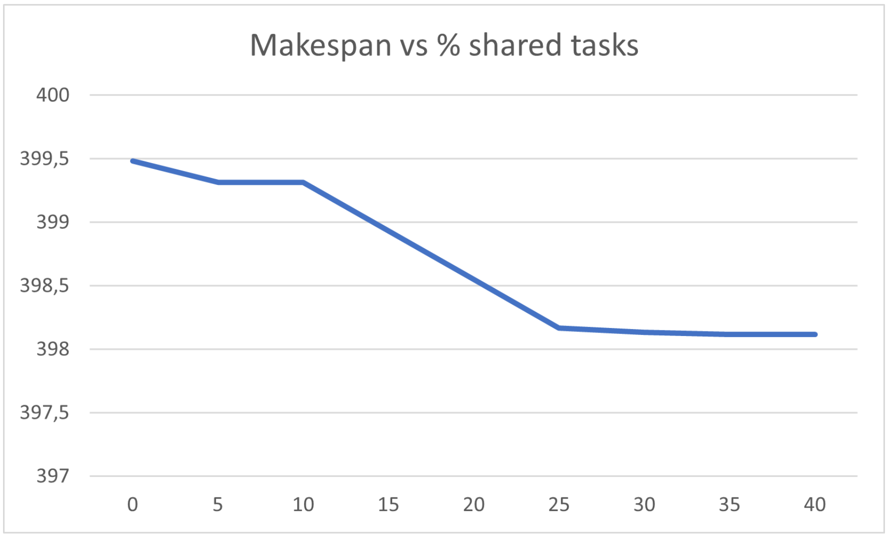


Figure 4.8: Product Mix 30 per cent with sequencing and task sharing. Makespan in minutes and shared tasks in percentage

In this case, the reduction in the makespan starts once again after the 10 per cent degree of shared tasks is reached and continues until reaching 25 per cent, with a 398,17 minutes makespan ( 0,33 per cent improvement). We can observe, how intelligent sequencing has influenced the effect of task sharing. For a 25 per cent degree of shared tasks , the improvement was settled on a 0,59 per cent for the previous Product Mix 30 per cent graphic.

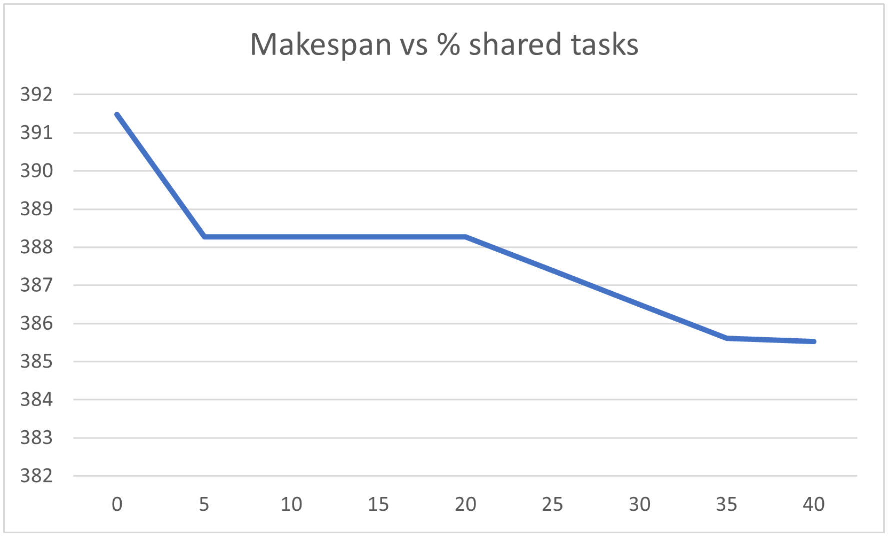


Figure 4.9: Product Mix 70 per cent with sequencing and task sharing. Makespan in minutes and shared tasks in percentage

As explained before, the effect of task sharing is immediate for this product mix. At a 5 per cent sharing level, the makespan is 388,27 minutes (0,82 per cent improvement) We also observe a reduction in the makespan after 35 per cent degree of shared tasks, up to 385,62 minutes ( 1,50 per cent improvement).
We can still observe how we got better results for this mix than for the others. However, the improvement gets once again lowered when using intelligent sequencing.

