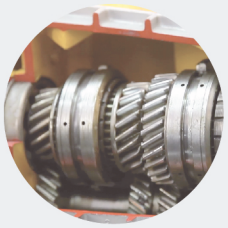
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL






Desenvolvimento e Implementação de uma Ferramenta de Balanceamento de Linhas de Produção para Processos de Montagem e Injeção

ANDRÉ CASTELO DE JESUS MONTEIRO
outubro de 2024

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

DEVELOPMENT AND IMPLEMENTATION OF A PRODUCTION LINE BALANCING DECISION SUPPORT SYSTEM FOR ASSEMBLY AND INJECTION MOLDING PROCESSES

André Castelo de Jesus Monteiro

Dissertation for the attainment of a Master's Degree in Industrial Engineering and Management, Specialization in Industrial Engineering and Management.

Supervisor: Professor Manuel Pereira Lopes

Committee:

Chair:

Professora Alzira Maria Teixeira da Mota, Instituto Superior de Engenharia do Porto

Members:

Professor Glauco Garcia Martins Pereira da Silva, Universidade Federal de Santa Catarina, Brasil

Porto, October 2024

Resumo

A Simoldes Plásticos, uma empresa portuguesa focada na injeção de componentes e peças plásticas para a indústria automóvel, procurou melhorar a capacidade de resposta do seu departamento de processo na adaptação às condições de produção que se encontram em constante mudança, desenvolvendo sistema de apoio à decisão para o problema de balanceamento das linhas de montagem (*ALB DSS*).

Inicialmente, o problema foi estudado de forma aprofundada para identificar as variáveis-chave, as restrições dos sistemas, os métodos para quantificar e atingir os objetivos pretendidos, e os resultados pretendidos. Adicionalmente, foram identificadas várias tipologias de problemas para além do balanceamento individual de uma única linha, como linhas de montagem paralelas com trabalhadores partilhados. Uma formulação matemática foi então desenvolvida como base para um modelo de programação linear capaz de produzir soluções viáveis. Um modelo de menor dimensão e especializado foi também criado para otimizar os sistemas de montagem com múltiplas linhas paralelas.

De forma a criar um programa eficaz e capaz de auxiliar na tomada de decisão, bem como de se ajustar a várias tipologias de linha com diferentes objetivos de balanceamento, foram desenvolvidas cinco funções objetivo para o modelo de programação linear. Estas opções dividem-se entre o balanceamento de linhas individuais e o de linhas paralelas. No caso das linhas individuais, o sistema de suporte à decisão consegue saturar as linhas no início, concentrando as tarefas nos primeiros trabalhadores da mesma; saturar as linhas no final, atribuindo as tarefas aos últimos trabalhadores e libertando tempo de ciclo nos primeiros; ou equilibrar os tempos de ciclo, distribuindo de forma uniforme o tempo de ciclo entre todos os trabalhadores de uma linha. Para as linhas paralelas, é possível criar um operador partilhado que executa tarefas em mais do que uma linha, combinando operadores com bastante tempo inativo. Este operador pode ser alocado ao início das linhas de produção ou ao final das mesmas.

Em paralelo ao desenvolvimento dos modelos de programação linear, foi criada uma interface intuitiva e funcional no Excel. Esta interface permite a inserção de dados de produção e listas de tarefas, que são usados para alimentar os modelos do SolverStudio. Esta inclui funcionalidades adicionais, como validação de entradas e bibliotecas de tarefas e máquinas, facilitando a introdução de dados frequentemente utilizados.

Os resultados obtidos com a utilização do *ALB DSS* representam uma melhoria significativa em relação ao balanceamento manual. O sistema foi capaz de produzir uma solução em 3 minutos para a variante mais demorada, em comparação com os quase 50 minutos necessários para balancear manualmente as mesmas linhas, sendo que, embora ambos os métodos apresentem resultados bastante semelhantes, o *ALB DSS* apresenta sempre a melhor solução.

As grandes vantagens da utilização do *ALB DSS* não se limitam apenas à capacidade de produzir resultados viáveis num curto espaço de tempo, mas também à possibilidade de gerir, de forma

mais detalhada e sistemática, as várias restrições envolvidas no problema, permitindo uma alocação de tarefas otimizada. No caso do balanceamento manual, a probabilidade de erros aumenta significativamente, o que pode resultar num maior consumo de recursos para identificar e corrigir essas falhas.

À data de conclusão do projeto, o *ALB DSS* está pronto a ser usado nas tarefas de balanceamento de sistemas de montagem. Numa fase inicial, a sua aplicação focar-se-á no departamento de processo central. Após isso, esta será distribuída pelas várias fábricas do grupo Simoldes para a gestão à escala diária das linhas de produção.

Palavras-chave: Balanceamento de Linhas de Montagem, Linhas Paralelas, Programação Linear, SolverStudio, Engenharia de Processo

Abstract

Simoldes Plastics, a Portuguese company primarily focused on the injection of plastic components and parts for the automotive industry, sought to improve the responsiveness of its process department in adapting to ever-changing production conditions by developing a decision support system (*DSS*) capable of automating the assembly line balancing (*ALB*) of its systems.

To begin, the problem was thoroughly studied to identify key variables, system constraints, methods to quantify and achieve the intended goals, and outputs requirements. Additionally, various problem typologies were identified beyond the individual balancing of a single line, such as parallel assembly lines with shared workers. A mathematical formulation was then developed as the foundation for a linear programming model capable of producing feasible solutions. A smaller, specialized model was also created to optimize assembly systems involving multiple parallel lines.

Alongside the development of the linear programming models, an intuitive, feature-rich user interface was designed in Excel. This interface allows for the input of production data and task lists, which are then used to feed the SolverStudio models. It also includes additional functionalities such as input validation and a task and machine libraries to streamline the introduction of frequently used data.

The results achieved using the *ALB DSS* were a significant improvement over manual balancing. The *ALB DSS* provided the optimal solution within 3 minutes for the most time-consuming variant, compared to the nearly 50 minutes required to manually balance the same lines.

Keywords: Assembly Line Balancing, Parallel Lines, Linear Programming, SolverStudio, Process Engineering

Index

1 Introduction 1

1.1 Problem of Research, Framework, and Relevance 1

1.2 Research Question, Objectives, and Outputs 3

1.3 Methodological Options 4

2 Literature Review 5

2.1 Processes 5
2.1.1 Injection Molding 5
2.1.2 Assembly Lines 7

2.2 Assembly Line Balancing 8
2.2.1 Simple and General Assembly Line Balancing 9
2.2.2 Parallel Lines 12
2.2.3 Classification Scheme for *Assembly Line Balancing* 15
2.2.4 Precedence Graph 16
2.2.5 Objective Functions 18
2.2.6 Algorithms 19
2.2.7 The Future and Innovation in Assembly Line Balancing 21

3 Methods and Application 23

3.1 Problem Description 23
3.1.1 Case-Study of Parallel Assembly Lines 25
3.1.2 Classification of the Case-Study 30

3.2 Process Description 30
3.2.1 Example 1: Balancing of an Injection Line 34
3.2.2 Example 2: Balancing of an Assembly Line 35

3.3 Development of the *ALB DSS* 37
3.3.1 Individual *ALB* Variant 39
3.3.2 Parallel *ALB* Variant 42
3.3.3 Other Balancing Objectives 50

3.4 *ALB DSS* 51
3.4.1 Mathematical Formulation 51
3.4.2 SolverStudio with PuLP 54
3.4.3 Workstation Merger Algorithm 55
3.4.4 User Interface in Excel with *VBA* 57
3.4.5 *DSS* Outputs 66

4 Results 69

4.1 Comparison with Manual *ALB* 73

5 Conclusion 77

5.1 Future of the *ALB DSS* 78

Index of Figures

Figure 1 – Injection molding (Leo, 2022) 6
Figure 2 – Classification of *ALB* problems (adapted from Sivasankaran & Shahabudeen, 2014) 16
Figure 3 – Example of a precedence graph 17
Figure 4 – Assembly line layout 25
Figure 5 – Project timeline overview (Simoldes Plastics, 2022) 31
Figure 6 – Example of part used to verify estimated task times 33
Figure 7 – Door panel, example 1 34
Figure 8 – Line layout, example 1 35
Figure 9 – Door panel, example 2 36
Figure 10 – Line layout, example 2 36
Figure 11 – Variants of the *ALB DSS* 37
Figure 12 – Input Window (*ALB DSS*) 57
Figure 13 – Information screen (*ALB DSS*) 59
Figure 14 – Task List (*ALB DSS*) 60
Figure 15 – Task Library screen (*ALB DSS*) 61
Figure 16 – Machine/Equipment List (*ALB DSS*) 62
Figure 17 – Machine Library screen (*ALB DSS*) 63
Figure 18 – Model Options screen (*ALB DSS*) 64
Figure 19 – Task Library sheet (*ALB DSS*) 65
Figure 20 – Machine Library sheet (*ALB DSS*) 65
Figure 21 – *ALB* Output, 1 (*ALB DSS*) 66
Figure 22 – *ALB* Output, 2 (*ALB DSS*) 67

Index of Tables

Table 1 – Notation for *ALBP* 8
Table 2 – Task list for *FR LH* panel 26
Table 3 – Task list for *FR RH* panel 28
Table 4 – Task list for *RR LH* 29
Table 5 – *ALB* table before manual balancing 32
Table 6 – *ALB* table after manual balancing 32
Table 7 – Example task list 38
Table 8 – Solution with forward loading variant 40
Table 9 – Solution with backward loading variant 41
Table 10 – Solution with balance variant 42
Table 11 - Solution for the merge finish variant, after step 1 46
Table 12 – Workers assigned to workstations W1 to W9, after step 1 46
Table 13 – Solution for the merge finish variant, after step 2 47
Table 14 – Workers assigned to workstations W1 to W9, after step 2 47
Table 15 – Final solution for the assembly system with merge finish variant 48
Table 16 - Solution for the merge start variant, after step 1 49
Table 17 - Final solution for the assembly system with merge start variant 50
Table 18 – Results of the *ALB DSS* for forward loading variant 70
Table 19 – Results of the *ALB DSS* for backward loading variant 71
Table 20 - Results of the *ALB DSS* for Balance variant 71
Table 21 – Results of the *ALB DSS* for Merge Start variant 72
Table 22 – Results of the *ALB DSS* for Merge Finish variant 73
Table 23 - Results of the manual balancing using balance objective 74

Lists of Acronyms

Acronyms

ALB	Assembly Line Balancing
ALB DSS	Assembly Line Balancing Decision Support System
ALBP	Assembly Line Balancing Problem
BBR	Branch, Bound, and Remember
CAD	Computer Aided Design
CT	Cycle Time
DSS	Decision Support System
GALB	General Assembly Line Balancing
ISEP	Instituto Superior de Engenharia do Porto
MOLP	Multi-Objective Linear Programming
MnG	Minimum Graph
MxG	Maximum Graph
NP	Nondeterministic Polynomial-Time
OEE	Overall Equipment Effectiveness
OEM	Original Equipment Manufacturer
OF	Objective Function
pCT	Planned Cycle Time
P.Porto	Polytechnic Institute of Porto
SALB	Simple Assembly Line Balancing
SALBP	Simple Assembly Line Balancing Problem
SALBP-1	Simple Assembly Line Balancing Problem: Type-1
SALBP-2	Simple Assembly Line Balancing Problem: Type-2
SALBP-E	Simple Assembly Line Balancing Problem: Efficiency
SALBP-F	Simple Assembly Line Balancing Problem: Feasible
TG	Target Graph
VBA	Visual Basic for Applications

1 Introduction

This section provides a brief overview of the project's context, including the development environment and its relevance to the organization. Then, it presents the research question and the objective. Finally, the section explores the available methodological options used in the project.