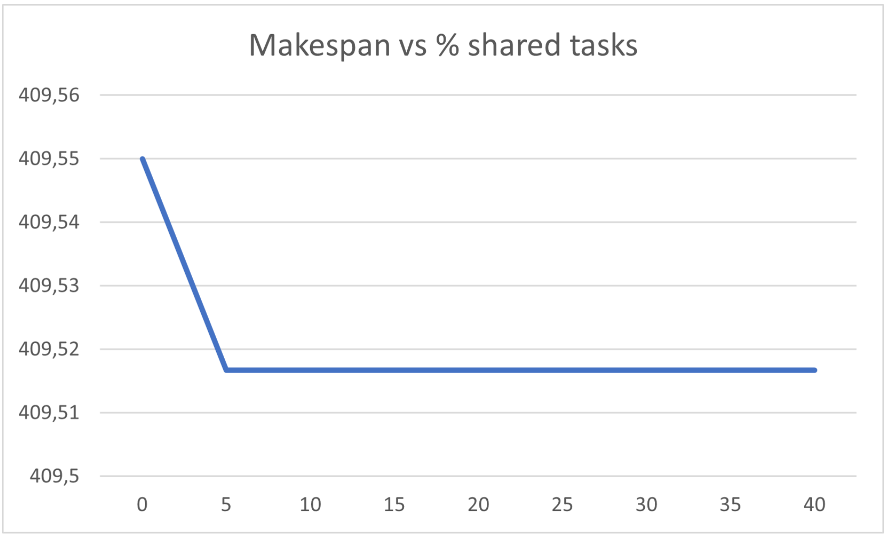


Figure 4.10: Product Mix 10 per cent with sequencing and task sharing. Makespan in minutes and shared tasks in percentage

In this peculiar case, the effect of task sharing goes unnoticed. Even if we can observe a difference in makespan when 5 per cent task sharing is allowed, this is just a matter of seconds and the makespan remains constant afterwards. This is the clearest example of how important a proper sequencing is when minimizing the makespan. As mentiones before, even with no effect from task sharing, the makespan reduces when using intelligent sequencing for this product mix.

# 5 Conclusion

In this thesis, we started our research by modelling an unpaced flow line with limited buffer capacity to determine the different starting and completion times of the jobs at every station to minimize the makespan. This leads us with the initial solution that the system provided before we included task sharing or sequencing.
We have shown how this flow line can be modelled to include task sharing. The results obtained illustrate that even for a good sequence, provided by a workload balancing model created for the purpose, task sharing would have a positive effect on reducing the makespan up to 2,43 per cent with a degree of shared tasks of 40 per cent ( Product mix 70 per cent). Also, it has been proved that task sharing affects even with a low level of shared tasks. For most of the models, a 15 per cent degree of shared tasks is already a level at which task sharing has an impact on the makespan. What is more, with just a 5 per cent degree, tasks sharing starts reducing the makespan for some product mixes. This also happens when we are not allowed to resequence before entering the line and we are provided with a random sequence, although these calculations were not included so that the results offered remained comparable.
After including intelligent sequencing in the model, we showed how the solution got more efficient and results on makespan got generally lower. However, we were able to see here the interaction between both techniques and we understood that the optimal sequence made task sharing less important in every case. Intelligent sequencing was proven to even neglect task sharing in some of the cases.
This opens a new direction of research where an optimal combination between both techniques could be studied to obtain more efficient flow lines, as well as deep diving in the trade-off that derives from the economic investments of applying these techniques in the industry. In this case, results should also be considered at a tactical level to make these balancing decisions.
The models created for this thesis could also be extended or applied for simulations, focusing on a bigger number of jobs or adding real-life constraints that have not been considered in this work. Upstream task sharing could also be included for higher flexibility in the process, together with other objectives such as fairness in the distribution of tasks for the workers, although it has been studied that fairness contradicts the makespan objective and, therefore, it was not included in this work. Also, to be able to adapt task sharing in an industrial application, indirect costs such as the learning process needed by the workers, or the quality defects that sharing could create should be considered in the analysis.

# Bibliography

Abdollahpour, Sana, Javad Rezaeian. 2015. Minimizing makespan for flow shop scheduling problem with intermediate buffers by using hybrid approach of artificial immune system. *Applied Soft Computing* 28 44–56.

Ahn, Hyun-Soo, Rhonda Righter. 2006. Dynamic load balancing with flexible workers. *Advances in Applied Probability* 38(3) 621–642.

Aigbedo, Henry. 2004. Analysis of parts requirements variance for a jit supply chain. *International Journal of Production Research* 42(2) 417–430.

Andradottir, Sigrun, Hayriye Ayhan. 2005. Throughput maximization for tandem lines with two stations and flexible servers. *Operations Research* 53(3) 516–531.

Andradottir, Sigrun, Hayriye Ayhan, Douglas G. Down. 2001. Server assignment policies for maximizing the steady-state throughput of finite queueing systems. *Management Science* 47(10) 1421–1439.

Andradottir, Sigrun, Hayriye Ayhan, Douglas G. Down. 2007. Dynamic assignment of dedicated and flexible servers in tandem lines. *Probability in the Engineering and Informational Sciences* 21(4) 497–538.

Anuar, Rouie, Yossi Bukchin. 2006. Design and operation of dynamic assembly lines using work-sharing. *International Journal of Production Research* 44(18-19) 4043–4065.

Armbruster, Dieter, Esma S. Gel, Junko Murakami. 2007. Bucket brigades with worker learning. *European Journal of Operational Research* 176(1) 264–274.

Askin, Ronald G., Jiaqiong Chen. 2006. Dynamic task assignment for throughput maximization with worksharing. *European Journal of Operational Research* 168(3) 853–869.

Bartholdi, John J., Leonid A. Bunimovich, Donald D. Eisenstein. 1999. Dynamics of two- and three-worker “bucket brigade” production lines. *Operations Research* 47(3) 488–491.

Bautista, J., R. Companys, A. Corominas. 1996. Heuristics and exact algorithms for solving the monden problem. *European Journal of Operational Research* 88(1) 101–113.

Bolat, A. 1997. Stochastic procedures for scheduling minimum job sets on mixed model assembly lines. *Journal of the Operational Research Society* 48(5) 490–501.

Boysen, Nils, Malte Fliedner, Armin Scholl. 2009a. Assembly line balancing: Joint precedence graphs under high product variety. *IIE Transactions* 41(3) 183–193.

Boysen, Nils, Malte Fliedner, Armin Scholl. 2009b. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research* 192(2) 349–373.

Bratcu, A. I., A. Dolgui. 2009. Some new results on the analysis and simulation of bucket brigades (self-balancing production lines). *International Journal of Production Research* 47(2) 369–387.

Brucker, Peter, Silvia Heitmann, Johann Hurink. 2003. Flow-shop problems with intermediate buffers. *OR Spectrum* 25(4) 549–574.

Bultmann, Matthias, Sigrid Knust, Stefan Waldherr. 2018a. Flow shop scheduling with flexible processing times. *OR Spectrum* 40(3) 809–829.

Bultmann, Matthias, Sigrid Knust, Stefan Waldherr. 2018b. Synchronous flow shop scheduling with pliable jobs. *European Journal of Operational Research* 270(3) 943–956.

Burdett, R. L., E. Kozan. 2001. Sequencing and scheduling in flowshops with task redistribution. *Journal of the Operational Research Society* 52(12) 1379–1389.

Buzacott, John A. 2002. The impact of worker differences on production system output. *International Journal of Production Economics* 78(1) 37–44.

Buzacott, John A. 2004. Modelling teams and workgroups in manufacturing. *Annals of Operations Research* 126(1-4) 215–230.

Celano, Giovanni, Antonio Costa, Sergio Fichera, Giovanni Perrone. 2004. Human factor policy testing in the sequencing of manual mixed model assembly lines. *Computers & Operations Research* 31(1) 39–59.

Dar-el, E. M., R. F. Cother. 1975. Assembly line sequencing for model mix. *International Journal of Production Research* 13(5) 463–477.

Ding, F.-Y., H. Sun. 2004. Sequence alteration and restoration related to sequenced parts delivery on an automobile mixed-model assembly line with multiple departments. *International Journal of Production Research* 42(8) 1525–1543.

Dutta, Sujit K., Andrew A. Cunningham. 1975. Sequencing two-machine flowshops with finite intermediate storage. *Management Science* 21(9) 989–996.

Gagné, Caroline, Marc Gravel, Wilson L. Price. 2006. Solving real car sequencing problems with ant colony optimization. *European Journal of Operational Research* 174(3) 1427–1448.

Gel, Esma S., Wallace J. Hopp, Mark P. Van Oyen. 2002. Factors affecting opportunity of worksharing as a dynamic line balancing mechanism. *IIE Transactions* 34(10) 847–863.

Gel, Esma S., Wallace J. Hopp, Mark P. Van Oyen. 2007. Hierarchical cross-training in work-in-process-constrained systems. *IIE Transactions* 39(2) 125–143.

Gultekin, Hakan. 2012. Scheduling in flowshops with flexible operations: Throughput optimization and benefits of flexibility. *International Journal of Production Economics* 140(2) 900–911.

Gupta, Jatinder N.D., Christos P. Koulamas, George J. Kyparisis, Chris N. Potts, Vitaly A. Strusevich. 2004. Scheduling three-operation jobs in a two-machine flow shop to minimize makespan. *Annals of Operations Research* 129(1-4) 171–185.

Hopp, Wallace J., Mark P. Van Oyen. 2004. Agile workforce evaluation: a framework for cross-training and coordination. *IIE Transactions* 36(10) 919–940.

Hsieh, Y.-C., P.-S. You, C.-D. Liou. 2009. A note of using effective immune based approach for the flow shop scheduling with buffers. *Applied Mathematics and Computation* 215(5) 1984–1989.

Inman, Robert R. 2003. Asrs sizing for recreating automotive assembly sequences. *International Journal of Production Research* 41(5) 847–863.

Kubiak, Wieslaw. 1993. Minimizing variation of production rates in just-in-time systems: A survey. *European Journal of Operational Research* 66(3) 259–271.

Leisten, Rainer. 1990. Flowshop sequencing problems with limited buffer storage. *International Journal of Production Research* 28(11) 2085–2100.

Lin, Bertrand M. T., F. J. Hwang, Jatinder N. D. Gupta. 2017. Two-machine flowshop scheduling with three-operation jobs subject to a fixed job sequence. *Journal of Scheduling* 20(3) 293–302.