1.1 Problem of Research, Framework, and Relevance

Simoldes is a notable international company, operating in the automotive industry and several others, with roots in Portugal and a long history that started in 1959. Simoldes began its operation with the production of molds for toy companies and the first time they ventured into the automotive industry occurred through a collaboration with with the Spanish firm Planosa, a supplier for Citroën. A few years later, Simoldes Plastics was founded to produce plastic components and found early success by supplying Renault, which had a brand-new factory in Setúbal. Simoldes works mainly with Original Equipment Manufacturers (*OEMs*) such as the Volkswagen Group (Volkswagen, Audi, Porsche) and Stellantis (Fiat, Citroën, Peugeot) (Fernandes, 2021). Between 1980 and 2018, Simoldes Plastics established a total of nine plants, scattered over three continents, with shared headquarters in Portugal to support their operations (Simoldes Plastics, n.d.-a).

INPLAS, a subsidiary of Simoldes Plastics, has been producing automotive parts using thermoplastics through the injection molding process since 1993. In addition, INPLAS has the capability of delivering painted parts or customized elements using the process of tampography (Simoldes Plastics, n.d.-b). This project was conducted within in the Operations Department at INPLAS, responsible for ensuring that the day-to-day activities of the company are running efficiently and effectively (Wicky, 2023).

In the modern world, owning an automobile is an indispensable necessity for a lot of people, being sometimes the only reliable form of transportation available for the daily commute to work, enabling personal mobility or allowing for bigger flexibility in managing professional and

personal appointments. In consequence, the automotive industry has become the beating heart of many industrialized countries, making the sector responsible for maintaining healthy international trade balances in these countries (Bell Rae & K. Binder, 2024). The evaluation of the entire automotive industry was quoted to be around the 2.7 trillion-dollar mark in 2021, as quoted by Business Research Insights (2024). Due to its size and being reliant on a very big amount of variables, it is subject to fluctuations in demand, which may only represent a small fraction of the overall worth but still represent considerable amounts of value. Some of the key factors driving market demand include the inflation rate, the price of consumables, the living standard of the population, and the public image of the companies (Sharma, 2012). These fluctuations are primarily felt by the automotive manufacturers but subsequentially trickle down the system of parts suppliers and service providers. For this reason, having a dynamic and flexible system is crucial to review these variations and maintaining a stable and efficient organization, providing quality to the clients in respectable delivery times.

As previously mentioned, the ever changing market demands in the automotive industry force the companies involved in the sector to have a certain degree of flexibility in their operations, from inventory management to production flexibility. Regarding the latter, the ability to effectively and swiftly program and balance the production line for new orders is one of the key factors in achieving flexibility. A balanced production line is better equipped to utilize its resources more efficiently, distributing the workload through its workers and machines. Balancing production lines is highly relevant in practice, as it enables greater adaptability to demand fluctuations. Other benefits can be the minimization of the number of workstations, the minimization of the cycle time, and the maximization of work-relatedness (Esmaeilian et al., n.d.).

Despite having clear objectives and sounding like a simple task, balancing a production line can be a complex undertaking. This exercise involves several input variables associated with the production and its tasks. Some of the variables considered are the daily production target, the available production time, the estimated cycle time, the overall equipment effectiveness (*OEE*), and many others. Another challenge facing assembly line balancing (*ALB*) is determining the minimum required number of workers and the distribution of the tasks among them. When only one worker is necessary to achieve the required output, all the tasks are attributed to that person. When two workers are needed, the question becomes determining the final task assigned to operator A and the first task assigned to operator B, introducing an additional level of complexity. The difficulty increases when more than three workers are needed, as in this scenario the problem enters the combinatory category, where the use of heuristics is valuable (Sivasankaran & Shahabudeen, 2014).

At the start of the project, at INPLAS, the *ALB* was achieved by manually calculating the minimum required number of operators, considering the production conditions and variables. After that, also done manually, the tasks were distributed by each of the required workers in a way that the cycle times were balanced. An important distinction was made between production lines, where an injection molding machine is present and dictates the cycle time (*CT*) to be achieved (taking the effective role of the bottleneck), and an assembly line, where the

pace is imposed by the takt time. The necessity of performing an *ALB* task may also arise in the definition of the process or in the production line itself, implicating that it is performed by different people. This can lead to inconsistencies in the generated work instructions and, as with any human-driven process, is prone to errors. These errors can lead to either an undetected issue, causing an imbalance in the process, or an identified mistake that requires rework, resulting in wasted time.

The envisioned solution to the presented problem is the creation of a Decision Support System (*DSS*) that is capable of determining the minimum required number of workers for an assembly/production line and balancing the tasks between those workers, approximating, as much as possible, the cycle time of each one. This *DSS* takes into consideration fixed parameters that intend to standardize the output generated. This *DSS* should also make it possible to test different scenarios, as it aims to substantially reduce the needed time to produce a work instruction for a process.

1.2 Research Question, Objectives, and Outputs

Confronted with the challenge presented, the current project attempts to answer the following question: How does an *ALB DSS* make the dimensioning and balancing of production lines faster and easier while delivering superior results and supporting more effective decision-making?

In summary, the project's primary objective is to develop a program that effectively dimensions and balances a production line. This program takes user-inputted information into account, including available production time, production volume, *OEE*, individual tasks, task durations, and task precedences. The objectives are outlined as follows:

1) Understanding the current method for balancing production lines and identifying how the process can be standardized, improved, and automated.

2) Pinpointing the necessities of the user and determining the constraints that influence the balancing.

3) Developing an algorithm that can efficiently determine the minimum number of operators required for the process and that allocates tasks in a balanced manner.

4) Developing a user interface that can facilitate the use of the balancing *DSS* and that has user-friendly features, ensuring compliance with the Simoldes standard.

5) Implementation of the *DSS* in the concerned departments.

6) Evaluation of the impact of the *DSS*, comparing its utilization with the manual calculations made before the development of the project, and quantifying the improvements achieved.

The expected outputs of the project are:

1) Production line dimensioned for the minimal number of workers necessary, with a balanced occupation rate in case the number of workers is bigger than one.

2) Construction of a *DSS* capable of being used for everyday use that facilitates the balancing of assembly lines in a reasonable timeframe, giving several options for the user to manipulate the output so it is as desired.

3) Creation of a dashboard with a global overview of the process, demonstrating the *CT* for each operator, the takt time, the planned *CT*, and highlighting the key details of each assembly line.

4) Creation of tools to further propel the creation of the work instruction and improve the user experience, such as libraries of frequently used tasks that can be quickly added to the task list.

1.3 Methodological Options

According to Coutinho (2014), the methodological options are organized within a hierarchical structure. From a bottom-up orientation, the base is the methodology as the system of methods used in a general field of application. Higher up, and taking a more focused approach, the methods are the aggregate of techniques utilized in the area. Finally, and at the top of the pyramid, taking the most specialized approach, the instruments and techniques are the tools used to perform a specific task (Caixeta & Fabricio, 2018).

Both Coutinho (2014) and Stockemer (2018) agree that the majority of scholars distinguish between two methodological perspectives, qualitative and quantitative. A qualitative approach intends to understand the situation and intention without imposing a theoretical referential (Coutinho, 2014). The quantitative approach examines observable facts and measures variables, enabling grounds for comparisons and relations (Coutinho, 2014).

Researchers use a set of strategies that are implemented in the investigation plan, to achieve the proposed goals (Fernandes et al., 2020). Some of these strategies can be classified as experimental investigation, where the output is studied in dependence on changes to the input variables; case study, where the investigation is conducted in the real-world scenario; and action research, where the investigator solves practical issues in the organization (Fernandes et al., 2020).

The current project adopts a quantitative approach, as it tries to dimension and balance production lines based on a tailormade algorithm and measure the induced impacts of this system on the industrial environment. It also falls under the category of action research, as it embodies the typical characteristics associated with this type of project.

2 Literature Review

With the aim of contextualizing the utilization of assembly line balancing (*ALB*) techniques and the several challenges associated with them, this section starts by describing the industrial environment in which the project was developed, exploring the manufacturing processes involved and the way they are integrated. Afterward, *ALB* is investigated in detail, starting with the different approaches to the problem and the classification scheme. The different objective functions for *ALB* are presented, as well as the algorithms used to solve it. Finally, the topic of the future and innovation in *ALB* is also discussed.

2.1 Processes

The dynamic relationship and integration of numerous processes are key attributes of automotive parts manufacturing. Each one of these procedures has been precisely tuned to achieve the desired output but only constitutes a small fraction of the final product. In this section, the various processes present at Simoldes, that take part in the *ALB*, are explained. By analyzing the intricacies of these processes, the reader may have a better understanding of the specific attributes, challenges, and issues surrounding them, which improve the comprehension of the measures used and methods applied in this project.

2.1.1 Injection Molding

Injection molding takes its role as one of the most important processes for mass production, and in many applications, it can deliver the final product with no extra steps required. The process itself has a very simple premise, which can be summarized as a quantity of melt being forcefully inserted into a close mold that represents the negative of the desired object, Figure 1. The object is then cooled until the material can hold its enacted shape and gets removed from the mold, going through other finishing processes if required (Ebnesajjad, 2015). Injection molding is able to generate complex three-dimensional shapes in a large range of materials and

specifications, making this process extremely useful in many fields of application (F. Francis, 2016).

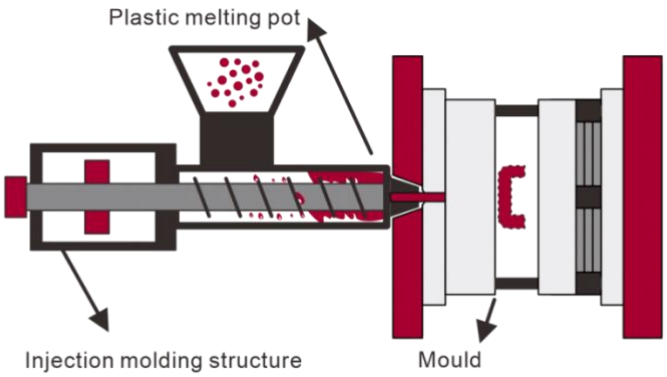


Figure 1 – Injection molding (Leo, 2022)

The majority of injection molding machines are designed to accommodate various molds within certain limits in dimension and closing force (Ebnesajjad, 2015). Another important consideration is the production of molds, which is a costly and intricate task, even for the manufacturing of simple components. Molds are often dedicated to single-product design and can only be used for the production of that specific component. The introduction of changes to the part demand, most of the time, the production of a new mold, which may take several months and elevate costs (Galizia et al., 2019).