Liu, Bo, Ling Wang, Yi-hui Jin, De-xian Huang. 2006. An effective pso-based memetic algorithm for tsp. De-Shuang Huang, Kang Li, George William Irwin, eds., *Intelligent Computing in Signal Processing and Pattern Recognition*, *Lecture Notes in Control and Information Sciences*, vol. 345. Springer Berlin Heidelberg, 1151–1156.

Meyr, Herbert. 2004. Supply chain planning in the german automotive industry. *OR Spectrum* 26(4).

Nawaz, Muhammad, E. Emory Enscore, Inyong Ham. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* 11(1) 91–95.

Nick T. Thomopoulos. 1967. Line balancing-sequencing for mixed-model assembly. *Management Science* 14(2) 59–75.

Norman, Bryan A. 1999. Scheduling flowshops with finite buffers and sequence-dependent setup times. *Computers & Industrial Engineering* 36(1) 163–177.

Nowicki, Eugeniusz. 1999. The permutation flow shop with buffers: A tabu search approach. *European Journal of Operational Research* 116(1) 205–219.

Pan, Quan-Ke, Ling Wang, Liang Gao. 2011. A chaotic harmony search algorithm for the flow shop scheduling problem with limited buffers. *Applied Soft Computing* 11(8) 5270–5280.

Papadimitriou, Christos H., Paris C. Kanellakis. 1980. Flowshop scheduling with limited temporary storage. *Journal of the ACM (JACM)* 27(3) 533–549.

Parrello, Bruce D., Waldo C. Kabat, L. Wos. 1986. Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem. *Journal of Automated Reasoning* 2(1).

Pil, Frits K., Matthias Holweg. 2004. Linking product variety to order-fulfillment strategies. *Interfaces* 34(5) 394–403.

Pinedo, Michael. 2010 *Scheduling: Theory, algorithms, and systems*. Fifth edition

Pratama, Aditya Tirta, Katsuhiko Takahashi, Katsumi Morikawa, Keisuke Nagasawa, Daisuke Hirotani. 2018. Integration of bucket brigades and worker collaboration on a production line with discrete workstations. *Industrial Engineering & Management Systems* 17(3) 514–530.

Qian, Bin, Ling Wang, De-xian Huang, Wan-liang Wang, Xiong Wang. 2009. An effective hybrid de-based algorithm for multi-objective flow shop scheduling with limited buffers. *Computers & Operations Research* 36(1) 209–233.

Röder, Axel, Bernd Tibken. 2006. A methodology for modeling inter-company supply chains and for evaluating a method of integrated product and process documentation. *European Journal of Operational Research* 169(3) 1010–1029.

Rossi, Andrea, Michele Lanzetta. 2013. Scheduling flow lines with buffers by ant colony digraph. *Expert Systems with Applications* 40(9) 3328–3340.

Scholl, Armin. 1999. *Balancing and sequencing of assembly lines*. 2nd ed. Contributions to management science, Physica-Verlag, Heidelberg and New York.

Sennott, Linn I., Mark P. Van Oyen, Seyed M.R. Iravani. 2006. Optimal dynamic assignment of a flexible worker on an open production line with specialists. *European Journal of Operational Research* 170(2) 541–566.

Smith, Kate, M. Palaniswami, M. Krishnamoorthy. 1996. Traditional heuristic versus hopfield neural network approaches to a car sequencing problem. *European Journal of Operational Research* 93(2) 300–316.

Smutnicki, Czesław. 1998. A two-machine permutation flow shop scheduling problem with buffers. *OR Spectrum* 20(4) 229–235.

Uruk, Zeynep, Hakan Gultekin, M. Selim Akturk. 2013. Two-machine flowshop scheduling with flexible operations and controllable processing times. *Computers & Operations Research* 40(2) 639–653.

van Zante-de Fokkert, Jannet I., Ton G. de Kok. 1997. The mixed and multi model line balancing problem: a comparison. *European Journal of Operational Research* 100(3) 399–412.

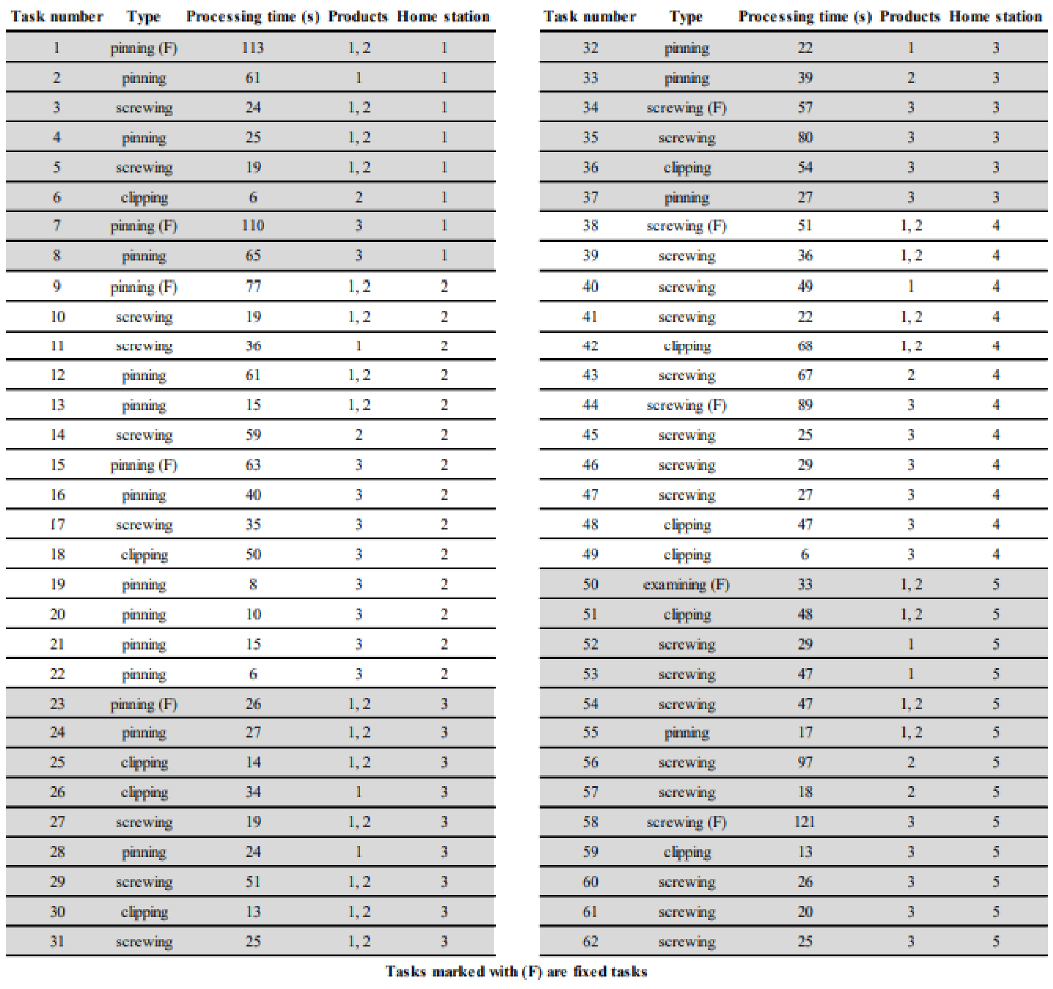
Wang, Ling, Liang Zhang, Da-Zhong Zheng. 2006. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research* 33(10) 2960–2971.

Wild, Ray. 1972. *Mass-production management: The design and operation of production flow-line systems*. John Wiley & Sons, London and New York.

Xu, J., X. Xu, S. Q. Xie. 2011. Recent developments in dual resource constrained (drc) system research. *European Journal of Operational Research* 215(2) 309–318.

Zavadlav, Emil, John O. McClain, L. Joseph Thomas. 1996. Self-buffering, self-balancing, self-flushing production lines. *Management Science* 42(8) 1151–1164.

## Appendix



| Task number | Type | Processing time (s) | Products | Home station |
|---|---|---|---|---|
| 1 | pinning (F) | 113 | 1, 2 | 1 |
| 2 | pinning | 61 | 1 | 1 |
| 3 | screwing | 24 | 1, 2 | 1 |
| 4 | pinning | 25 | 1, 2 | 1 |
| 5 | screwing | 19 | 1, 2 | 1 |
| 6 | clipping | 6 | 2 | 1 |
| 7 | pinning (F) | 110 | 3 | 1 |
| 8 | pinning | 65 | 3 | 1 |
| 9 | pinning (F) | 77 | 1, 2 | 2 |
| 10 | screwing | 19 | 1, 2 | 2 |
| 11 | screwing | 36 | 1 | 2 |
| 12 | pinning | 61 | 1, 2 | 2 |
| 13 | pinning | 15 | 1, 2 | 2 |
| 14 | screwing | 59 | 2 | 2 |
| 15 | pinning (F) | 63 | 3 | 2 |
| 16 | pinning | 40 | 3 | 2 |
| 17 | screwing | 35 | 3 | 2 |
| 18 | clipping | 50 | 3 | 2 |
| 19 | pinning | 8 | 3 | 2 |
| 20 | pinning | 10 | 3 | 2 |
| 21 | pinning | 15 | 3 | 2 |
| 22 | pinning | 6 | 3 | 2 |
| 23 | pinning (F) | 26 | 1, 2 | 3 |
| 24 | pinning | 27 | 1, 2 | 3 |
| 25 | clipping | 14 | 1, 2 | 3 |
| 26 | clipping | 34 | 1 | 3 |
| 27 | screwing | 19 | 1, 2 | 3 |
| 28 | pinning | 24 | 1 | 3 |
| 29 | screwing | 51 | 1, 2 | 3 |
| 30 | clipping | 13 | 1, 2 | 3 |
| 31 | screwing | 25 | 1, 2 | 3 |
| 32 | pinning | 22 | 1 | 3 |
| 33 | pinning | 39 | 2 | 3 |
| 34 | screwing (F) | 57 | 3 | 3 |
| 35 | screwing | 80 | 3 | 3 |
| 36 | clipping | 54 | 3 | 3 |
| 37 | pinning | 27 | 3 | 3 |
| 38 | screwing (F) | 51 | 1, 2 | 4 |
| 39 | screwing | 36 | 1, 2 | 4 |
| 40 | screwing | 49 | 1 | 4 |
| 41 | screwing | 22 | 1, 2 | 4 |
| 42 | clipping | 68 | 1, 2 | 4 |
| 43 | screwing | 67 | 2 | 4 |
| 44 | screwing (F) | 89 | 3 | 4 |
| 45 | screwing | 25 | 3 | 4 |
| 46 | screwing | 29 | 3 | 4 |
| 47 | screwing | 27 | 3 | 4 |
| 48 | clipping | 47 | 3 | 4 |
| 49 | clipping | 6 | 3 | 4 |
| 50 | examining (F) | 33 | 1, 2 | 5 |
| 51 | clipping | 48 | 1, 2 | 5 |
| 52 | screwing | 29 | 1 | 5 |
| 53 | screwing | 47 | 1 | 5 |
| 54 | screwing | 47 | 1, 2 | 5 |
| 55 | pinning | 17 | 1, 2 | 5 |
| 56 | screwing | 97 | 2 | 5 |
| 57 | screwing | 18 | 2 | 5 |
| 58 | screwing (F) | 121 | 3 | 5 |
| 59 | clipping | 13 | 3 | 5 |
| 60 | screwing | 26 | 3 | 5 |
| 61 | screwing | 20 | 3 | 5 |
| 62 | screwing | 25 | 3 | 5 |