Although injection molding is a commonly used process, it presents several challenges. Some of them are process-related, such as weld lines formed by the merging of the molten material flowing around an obstacle inside the mold, shrinkage and warping caused by the cooling of the material resulting in imperfections, or sink marks related to surface imperfections (Singh & Verma, 2017). Other challenges arise from the required injection molding technologies. The injection molding machines represent sizable and expensive equipment that requires fine tuning for optimal results, while the molds, as mentioned previously, incur significant costs and demand extended development periods.

In sum, despite presenting some constraints and challenges, the process of injection molding displays high productivity, which allows the process to achieve much lower prices per piece than its most relevant alternatives (Kryachek, 2004).

2.1.2 Assembly Lines

The most commonly used process in mass production environments is assembly lines, where workers with very limited training can efficiently perform simple tasks that, when added together, enable the assembly of complex products (Rekiek et al., 2002). This process can produce both standard and customized products, and significant savings can be realized through the optimal or near-optimal balancing of the production lines (Saif et al., 2014). Additionally, to increase efficiency, the engineers must increase the ratio between throughputs and required costs. The design, balancing, and optimization of these systems can be achieved through the use of exact methods, heuristics, or meta-heuristics (Rekiek et al., 2002).

Assembly lines are flow-oriented production systems where a set of equipment and tools, considered as stations, are used to perform incremental tasks in a workpiece. These stations are located in a specified order in the assembly line and each one of them requires the output of the previous one to proceed with its set of tasks (Saif et al., 2014).

The average time taken by a station to perform its assigned task is called cycle time (*CT*). A huge effort is made to balance the cycle times of all the stations along a production line to minimize the waiting times in the stations with lower workloads (Saif et al., 2014). This type of problem is called *ALB* and is the focus of the current project.

Assembly lines can vary in size, shape, and complexity, each solution being tailor-made to the desired application. According to Saif et al. (2014), assembly lines can be classified using the following aspects:

1) Assumption characteristics, taking either the simple or generalized classification.

2) Workflow characteristic, where the assembly line is either paced or unpaced.

3) Product characteristics, as the assembly line can produce only a single, mixed, or multiple models.

4) Layout characteristics, taking different shapes such as serial (a line), parallel, u-shaped, and two-sided.

Despite its extreme usefulness, the use of assembly lines carries some challenges. Due to the high interdependence between the workers and the machines, any minor shortcomings in a single station might have a trickle-down effect over the entire line. The solution starts with a meticulously executed assembly line balancing that synchronizes the stations. Predicting possible variations and applying measures to combat the variability of the processes is also important. Some of these measures include utilizing buffer inventory to supply the stations in the event of an upstream issue or not fully leveraging the equipment's performance to absorb variations (Pereira & Álvarez-Miranda, 2018). These specific solutions carry their drawbacks, so every scenario requires different thinking.

2.2 Assembly Line Balancing

According to Boysen et al. (2022), one could say that the assembly line balancing problem is to the production domain what the traveling salesman problem is to the transportation area. Line balancing consists of assigning a set of tasks to assembly stations distributed along a production line (Dolgui & Proth, 2013). This problem is known as the assembly line balancing problem (*ALBP*) and has been investigated for more than fifty years, starting with the first formulation by Salveson (1955). It aims to optimally assign tasks with a guiding metric to find a solution that best suits the situation at hand (Schmid et al., 2022). The assembly activities performed have an impact not only on the product's quality but also on its time-to-market. This reinforces the idea that the design and balancing of an assembly line are powerful tools for controlling the efficiency and quality of the operation (Rekiek et al., 2002).

According to Rekiek et al. (2002), the simplest form of an *ALBP* is popularly known as the Bin Packing Problem, which refers to a single product assembly line. The experiment aims to pack unidimensional objects of different sizes into as few as possible containers of a given capacity. Transforming the objects into tasks of different durations and the container as a metaphor for the cycle time, this problem becomes how to distribute as many tasks as possible to a station until the cycle time or the limit of the capacity of that station is reached. This oversimplified approach to an *ALBP* does not take into consideration some critical aspects, such are the precedence constraints faced in almost all real case settings.

The notation for an *ALBP* is offered below (Boysen et al., 2022):

$\boldsymbol{n}$	Number of tasks (index $\boldsymbol{j} = \boldsymbol{1}, \dots, \boldsymbol{n}$)
$\boldsymbol{V}$	Set of tasks $V = \{1, \dots, n\}$
$\boldsymbol{E}$	Set of precedence constraints $(i, j) \in E$ describing that task i must precede j
$\bar{\boldsymbol{m}}$	Upper bound on the number of workstations (index $k = 1, \dots, \bar{m}$)
$\boldsymbol{K}$	Set of potential stations $K = \{1, \dots, \bar{m}\}$
$\boldsymbol{p_j}$	Processing time of task j
$\boldsymbol{p_k}$	Processing time of station k
$\boldsymbol{c}$	Cycle time
$\boldsymbol{x_{jk}}$	Binary variable: 1, if task j is assigned to station k (0, otherwise)
$\boldsymbol{y_k}$	Binary variable: 1, if station k is utilized (0, otherwise)

Table 1 – Notation for *ALBP*

2.2.1 Simple and General Assembly Line Balancing

The simple assembly line balancing (*SALB*) is the most known and studied form of the *ALB* problem. It contains several simplifying assumptions that allow for its application in a large range of cases (Baybars, 1986). Succeeding attempts at widening the scope of the problem considered other practical concerns such as U-shaped lines and parallel stations (Becker & Scholl, 2006). As a result, the general assembly line balancing (*GALB*) was derived from the *SALB* problem. Because it contains fewer assumptions, *GALB* is considered a more practical approach (Saif et al., 2014).

As stated by Boysen et al. (2007), the base of any *SALB* problem is an assembly line with $k = 1, \dots, \overline{m}$ workstations aligned sequentially. The work in progress is launched down the line, passing through each one of the stations, where each task j is performed in order to modify the object. The time window between the arrival of two consecutive workpieces is considered the cycle time, c. The core objective of an *ALB* problem is to distribute the tasks as uniformly as possible through all the available stations to minimize c at the station with the highest processing time, which is the sum of all the task times p_j performed by that station, p_k. The execution of the tasks is subject to a precedence order, determining the sequence in which the tasks can be performed, and this is a necessary input parameter for any *SALB* formulation.

The processing time of station k, p_k, may never exceed the imposed cycle time for the line, but it can be smaller. In this case, the station has an inactive period, also known as an idle time, of $c - p_k$ time units for each cycle. The restricting assumptions for *SALB* are as follows (c.f. Baybars, 1986; Boysen et al., 2007; Boysen et al., 2022; Dolgui & Proth, 2013).

1) Production of a uniform product in mass quantities. (R1)
2) All tasks follow a predetermined process, with no alternatives. (R2)
3) Equal and fixed *CT* for all the stations, paced line. (R3)
4) Linear production, without feeder or parallel elements. (R4)
5) The sequence of tasks is determined by precedence constraints. (R5)
6) Deterministic task times. (R6)
7) The only restriction regarding tasks is the precedence constraint. (R7)
8) A given station needs to perform a task in its entirety. (R8)
9) All stations are equal, taking the same time to perform a specific task. (R9)

The assumptions of *SALB* are very restricting, limiting its application in real-world systems (Becker & Scholl, 2006). To create a more embracing approach, Boysen et al. (2007) proposes the following set of assumptions for *GALB*, making several modifications to the assumptions presented above for *SALB*.

1) Manufacturing of one or more known products. (R1')
2) A possible set of processing alternatives is given. (R2')
3) The *CT* can be imposed for customer demands or maximized according to assembly line capacity. (R3')
4) The flow is unidirectional. (R4')
5) The sequence of tasks is determined by precedence constraints. (R5)
6) The remaining restrictions are given up. (R6 to R9)

In conclusion, the comparison between *SALB* and *GALB* highlights their distinct levels of detail and practicality. Understanding the details and assumptions inherent to each method is crucial for selecting the most suitable strategy for creating an *ALB DSS*.

2.2.1.1 Subdivisions of the Simple Assembly Line Balancing Problem

The simple assembly line balancing problem (*SALBP*) can be subdivided into two main categories, regarding the objective function pursued. *SALBP-1* consists of assigning tasks to workstations so that, for a fixed production rate, the number of stations is minimized. *SALBP-2*, conversely, tries to maximize the production rate or, equivalently, reduce the sum of idle times across the stations on the assembly line (Esmaeilian et al., n.d.). Type-1 problems are more frequently associated with the design of new assembly lines, where the demand is pre-specified and, consequently, the production rate is fixed. On the other hand, Type-2 problems are commonly associated with changes to the assembly process or the production rate (Almeida Simaria, 2006).

The formulation for *SALBP-1* can be constituted by an objective function (1) and several restrictions (2) to (6).

$$\text{Minimize:} \quad \sum_{k \in K} y_k \tag{1}$$

$$\text{Subject to:} \quad \sum_{k \in K} x_{jk} = 1 \qquad \forall\, j \in V \tag{2}$$

$$\sum_{j \in V} p_j \cdot x_{jk} \leq c \cdot y_k \qquad \forall\, k \in K \tag{3}$$

$$\sum_{k \in K} k \cdot x_{ik} \leq \sum_{k \in K} k \cdot x_{jk} \qquad \forall\, (i,j) \in E \tag{4}$$

$$x_{jk} \in \{0,1\} \qquad \forall\, j \in V; k \in K \tag{5}$$

$$y_k \in \{0,1\} \qquad \forall\, k \in K \tag{6}$$

The objective of the formulation (1) is to minimize the number of stations utilized. Constraint (2) ensures that each task is assigned to only one station. Constraint (3) ensures that the sum of the task times for a station does not surpass the imposed cycle time and that when a station is used, the corresponding binary variable has the value of 1. Constraint (4) enforces the precedence relations, while constraints (5) and (6) define the domain of the variables (Pereira & Álvarez-Miranda, 2018).

Once again regarding *SALBP-1*, approximate solutions for this problem can be achieved by a simple algorithm. At each iteration, it selects the tasks with no predecessors or whose predecessors have all been completed and chooses the one with the biggest cycle time unless the station does not have enough capacity to perform said task, as the longer jobs are harder to allocate. The algorithm cycles through all the possible tasks until one is attributed or there are no more viable tasks, in which case a new station is required. It is necessarily assumed that there is not an operation with a task time bigger than the required cycle time for the stations, which would render the problem infeasible (Dolgui & Proth, 2013).

To improve this way of solving the problem, an iterative heuristic algorithm can be used. It would be more challenging to develop and require more computational power, but with enough time and iterations, it may be able to find optimal or near-optimal solutions. To achieve this, instead of selecting the task with the bigger cycle time, the algorithm would pick one at random within the pool of tasks that have a cycle time smaller or equal to the remaining capacity of the station and that have any or incomplete precedence tasks. This random selection would introduce a large number of possible combinations and improve the flexibility of the algorithm (Dolgui & Proth, 2013).

According to Dolgui and Proth (2013), in *SALBP-2*, "the investment is known, and the goal is to use it optimally" and, in Type-1 problems, "the investment is optimized while the desired productivity is known". To solve *SALBP-2*, *SALBP-1* can be resolved iteratively for a range of cycle times, to check the feasibility of a pre-determined number of stations (Almeida Simaria, 2006). Heuristics have been used for this application, as did Hackman et al. (1989) and Scholl and Voß (1997), but so have meta-heuristics. Anderson and Ferris (1994) and Kim et al. (1998) used Genetic Algorithms, and Taboo Search was used by Scholl and Voß (1997).

Over time, more variations of the *SALBP* start emerging, with different objectives and formulations. One well-known version is the *SALBP-E*, standing for efficiency. In this problem, the focus shifts to maximizing the line's efficiency, which is the ratio between the sum of all task times by the product of the imposed cycle time and the number of stations (Saif et al., 2014). *SALBP-E* tries to find the optimal combination of the number of stations and cycle time from within an instance that provides a set of possible cycle times and a window for the number of stations (Oksuz et al., 2017). Another commonly used variation is *SALBP-F* which attempts to find a feasible solution for a fixed number of stations and a given cycle time (Saif et al., 2014).

2.2.2 Parallel Lines

An additional variation to assembly systems is the inclusion of parallel assembly lines. Parallel lines can be considered as a system of several individual assembly lines, implemented near each other. These lines usually are similar in nature and follow the same direction of flow (Ismail et al., 2011).

The need for parallel assembly lines can arise from various factors. One common reason is to increase production speed when a bottleneck prevents further reduction in *CT*. Additionally, parallel lines can be beneficial for producing similar products with slight variations that require specialized equipment for assembly or transformation. Other reasons for implementing parallel assembly lines include the type of products being assembled, the processes involved, required output levels, available space, and other constraints.

Given the widespread use of parallel assembly lines, optimizing them as a cohesive system rather than as individual lines becomes crucial. By strategically arranging the lines and effectively managing task allocation within each line, overall system efficiency can be enhanced. This can be achieved by combining workstations across different lines to streamline processes and improve productivity (Ismail et al., 2011).

Gökçen et al. (2006) proposes two methods to tackle the parallel line problem, naming them passive and active approach. Both are subject to the current set of constraints:

1) One assembly line only produces one product.
2) Precedence relations for each product are known.
3) Task times for each product are known.
4) Operators working in each workstation of each line are multi-skilled (flexible workers).
5) The work can be performed on both sides of every line.

2.2.2.1 Passive Case Procedure

Regarding the passive case procedure, it focusses on the production of the same product in more than one parallel assembly line, with the same *CT*. The steps used to perform the balancing are:

1) Balance each assembly line as its own, not considering the parallel aspect of the problem.
2) Calculate the idle time for each workstation across all assembly lines.

3) Find a workstation with idle time equal or superior to half the imposed *CT* of the line and merge both parallel workstations as they should be the same. Repeat this step until it isn't possible to find a workstation that fits this criterion.

The model is also prepared for the addition of walking times between lines, which introduces a change to the third step. This is also considering that after performing the tasks related to a parallel station, the worker immediately returns to the original place, not performing several sets of tasks in a row on any particular line.

3') Find a workstation with idle time (where the idle time is the workstation real idle time minus the walking time between assembly lines) equal or superior to half the imposed *CT* of the line and merge both parallel workstations as they should be the same. Repeat this step until it is not possible to find a workstation that fits this criterion.

Overall, this procedure provides a logical approach to optimizing parallel assembly lines by balancing each line individually and then integrating workstations to enhance efficiency, not taking excessive impact in the distribution of tasks among workstations.

This approach provides greater flexibility in task allocation, as it avoids the complications of excessively managing tasks across multiple lines. Additionally, it effectively addresses idle time and optionally accounts for walking times between lines, ensuring that the final arrangement accurately reflects a real-world assembly system.

2.2.2.2 Active Case Procedure

The active case procedure was developed for assembly lines that produce different products to each other or variations of the same. Once again, Gökçen et al. (2006) propose the following steps to balance parallel assembly lines, now using the active approach:

1) Initialize of counters and reference values.

2) Start new trial.

3) Open new assembly line and new workstation.

4) For all tasks of the multiple assembly lines opened, determine a list of assignable tasks for a certain station (list F), guaranteeing that the task times are smaller or equal than the station's idle time.

5) The tasks in list F whose task time is equal to the idle time of the station are moved to list Z.

6) If the list Z has tasks, one is randomly chosen to be assigned to the workstation.

7) If the Z list is empty but the F list isn't, a task from this list is randomly chosen to be assigned.

8) If list F is empty and there are unassigned tasks available, go to step 3. If F is not empty, go to step 4.

9) End trial, save the solution in case it used up less workstations than the previous best solution.

As with the previous example, if walking times are taken into account, step 4 changes to the following:

4') For all tasks of the multiple assembly lines opened, determine a list of assignable tasks for a certain station (list F), guaranteeing that the task times are smaller or equal to the station's idle time for tasks of the same line or that the task times plus walking times are smaller or equal to the station's idle time.

The procedure is executed for all possible assembly line sequences with the best result being the one that utilizes the minimum number of stations.

This method adopts a more invasive approach to balancing parallel lines. Any changes to tasks, whether in duration, precedence, addition, or removal, can lead to significantly different balancing outcomes. While this flexibility is advantageous during the development phase of the assembly system, it may pose challenges once the system is implemented. Despite these potential drawbacks, the method can achieve higher efficiency by treating the assembly system as a unified problem rather than as separate lines to be integrated later (Gökçen et al., 2006).

2.2.2.3 Different Cycle Time

Gökçen et al. (2006) additionally suggest an approach for a particular case of the passive case procedure. When producing the same product across several production lines with the same cumulative time of all assembly tasks, a situation can arise in which the production values of the assembly lines aren't balanced. This leads to a necessity of rearranging the *CT* of the lines so that the volume produced at the end of the period is the desired one (disregarding the production of half units). A practical example of this is the need to produce 11 units during a shift by two parallel lines. One of these lines has to manufacture 6 units and the other 5, leading to a *CT* imbalance.

The steps proposed for this approach are:

1) Find the least common multiple of the cycle times.

2) Calculate D1 and D2 by dividing each worker's cycle time by the least common multiple. D1 shows the first worker's share, and D2 shows the second worker's share.

3) Calculate two precedence diagrams with different task times by multiplying the task times in each diagram with D1 and D2 values, separately.

4) Select the least common multiple as cycle time and balance the assembly line using the active case procedure.

2.2.2.4 Integration of Feeder Lines

Gökçen et al. (2010) take a new approach to the parallel *ALB* problem. The authors tackle not only parallel lines but also feeder lines. This establishes an even greater degree of interconnectivity and flexibility, but also adds complexity to the problem, all in a quest to achieve *Shojinka* (*sho* – to reduce, *jin* – worker, *ka* – to change).

The approach suggested starts by combining all precedence diagrams into a super-diagram containing all the tasks from all the assembly lines. This agglomerated diagram should take into consideration connectivity opportunities between multiple assembly lines, demanding some additional effort from the team responsible.

The proposed formulation has the objective of minimizing the total number of workstations required for the entire production system. As the precedences are combined, the resolution of the problem behaves in a way similar to *SALB*, where the allocation of tasks to workstations is only subject to precedence constraints and *CT* limitations. The formulation allows for some particularities such as tasks that must follow strictly after one another and disconnect lines that cannot be connected for physical reasons.

2.2.3 Classification Scheme for *Assembly Line Balancing*

ALB problems can be classified using the combination of different criteria such as the number of models produced in an assembly line, the nature of task times (whether they are deterministic or probabilistic), and the layout of the system (U-shaped, parallel, or straight) (Sivasankaran & Shahabudeen, 2014).

In a single production system, multiple different products or a mix of products can be manufactured, classifying the line as a multi-model assembly system. Alternatively, if the assembly system produces only one product, it is classified as a single-model system.

Regarding the task times, these can be considered deterministic when the tasks are very simple and performed by very precise tools and highly skilled workers, as the variability of the process is very reduced and controlled. In case the task times are longer, and the activities performed are more diverse and less simple, the variability should be considered in the construction of the problem, as it may have a significant impact on the overall performance of the system (Almeida Simaria, 2006; Sivasankaran & Shahabudeen, 2014). To further approximate the model to the real-world case, a stochastic component should be added to reflect the probability of

breakdowns and the duration of the consequent maintenance actions. Dynamic task times may also be considered to replicate the improved work cadence of workers during the shift (Almeida Simaria, 2006).

The third criterion contemplated refers to the layout and how the workstations are arranged. Sivasankaran and Shahabudeen (2014) proposed the diagram in Figure 2 to represent the classification of *ALB* problems.

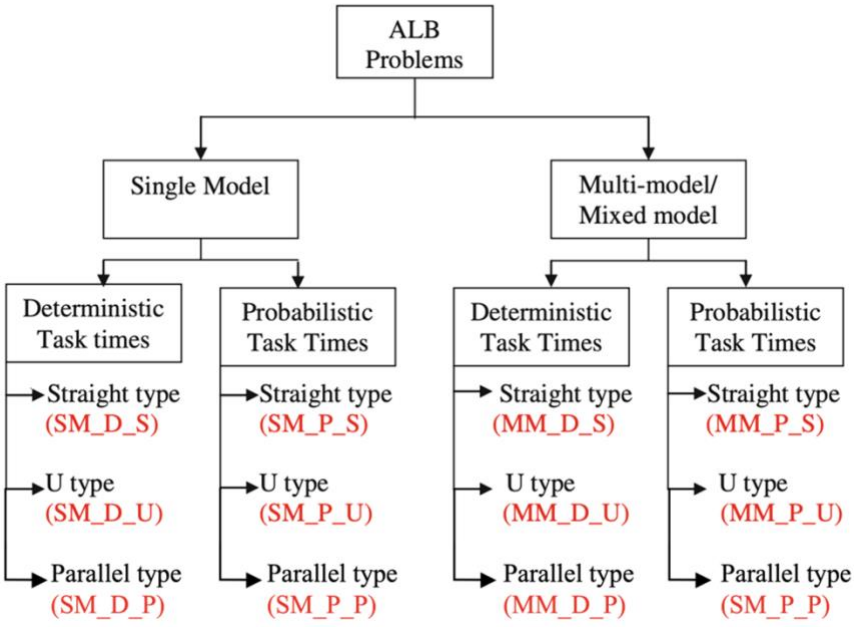


Figure 2 – Classification of *ALB* problems (adapted from Sivasankaran & Shahabudeen, 2014)

2.2.4 Precedence Graph

The development of models that correctly and efficiently resolve *ALB* problems is reliant on a large quantity of information related to the processes, the products, and the relationships between the tasks. Although the vast majority of scholars refer to the importance of obtaining this data, only few reveal adequate methodologies and explain the necessary steps to do so. This lack of guidance might be one of the reasons for the scarcity of real-scenario application of *ALB* (Boysen et al., 2007; Boysen et al., 2022).