**Tasks marked with (F) are fixed tasks**

Figure 5.1: Overview of tasks and its characteristics

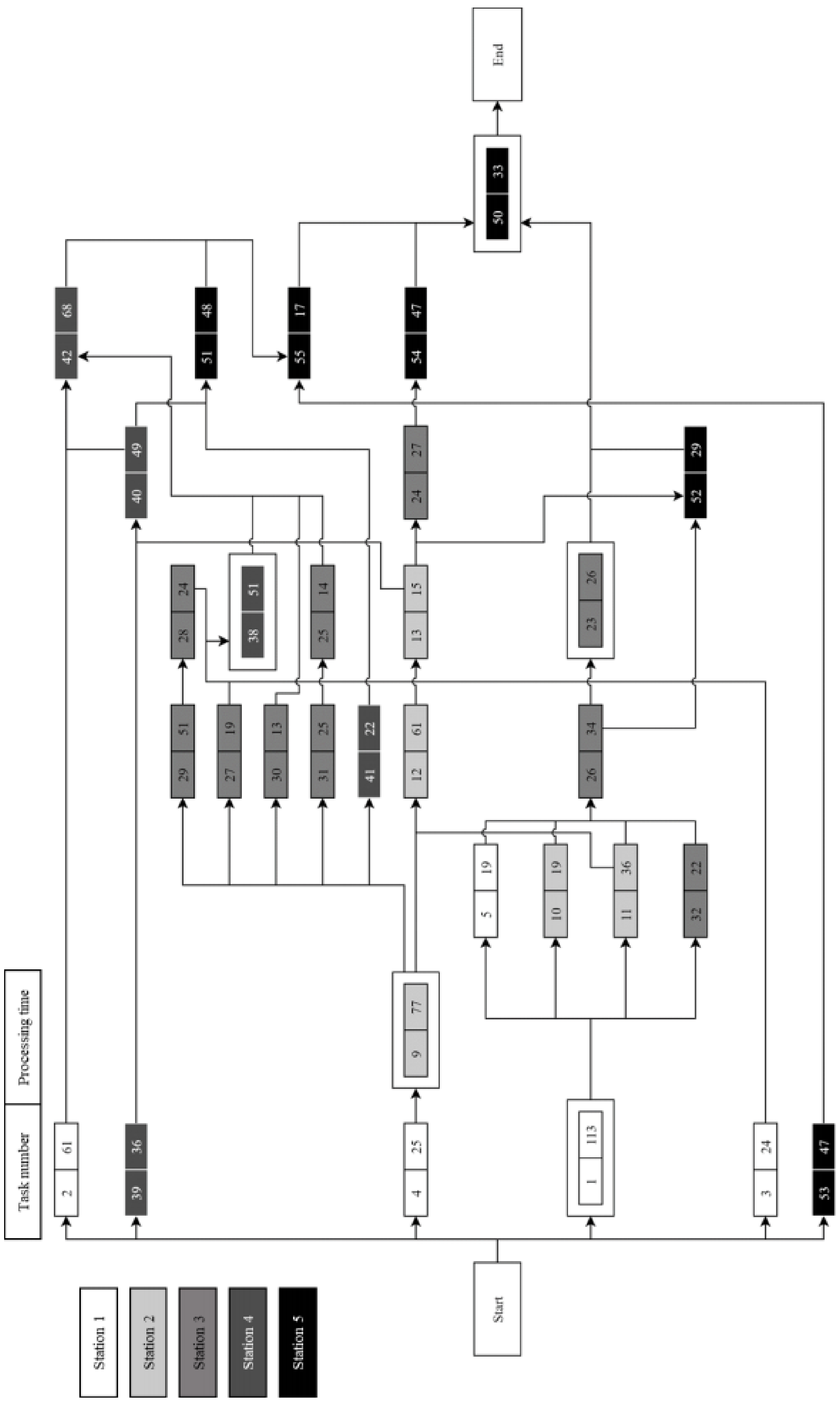


Figure 5.2: Precedence graph for product 1

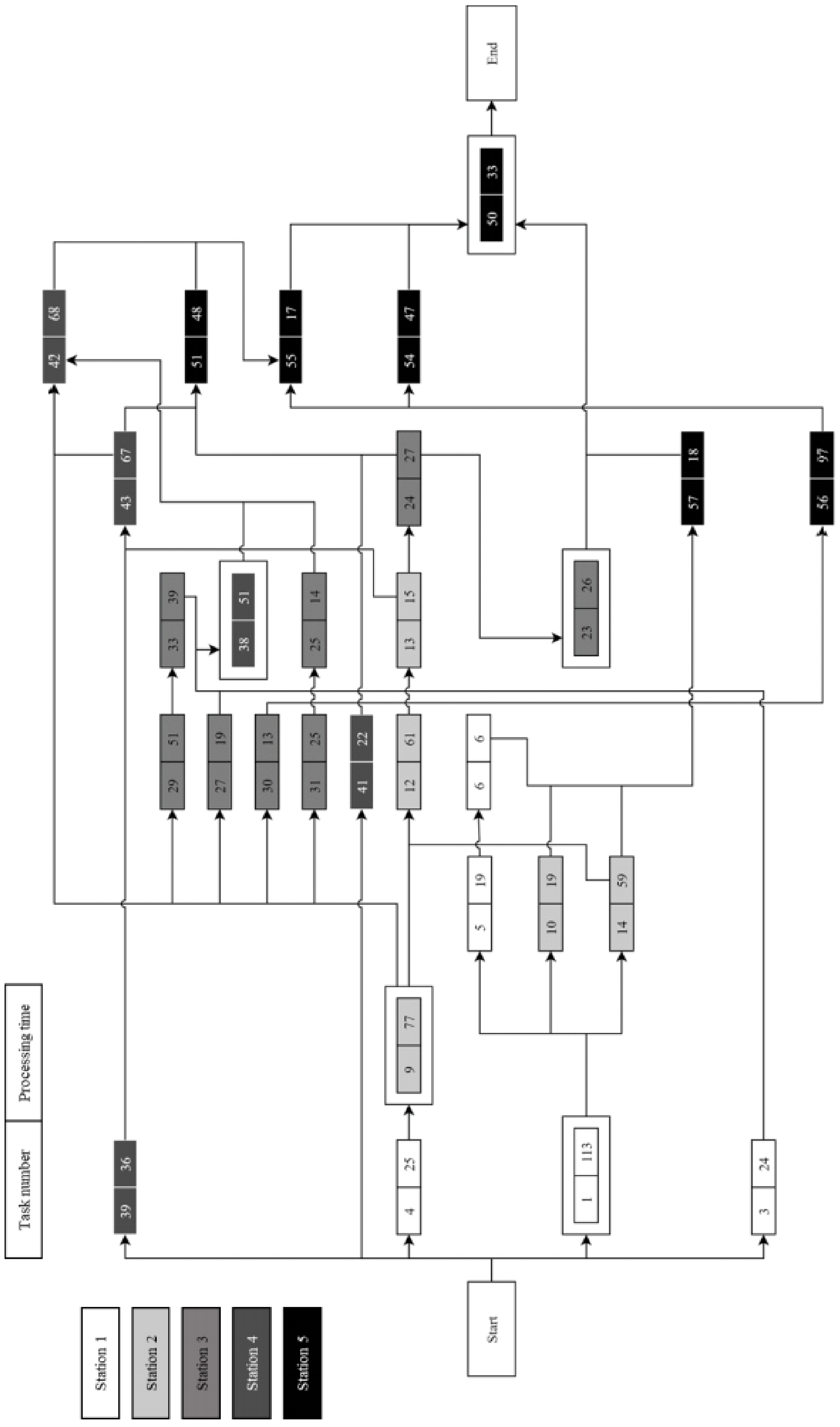