According to Boysen et al. (2022), and referring to the formulation of *SALBP-1*, even the simplest formulation of an *ALB* problem requires information about the tasks, the task times, and the precedence relationship between them. The latter can be visually represented in a precedence graph, as shown in Figure 3. This graph shows the different tasks, located in the nodes, the task time associated with each, and the precedence relationship, denoted by the arcs. Following the direction of the arrows, the originating task must be finished before the start of the receiving one.

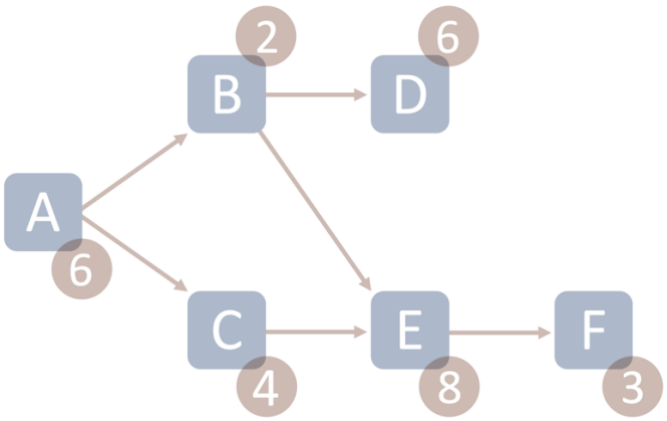


Figure 3 – Example of a precedence graph

In order to retrieve the necessary information to construct a precedence graph, several established procedures can be applied. To measure task times, Methods-Time Measurement can be utilized (Maynard et al., 1948). This system decomposes tasks into elementary procedures, and their durations are assigned through an analysis of pre-studied times. The task times are then calculated by summing up the durations of all the elementary procedures that collectively constitute the task (Boysen et al., 2022).

Although the required data can be gathered using proven tools, obtaining necessary precedence relations remains a challenge. Conventional research has focused on identifying promising task execution sequences, rather than constructing a complete precedence graph (Boysen et al., 2022).

However, these methods hardly see extensive use in the industry due to their high resource cost and the tendency to generate incomplete or inconsistent results that are challenging to maintain amidst ongoing product and process development. Consequently, accurate precedence information is often lacking or outdated, leading many companies to rely on manual line balancing done by human planners, rather than automated or semi-automated decision support tools (Boysen et al., 2022).

As explained by Boysen et al. (2022), a novel approach called the Learning Precedence Graph Concept was developed by Klindworth et al. (2012) and later proved to be capable of delivering good results for practical applications (Antani et al., 2013). This approach starts by approximating an initial unknown precedence graph, denominated by target graph or *TG*, through the construction of maximum and minimum graphs (*MxG* and *MnG*, respectively). *MxG* is developed from historic task sequences that have been used in the past. The *MnG* is constructed by adding verified precedence relations to an initially empty arc set. The learning process involves removing precedence relations from *MxG* in a bottom-up manner and adding them to *MnG* in a top-down way, until both graphs are equivalent. This stopping point reveals *TG* without having to manually establish each constraint related to the procedures. *TG* can be challenging to obtain directly, but the experiments conducted demonstrate that near-optimal

ALB can be achieved from this method, even if the initial gap between *MxG* and *MnG* is substantial and only a portion of the data regarding the problem is available.

2.2.5 Objective Functions

Objective functions (*OF*) guide the progression of algorithms, ensuring they stay on a path that generates increasingly better solutions. The definition of the *OF* is always problem-specific and it profoundly dictates the performance and results of the algorithm (Li et al., 2017). The most evident objective for an *ALB* problem is to minimize the number of workstations and minimize the *CT* (Boysen et al., 2022; Li et al., 2017), although this may be inefficient in solving larger instances.

Regarding the minimization of the number of stations, the lower bound is restricted by a planning horizon that dictates the production rate, a consequence of the established *CT*, and the upper bound might be restricted by physical aspects, such as available space. Another possible *OF* can be focused on maximizing the assembly line efficiency, guaranteeing the fulfillment of the production rate and minimum number of necessary stations (Boysen et al., 2007).

Expense-related functions can also be utilized, such as cost minimization or profit maximization (Boysen et al., 2007; Becker & Scholl, 2006). The costs can be attributed directly to segments of the production system, such as the tasks, the workstations, or the resources. The constructed model dictates how these costs are allocated in specific cases, but the flexibility provided allows for the development of accurate and useful solutions. The objective of a cost-oriented algorithm is to minimize the combined operational cost for a variable number of workstations, which takes a similar approach to *SALBP-1*. Additional elements could be introduced such as an initial investment to create a station (Amen, 2000). The profit simply reflects the difference between the revenue acquired and the indispensable costs to produce that turnout. It may be used as an extension for the cost-oriented formulation, introducing fixed selling prices for the items produced.

Furthermore, it is important for an *ALB* problem to measure how well the tasks are balanced between stations. This objective function is analogous to single-product lines and multi or mixed lines. Other metrics could be employed, for example creating a score to evaluate the bottleneck, or measuring the efficiency of the assembly line, such as the balancing efficiency of the system (Boysen et al., 2022). The balancing efficiency of an assembly system can be evaluated by the formula provided by Panneerselvam (2005), where $Balancing\ Efficiency = {Sum\ of\ all\ task\ times}/{Number\ of\ workstations * CT} * 100$. Disregarding any objective, the model could only be searching for feasible solutions, instead of trying to improve on any other aspect (Boysen et al., 2007).

Since the use of a single *OF* can be ineffective, as different goals might result in the same number of workstations and *CT*, the introduction of a second *OF* can prove itself to be useful in

producing more precise results (Li et al., 2017). Dealing with multiple *OFs* can take three different paths (Boysen et al., 2022). The first consists of using the weighted sum as a general and complex *OF* that the algorithm tries to improve upon. Each of the goals is given a different ponderation according to their relative importance and they are usually exploited to merge capacity- or *CT*-oriented formulations to workload balancing, or cost-oriented goals (c.f. Nearchou, 2008; Hamta et al., 2013). The second approach is to use a lexicographic ordering of objectives, in which a primary goal is defined and, in case different solutions are equivalent, the next objective in the hierarchy order is used as a "tiebreaker" (Mozdgir et al., 2013). The third and final methodology is to use Pareto-optimality. Multiple objectives are still present, but they are not combined into a single *OF* nor are they given different levels of relevance. Instead, an algorithm following this approach looks at the set of goals and find the solution that better improves them individually without affecting the others adversely (Yoosefelahi et al., 2012).

2.2.6 Algorithms

Since its conception, several optimization methods have been proposed to tackle *ALB*, addressing the inherent complexities in diverse ways, to improve the assembly system and streamline operations. In its majority, the *ALB* problem is *NP*-hard, meaning that it is "at least as hard as the hardest problems in nondeterministic polynomial-time (*NP*)". The *NP* class refers to problems for which the feasibility is easily checked, even if finding a possible solution is demanding. For this reason, many of the proposed solutions to the *ALB* problem, such as branch-and-bound and dynamic programming procedures, are only applicable to very restricted versions. To approximate the constructed model to practical cases, additional constraints must be introduced, and some assumptions must be altered or removed, increasing the complexity of the problem. Therefore, exact solution algorithms must be replaced by heuristics, that sacrifice optimality for computational efficiency and aim at providing reasonably good results within a practical timeframe (Boysen et al., 2022; Pereira & Álvarez-Miranda, 2018).

This section aims to review different types of algorithms commonly used in *ALB*, from more limiting perspectives applied to simple problems to more encompassing ones.

2.2.6.1 Exact Solution Procedures

Exact solution procedures are algorithms that search the entire space of possible solutions to a problem, in order to find the most fitting one. As opposed to an approximate solution, this method guarantees that the final output of the algorithm is without a doubt the best possible scenario for the restrictions imposed. However, for the procedure to explore all possible solutions, the problem cannot take very large proportions, which would lead to very considerable computational demand and extensive processing times, making this alternative unsuitable for many practical applications.

A new and effective branch-and-bound approach to *SALBP-1* was introduced by Sewell and Jacobson (2012), called branch, bound, and remember (*BBR*). It considerably outperforms other

exact methods and is able to tackle much larger problems in a reasonable timeframe. *BBR* effectively solves all small and medium-sized instances, most large-sized instances with only a few exceptions, and achieves solutions for two-thirds of very large-sized instances within an hour of computational time each (Boysen et al., 2022).

The main difference between the traditional branch-and-bound algorithms and *BBR* is that it stores the complete state in memory, which increases the available memory requirements. Nevertheless, this ability allows the avoidance of revisiting equivalent partial solutions, contributing to the efficiency and effectiveness of this method (Pereira & Álvarez-Miranda, 2018).

2.2.6.2 Construction Heuristics

To determine a starting point for local search procedures or to define the upper or lower bounds for exact solution algorithms in case of minimization and maximization respectively, the construction of a feasible *ALB* solution is required (Boysen et al., 2022). Some modern alternatives have been proposed, but the traditional approach is the popular Hoffmann heuristic (Hoffmann, 1963), which is still widely used for the *SALBP-1* (Pereira & Álvarez-Miranda, 2018).

As explained by Pereira and Álvarez-Miranda (2018), the heuristic details all the feasible tasks for a given station and selects the best alternative following a greedy criterion, which normally tries to reduce the idle time. This process is repeated for the next station in line until all the assignments have been allocated. Additionally, the author proposes four changes to the heuristic to better adapt it to the problem at hand. The first is limiting the maximum number of assignments generated for a station, in order to keep the running time within reasonable intervals. The second is to apply the heuristic on the original and reversed instances, taking advantage of the reversibility property. The third change focuses on the inclusion of robustness in the criteria used to choose from the available assignments. The fourth and final suggestion is to vary the importance attributed to the parameters to obtain different results.

2.2.6.3 Meta-Heuristics

Meta-heuristics are general search strategies used to solve combinatorial optimization problems by searching large regions of the solution space without getting caught in local optima, which is a drawback of pure local search algorithms (Almeida Simaria, 2006). Currently, researchers have at their disposal a large variety of meta-heuristics that are capable of reaching near-optimal solutions in many cases, with the field of *ALB* being one of those (Boysen et al., 2022). Many of the available meta-heuristics draw inspiration from real-life events and phenomena, to simulate characteristics that took nature millions of years to develop and that are often associated with evolution.

Local-search-based meta-heuristics engage with and guide procedures for local improvement, with tabu-search and simulated annealing fitting this category. Genetic algorithms, ant colony optimization, and particle swarm optimization are some examples of population-based and population-inspired meta-heuristics. Moreover, hybrid meta-heuristics that combine different algorithms to achieve improved results are emerging as a new trend (Boysen et al., 2022).

2.2.7 The Future and Innovation in Assembly Line Balancing

The field of Assembly Line Balancing holds the potential for substantial future advancements that could significantly enhance production systems. Automation and the evolution of integrated solutions offer promising avenues to attain heightened efficiency, cost-effectiveness, flexibility, and optimization within assembly line processes.

Boysen et al. (2022) identify a huge level of demand and potential for developments in the following three areas: combined models that integrate *ALB* with related decision tasks, *ALBPs* customized for the different life cycle phases of an assembly line, and *ALBPs* for smart factories of the Industry 4.0 era.

Combined models address the challenge of decision-making for interconnected aspects such as balancing, sequencing, and other related fields like logistics and personnel planning. Advancements in computing power and optimization software have facilitated the development of better models capable of improving efficiency, being more cost-effective, and more decisively predicting planning results. Traditional combined models were able to merge balancing and sequencing, but new technologies can create more deeply interconnected links with the production system. Despite their many advantages, combined models bring with them some drawbacks. To start, they are very complex systems, that demand careful consideration of the changes they may induce to the organizational decision process. These programs also require a huge investment of resources before they are operating at cruising speed, which may not make them immediately preferred (Boysen et al., 2022).

Current models restrict themselves by only focusing on the initial line balance, when heavy and hard-to-move equipment is being installed in their fixed spots, not considering future phases of the lifecycle of an assembly line and the products that result from it. Despite this, many deep changes happen further down the timeline, which forces the undergoing of *ALB* studies to accommodate the alterations. Future methodologies should consider pre-production line determination, re-balancing of existing lines, and adaptation to varying model mixes, cycle times, and oscillations in the workforce. The referred initial line balance is made several months before the start of production when there are still some uncertainties. In this phase, the product may not be fully defined, complicating the retrieval of necessary data such as the definition of tasks, calculation of *CTs*, and determination of the precedence constraints. For this reason, a significant advantage would come from the possibility of the *ALB* influencing product development collaboratively (Boysen et al., 2022).

In the context of smart factories, two different evolution paths can be outlined: the first refers to flexible shift models, and the second to flexible assembly paths. With societal modernization, the workforce has begun to advocate for more flexible work hours, shifting away from the fixed shifts that are commonly found today. In this proposed arrangement, shifts don't have a defined start or end point, with the workers being able to arrive at their discretion in between a specified time frame or at several incremental points during that window. This requires constant re-balancing and real-time adaptations to the everchanging resource availability, having to consider task relocation and the qualification of the available workers. The flexible assembly paths consider a reality in which the products are transported along the assembly line by a fixed conveyor but by independent and autonomous robots. Therefore, the workpieces aren't obliged to follow the same linear path and can visit the stations in an adaptable manner, assuming the precedence constraints are respected. Despite this system cutting the need for a uniform *CT* among all the stations, the one with the heaviest workload still becomes the bottleneck and generate unavoidable inefficiencies (Boysen et al., 2022).

The Industry 4.0 revolution brought with it the widespread utilization of collaborative robots in assembly systems, generating new challenges for the *ALB* revolving around task allocation, workload balancing, and operation scheduling (Kheirabadi et al., 2022; Kheirabadi et al., 2023, Li et al., 2019, Yoosefelahi et al., 2012). New and more detailed models have been developed to create optimization algorithms that comply with the Industrial Revolution and the advancements in technology. This evolution reflects a crucial shift in assembly line dynamics, and it represents a growing trend and signs for the future. The works mentioned go over the application of *ALB* formulations in industrial places where workers and collaborative robots work hand in hand to achieve a productive and safe work environment.

Finally, Boysen et al. (2022) suggest the use of machine learning software to support the generation of the precedence graphs, previously addressed in Section 2.2.3. The author believes that this technology could learn from *CAD* files of parts and their relative position in the final assembly to determine which tasks should be done initially and the ones that need the completion of others to be executed. The realization of this suggestion would significantly accelerate the data retrieval process.

3 Methods and Application

3.1 Problem Description

Various problem types related to injection and assembly lines were identified, such as parallel, mixed-model, and grouped tasks. However, establishing a starting point was essential to guide the project's development and prioritize the most pressing challenges. From these, the parallel lines problem emerged as the most significant. This problem type is not only the most prevalent but also the one that can generate the biggest improvement opportunities. Hence, the decision to address this problem first was strategic, as it also lays a solid foundation for future diversifications and enhancements of the *DSS*. The decision to prioritize this type of problem was validated by several process engineers at Simoldes, affirming its alignment with the company's objective and challenges. This section intends to explore the type of assembly systems present at Simoldes, understand what characterizes them, and how they can be integrated to achieve enriched results.

This study has been developed at the production department of Simoldes, with the main focus on the production lines of door panels produced in plastic, to be used in the automotive industry. Despite this, the proposed solution is applicable to other product lines within Simoldes and may even be relevant in other industries that share similar overall qualities.

The automotive industry is a demanding market, highly associated with high production rates and low *CT*. This means that small improvements and optimizations in operations can lead to significant results due to the large number of repetitions for each task. An accurate and efficient balancing of the assembly lines minimizes idle times for both operators and machines, ensuring optimal utilization of resources.

The selected product was chosen because it is a relevant source of revenue for the company, the most significant product within Simoldes' production facilities, and the one that most justified the need for an automated solution due to its complexity. Door panels are composed of multiple components assembled on the main carrier of the panel. Although every assembly line is unique, the common practice is to run production batches of each component on distinct production lines. Some of these parts only require the injection process while others are subject to the assembly of small components such as clips or foams, to painting, or wrapping. The feeder lines are outside of the scope of the first approach, as they are production and not assembly lines. Additionally, the majority of cars have a set of four door panels (both front and rear, left and right) that are produced on very similar parallel lines. On some occasions, however, left and right door panels might be different due to the presence of specific controls or buttons on the driver's side.

Assembly lines are product specific, as they are designed for a very precise sequence of tasks or processes, and the equipment are custom made to fit the shape of the assembly. In a single assembly line, it is only possible to have different versions of the same model, for which the variations are not significant in terms of key processes and geometry. In these cases, the lines are dimensioned for the most complex version of the part, and the simpler ones simply skip some of the steps.

The assembly lines can also vary depending on the type of final packaging. Returning to the case of door panels, they can either be packaged individually or be grouped by sets (one panel of each). This restricts the configuration of lines and their positioning relative to one another. There are additional variants to the packaging that influence the assembly lines themselves, such as the need for sequencing. This occurs in mixed-model production scenarios where different versions of the same door panel are manufactured on the same assembly line. These panels are sequenced in the order required by the client's production schedule for assembling cars at their own factory.

Inside Simoldes' production systems, several typologies of problems can be identified. The main ones with significant presence are related to grouped tasks (also known as zoning constraints), parallel assembly lines, and mixed-model assembly lines. Since each of these problems requires significant research and development time, it was decided to initially focus solely on parallel assembly lines. This decision allows for the construction of a good base on top of which the other problems can be solved in later stages.

The result provided by the *DSS* developed is only a suggestion to the collaborator responsible for the *ALB*, as a software program constructed for balancing is not capable of analyzing every other real-world constraint of the assembly line. In some cases, it may not be possible for a worker to move between two parallel assembly lines due to physical barriers or long distances, so it is up to the process engineer to accept or refuse a proposed solution.

The purpose of this *DSS* for the production department is to provide a means for testing various scenarios to support informed decision-making.

3.1.1 Case-Study of Parallel Assembly Lines

To get a better understanding of the type of assembly systems the *ALB DSS* has been developed to solve, consider the following example of four parallel lines dedicated to a set of door panels. This project remains in the development phase, so it was a good candidate to test the output of the *DSS*.

This example refers to an assembly system built to assemble a set of four door panels: front left (*FR LH*), front right (*FR RH*), rear left (*RR LH*), and rear right (*RR RH*). The daily production target is 1108 car sets and, considering an *OEE* of 90% and an opening time of 22,5 hours each day, the planned *CT* is 65,79 seconds for all the lines in the system, even though they produce different products. The left and right panels are identical for the rear panels, despite being mirrored. For the front ones, the left panel (driver's side) has a slight modification from the right one.

The layouts for the factory that will receive this project still have not been finalized, and changes are still possible. However, the latest development of the layout is presented in Figure 4.

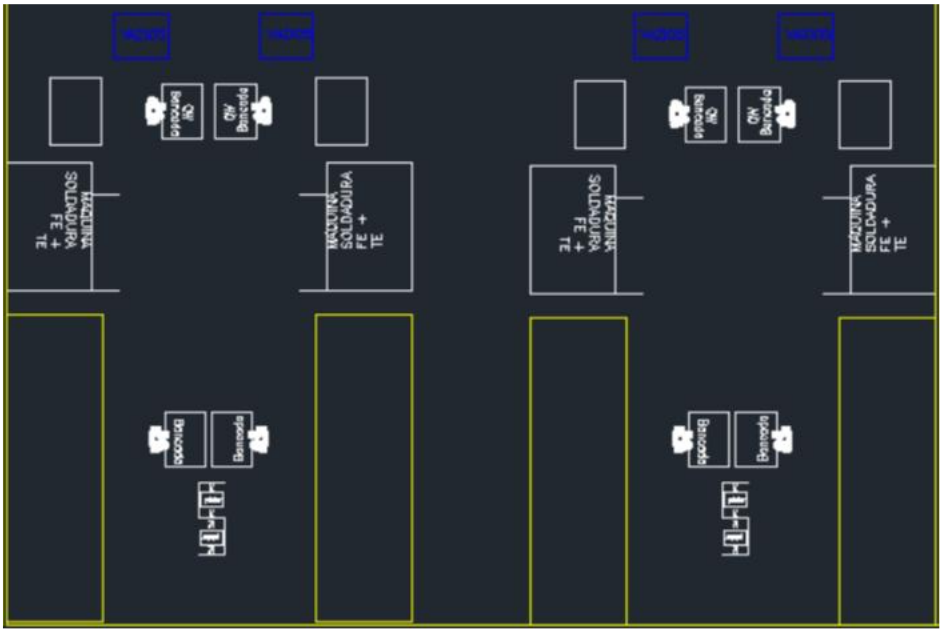


Figure 4 – Assembly line layout

The layout shows the four parallel assembly lines mentioned above. From the left to the right, it displays *FR LH*, *FR RH*, *RR LH*, and *RR RH*. As mentioned previously, the line works with a *CT* of 66 seconds to meet the demand of the client of 1108 car sets per day.

The task list for the *FR LH* panel is as follows in Table 2.