Figure 5.3: Precedence graph for product 2

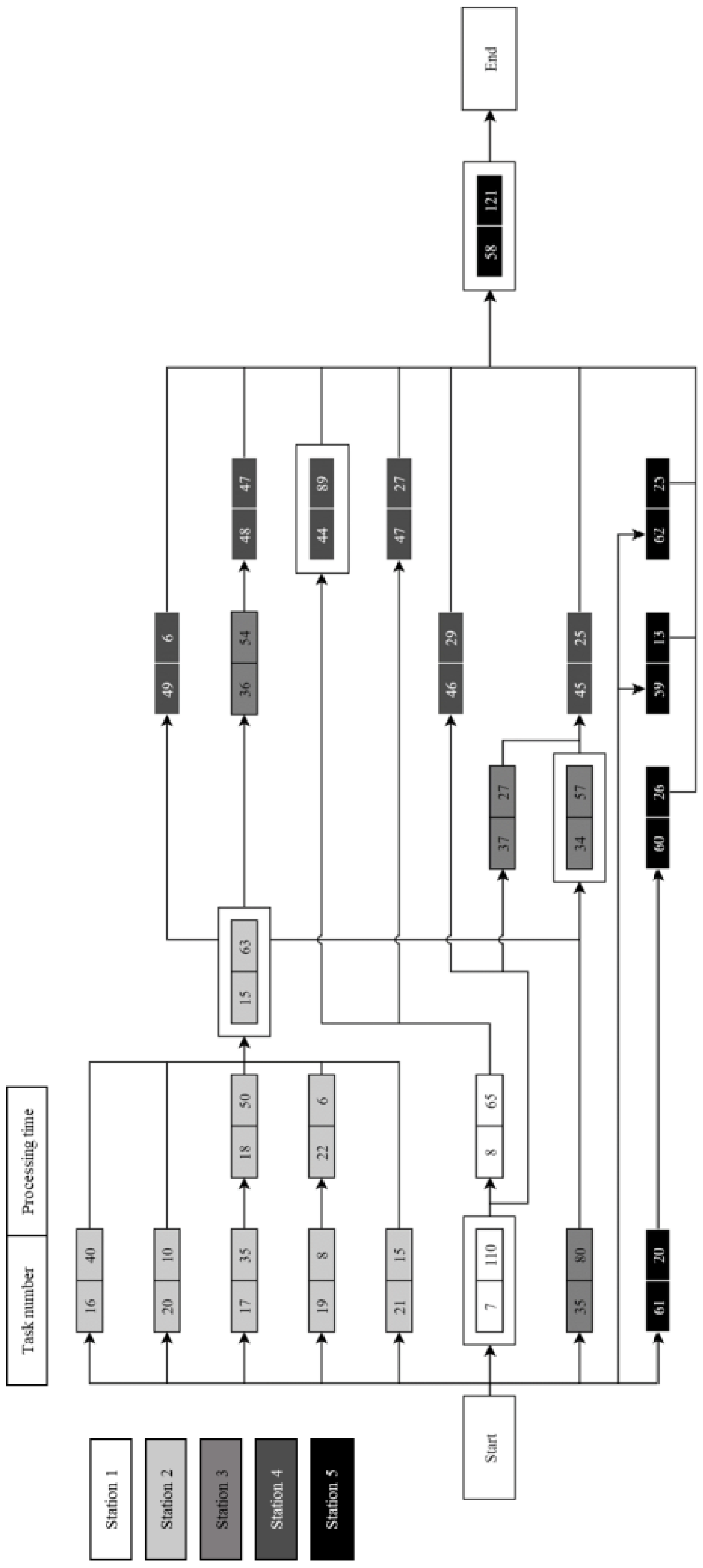


Figure 5.4: Precedence graph for product 3

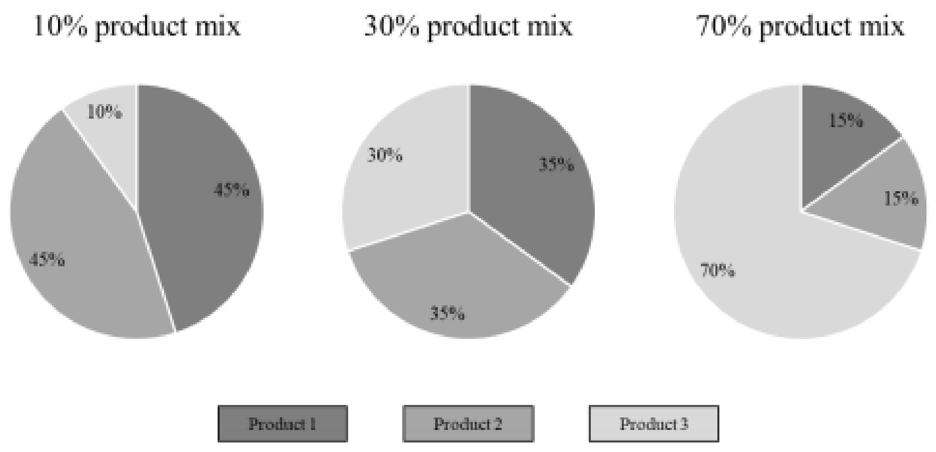


Figure 5.5: Graphical representation of the 3 different product mixes

## Declaration of authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here.

This paper was not previously presented to another examination board and has not been published.

Madrid, 1. October 2020

Javier Dolz Cifre J. Dolz