#	List of Tasks	*Task Times*	Precedences
A1	Read the Label, grab the Panel, glue the Label, and position the Panel on the assembly base (visible side facing up)	6	-
A2	Grab the Armrest	5	1
A3	Assemble the Armrest	5	2
A4	Rotate the Panel (technical side facing up) and change base	5	3
A5	Grab the Center Frame	5	4
A6	Place the Center Frame in the assembly base (visible side facing up)	5	5
A7	Grab the Handle Bezel	5	4
A8	Assemble the Handle Bezel on the Center Frame	5	6, 7
A9	Grab the Decor	5	4
A10	Assemble the Decor on the Center Frame	5	6, 9
A11	Grab the assembled Center Frame and assemble it on the Panel	5	8, 10
A12	Grab the Pocket	5	11
A13	Assemble the Pocket on the Panel	5	12
A14	Assemble Boot Button	5	13
A15	Assemble the Foams (x2)	8	4
A16	Assemble the Plastic Clips (x7)	21	4
A17	Place the Panel in a transfer base	5	11, 14, 15, 16
A18	Grab the Panel and place it in the welding machine	8	17
A19	Read the Label	3	18
A20	Remove the previously welded Panel from the welding machine	5	18
A21	Trigger the start of the welding equipment	2	18, 19, 20
A22	Place the Panel in the inspection base	5	20, 21
A23	Visually inspect the finished Panel (visible and technical side)	10	22
A24	Place the Panel of the final base	5	23
A25	Read the Label and place the Panel in the indicated place of the final packaging	5	24

Table 2 – Task list for *FR LH* panel

The sum of the task times required to produce the mentioned door panel is 148 seconds. Considering that the maximum available *CT* for a workstation is 66 seconds, at least 3 workers are required for this assembly line. In an ideal scenario, each one would share a *CT* of 49,33 seconds per cycle, meaning they would be perfectly balanced.

There is a total of 50 seconds of idle time shared across the 3 workstations of the line for each cycle, or 16,67 seconds per workstation. The distribution of this idle time across the workstations varies regarding task assignment.

The task list for the *FR RH* panel is as follows in Table 3. It varies slightly from the previous one because it does not require task 14.

#	List of Tasks	*Task Times*	Precedences
B1	Read the Label, grab the Panel, glue the Label, and position the Panel on the assembly base (visible side facing up)	6	-
B2	Grab the Armrest	5	1
B3	Assemble the Armrest	5	2
B4	Rotate the Panel (technical side facing up) and change base	5	3
B5	Grab the Center Frame	5	4
B6	Place the Center Frame in the assembly base (visible side facing up)	5	5
B7	Grab the Handle Bezel	5	4
B8	Assemble the Handle Bezel on the Center Frame	5	6, 7
B9	Grab the Decor	5	4
B10	Assemble the Decor on the Center Frame	5	6, 9
B11	Grab the assembled Center Frame and assemble it on the Panel	5	8, 10
B12	Grab the Pocket	5	11
B13	Assemble the Pocket on the Panel	5	12
B14	Assemble the Foams (x2)	8	4
B15	Assemble the Plastic Clips (x7)	21	4
B16	Place the Panel in a transfer base	5	11, 13, 14, 15
B17	Grab the Panel and place it in the welding machine	8	16
B18	Read the Label	3	17

B19	Remove the previously welded Panel from the welding machine	5	17
B20	Trigger the start of the welding equipment	2	17, 18, 19
B21	Place the Panel in the inspection base	5	19, 20
B22	Visually inspect the finished Panel (visible and technical side)	10	21
B23	Place the Panel of the final base	5	22
B24	Read the Label and place the Panel in the indicated place of the final packaging	5	23

Table 3 – Task list for *FR RH* panel

The sum of *CT* is now 143 seconds with the removal of task 14 from the *FR LH* line. This impact is not considerable, but it increases the production cadence of the line, which would become significant after a substantial number of cycles. However, as the number of door panels assembled by each of the lines is the same, these extra seconds become idle time, as the assembly system enforces the same planned *CT* to all its assembly lines.

The number of required workstations is still 3 and the perfectly balanced *CT* is 47,67 seconds. This results in 55 seconds of idle time across the three workstations.

The task list to assemble rear door panels is usually smaller when compared to the front ones. It is also unlikely that the right and left door panel are not exact copies of each other, which happens in the current example. For this reason, the task list of the assembly lines for *RR LH* is shown in Table 4. The task list for the *RR RH* line follows a similar structure, but the task codes begin with the letter D.

#	List of Tasks	*Task Times*	Precedences
C1	Read the Label, grab the Panel, glue the Label, and position the Panel on the assembly base (visible side facing up)	6	-
C2	Grab the Armrest	5	1
C3	Assemble the Armrest	5	2
C4	Grab the Decor	5	1
C5	Assemble the Decor on the Panel	5	4
C6	Rotate the Panel (technical side facing up) and change base	5	3, 5
C7	Grab the Handle Bezel	5	6

C8	Assemble the Handle Bezel on the Panel	5	7
C9	Grab the Pocket	5	6
C10	Assemble the Pocket on the Panel	5	9
C11	Assemble the Foams (x2)	8	6
C12	Assemble the Plastic Clips (x8)	24	6
C13	Grab the Panel and place it in the welding machine	8	8, 10, 11, 12
C14	Read the Label	3	13
C15	Remove the previously welded Panel from the welding machine	5	13
C16	Trigger the start of the welding equipment	2	13, 14, 15
C17	Place the Panel in the inspection base	5	15, 16
C18	Visually inspect the finished Panel (visible and technical side)	10	17
C19	Place the Panel of the final base	5	18
C20	Read the Label and place the Panel in the indicated place of the final packaging	5	19

Table 4 – Task list for *RR LH*

The sum of the *CT* for each of the rear door panels is 126 seconds, requiring only two workstations for the *RR LH* and another two for the *RR RH*. Dividing the 126 of total *CT* by the planned *CT* of the system, the result would be 1,91 workstations. This is an indication that the idle times are very small, and that the *CT* of both workstations are balanced. Naturally, this only applies if the tasks times and precedences allow the tasks to be allocated to only two workers. The sum of idle time is then 6 seconds across both workstations for a single line.

To combine workstations across assembly lines, they should be near to avoid considerable walking distances and times. For this, the ideal lines to merge a workstation across would be *FR LH* with *FR RH*, *FR RH* with *RR LH*, or *RR LH* with *RR RH*. As the layout has still not been finalized, the process engineer still has autonomy to modify the configuration of the system. This decision can be supported by an automatic *DSS* that proposes the best scenario for each assembly system, such as the one being developed during the course of this project.

3.1.2 Classification of the Case-Study

The presented case serves as the standard example of an assembly system. For that reason, the problem can be classified according to its characteristics.

In Section 2.2, various challenges were examined to identify the most effective approaches for addressing them. This section aims to identify the specific typology of challenges present in the case study to guide the chosen solution approach.

The problem deals with parallel assembly lines, falling under the General Assembly Line Balancing Problem (*GALBP*), which demands more flexibility than the stricter limitations of the Simple Assembly Line Balancing Problem (*SALBP*). It aligns with a Type 1 classification, with the primary objective being to reduce the number of required workstations across the entire assembly system, considering all the lines involved.

The case study naturally emphasizes parallel lines, where assembly lines operate under the same planned *CT* but may assemble either similar versions of the same product or different products. Since each line is considered to produce only a single version of the product, this falls under the single-model variant of *ALB* problems. Deterministic task times are also considered.

To summarize, according to the classification scheme in Figure 2, the case study is categorized as SM_D_P.

3.2 Process Description

The foundation for creating any enhancement tool or methodology lies in acquiring an in-depth and expansive understanding of the existing state. This involves a profound comprehension of how the information is gathered and subsequently utilized in shaping the *ALB* system. A critical prerequisite for conceiving a focused and impactful improvement strategy is to recognize the system's strengths, weaknesses, and operational intricacies from its outset. Consequently, the following section is dedicated to a thorough exploration and depiction of the initial state. It begins by explaining the various stages of a new project and detailing how and when the *ALB* is utilized. Two practical examples are provided in Section 3.1.1 and 3.1.2. This effort aims at establishing a succinct foundation that underpins all the subsequent proposed improvements.

A project for the industrialization of a new product must follow a roadmap and evolve through several established phases, such as suggested in Figure 5. These phases start with a request for quotation, the product and process are then developed, the serial sets are manufactured, and the product and process are validated. Ultimately, the team can approve the start of the ramp-up in production volumes, aligned with the date agreed upon with the customer for the start of production (*SOP*). At this point, production reaches the desired levels. Each phase is constructed upon the quality of the development achieved in the prior phase, systematically building up towards the goal of the project.

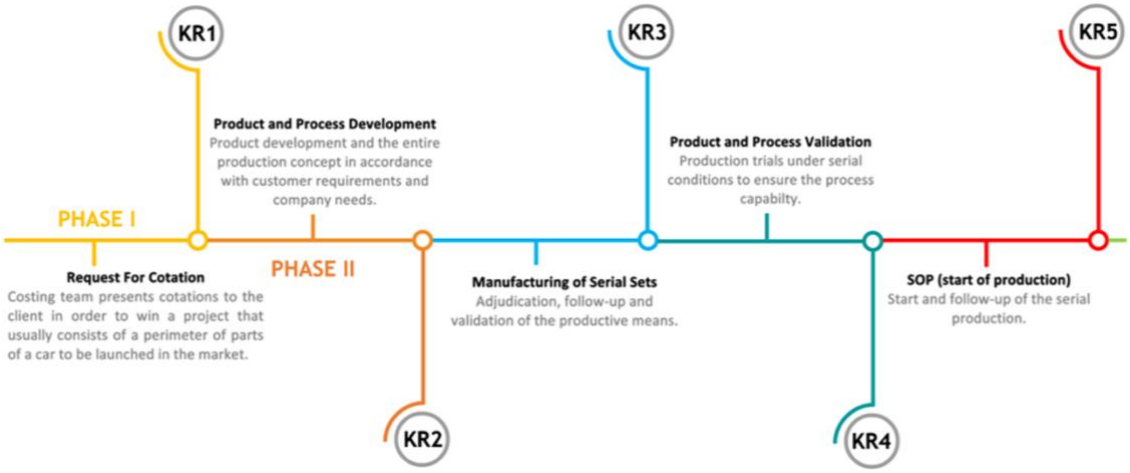


Figure 5 – Project timeline overview (Simoldes Plastics, 2022)

The first stage is the Request for Quotation (*RFQ*), where a costing team presents a quotation to a client to secure a project. This quotation is based on past experiences with similar components and technologies. A pre-study is carried out to determine inputs such as machine requirements, peripherals, other resources, processes, and production cadence that results in a price per piece for the client.

The second phase of the project refers to the specification of the production means. Here, the product is developed to align the vision of the client with the feasibility of the parts. The processes and their sequences are determined based on the required components to be produced and task precedences, originating a Value Stream Mapping (*VSM*). The different steps for the production and assembly of each component of the final product are added to a Bill of Process (*BOP*) spreadsheet and the *CT* for each task is estimated by an experienced process engineer, although the cadence of the line is determined by the injection mold engineering. The production cadence of the line is determined by the *CT* of the injection machine, as it is a very expensive equipment to operate, and its capabilities need to be explored to the maximum. In case the line only consists of assembly tasks, the cadence is dictated by the takt time, derived from the daily production target, operating time, and *OEE*. The values for the peak production numbers are calculated to check if the equipment and the factory are capacitive.

In order to develop a baseline version of the production layout, an *ALB* can be performed to facilitate the calculation of the required number of workers and the tasks attributed to each one. The *ALB* is achieved manually using similar tables to Table 5 (before balancing) and Table 6 (after balancing). In these, the tasks are added in sequential order to the first operator, and the standard time for each is determined based on experience, until this one achieves or surpasses the planned *CT* for the line. When this happens, the following tasks are added to the next worker and the process is repeated until all the tasks are attributed. Sequentially, the tasks are moved by hand between two workers until the line is balanced to a sufficient level. This process is inefficient, prone to errors, and does not guarantee the optimization of the balancing.

Balancing two workers might be an easy and straightforward undertaking, but as the number starts to grow, the number of possible combinations increases exponentially, and it becomes significantly harder to perform the *ALB* in respectable amount of time, while managing all the constraints reliably.

Before ALB

OPERATOR 1	t(s)	OPERATOR 2	t(s)
Collect mirror from packaging	4	Mount screw	7
Analyze the piece	2	Assembles spring assembly (spring + plate + screw)	13
Mount the mirror	6	Collect the part and check it according to the control range	20
Assemble the technical part	12		
Assemble the epdm mat	10		
Place the assembly on the equipment base	3		
Collect the part and check	6		
Glue 3 foams	15		
Place the piece on the base	3		
Assembles spring + mirror + shaft	12		
73		**40**	

Injection CT (seconds): 65

Table 5 – *ALB* table before manual balancing

After ALB

OPERATOR 1	t(s)	OPERATOR 2	t(s)
Collect mirror from packaging	4	Place the piece on the base	3
Analyze the piece	2	Assembles spring + mirror + shaft	12
Mount the mirror	6	Mount screw	7
Assemble the technical part	12	Assembles spring assembly (spring + plate + screw)	13
Assemble the epdm mat	10	Collect the part and check it according to the control range	20
Place the assembly on the equipment base	3		
Collect the part and check	6		
Glue 3 foams	15		
58		**55**	

Injection CT (seconds): 65

Table 6 – *ALB* table after manual balancing

The third phase adjudicates, follows-up, and validates the productive means specified in the previous phase with the production of pre-series. During this stage, the initial examples of the parts are produced and assembled manually, without utilizing serial production equipment. This approach allows for the examination of process validity and flow, as well as the identification of potential issues or sources of inefficiency. All the required equipment such as welding machines, wrapping presses, and peripherals for assembly and detection are specified and adjudicated during this step of the project. The majority of this equipment is specifically made for the product.

To check the accuracy of the task times used for the *ALB* performed in the second phase of the project, and to simulate real assembly conditions, a prototype of the part is constructed, and all its components are gathered. A worker, or even the process engineer responsible for the *ALB*, simulates the assembly of the product (example in Figure 6) in the sequence established

by the precedence constraints while timing the several steps individually, or videorecording, to measure the times afterward.

A modest number of samples is generally enough for the degree of precision required in this step, as the goal is only to validate the times used previously. The number of samples and the methods used for measurement are entirely at the discretion of the worker conducting the *ALB*.

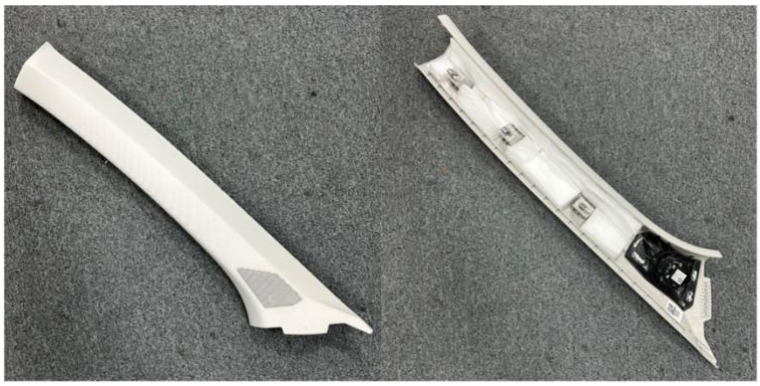


Figure 6 – Example of part used to verify estimated task times

Still in the third phase of the project, the *ALB* is again performed taking into consideration the more accurate task times, despite the process of executing the balancing being the same and still done manually. Any change results in the creation of a new work instruction, as they are continually refined through iteration.

Phase four of the project is when the production line starts to be created and the processes are industrialized and validated under serial conditions. The machines are installed, the workers are trained, and the mold is tested. Batches of production are tested in incrementally increasing cadence levels and duration to ensure that the equipment, factory, and workers can meet the desired targets. Any problems, issues, or improvement opportunities detected during a production trial originate a plan-do-check-act (*PDCA*) procedure, with a deadline of three months, so they can be solved or implemented before the following trial. With the workers now in their positions within the production layout, the *ALB* conducted in the previous stages of the project undergoes its most rigorous evaluation. The *ALB* is improved to reflect any changes in the process and to update the work instruction to reflect any modification in task times or number of workers. Despite the importance of having a well-balanced assembly line, the *ALB* always works under the upper limit of *CT* imposed by the injection molding process. The number of required workers is also derived from the time between injections and from the cumulative time of all the tasks. Despite this, the *ALB* is still performed manually.

The final phase of the project starts with the *SOP* as the team at the factory takes total responsibility for the project. The production line starts working continuously at full capacity and the *ALB* might be used to compensate for production volumes, operating times, and line efficiency. One year after the *SOP*, the client usually requests a cost-reduction plan, in which

the production means are studied and analyzed in order to locate possible gains. These may consist of *CT* reduction, removal of excess material and redundant components, improvements in the process, or adoption of new technologies. Any of these changes may generate the need for a new *ALB*.

To get an even deeper understanding of the *ALB* process, two examples of balancing are presented. The first relates to an injection line and the second one to an assembly line.

3.2.1 Example 1: Balancing of an Injection Line

The first case study focuses on the front door panels (left and right) of a commercial vehicle (Figure 7).

Figure 7 – Door panel, example 1

This production line combines both injection and assembly, but as the dominant element is the injection machine, the *CT* is dictated by the injection process. The layout of this production line is presented in Figure 8.

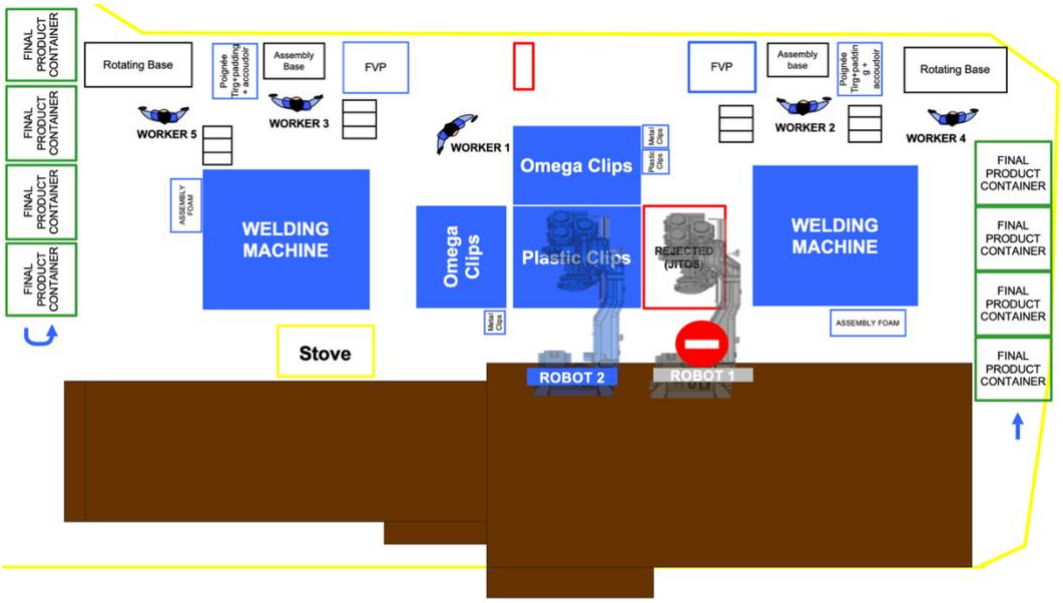


Figure 8 – Line layout, example 1

In this scenario, both left and right panels are injected simultaneously in a single cycle of the injection machine. Robot 2 removes both panels and places them in an automatic cell that inserts clips and that detects defects. Worker 1 then inspects the panels and places them in their respective buffer for worker 2 and 3, redirecting the workpieces into two similar parallel lines. After this separation, the two mirrored workpieces don't come into contact again, and are placed into final product containers that are composed solely of similar finished panels.

To simplify these cases, both sides of the line could be considered as standalone assembly lines, starting with the buffer of injected panels after the insertion of the clips. All the tasks between the injection and the placement of the panels in the buffers could also be considered another line to be shared by the other two. For a more generalist approach, with less rules and separation of the production line in sections, only the precedence constrains should be considered.

Other tasks, such as restocking clips (which occurs once over a predetermined number of cycles) or placing defective parts in racks (which happens intermittently), need to be performed by workers with low occupation rates or by a dedicated worker solely responsible for restocking components in the assembly lines.

3.2.2 Example 2: Balancing of an Assembly Line

This second example studies the assembly line for a front-left door panel (Figure 9). All four panels for this car model (front and rear, left and right) are produced in similar assembly lines. In this case, the lines are not considered as parallel because there is no common element or joint precedence. The lines start with queues of the injected base of the door panel, with all the

smaller components being supplied in supermarkets located near the workstations where they are required.

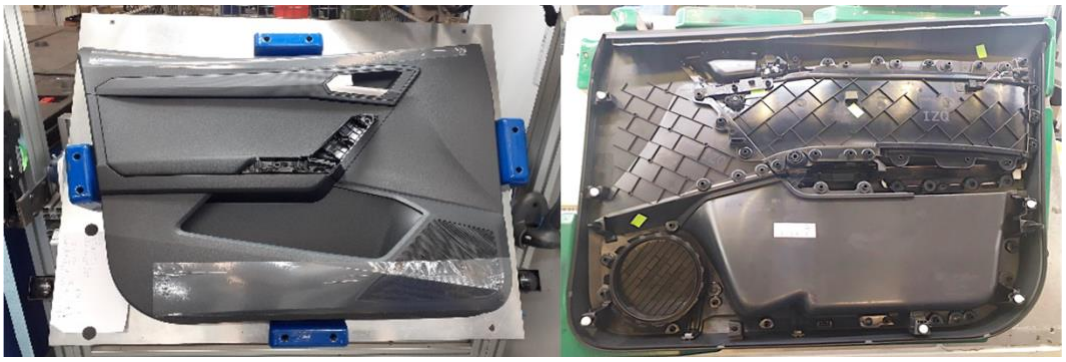


Figure 9 – Door panel, example 2

The layout for the assembly line is represented in Figure 10. To maintain the production rate and to satisfy the takt time, the line required five workers in five workstations (*WS*). This example fits perfectly in the description of *SM_D_S* (consult Figure 2) and, despite having a relatively large number of steps and more than a few workers, it is still considered a simple problem for *ALB*. The flow of materials is unidirectional, the precedence of tasks only allows one production path, and the workers perform an uninterrupted sequence of tasks before handing the workpiece to the next worker in line or to the final product package.

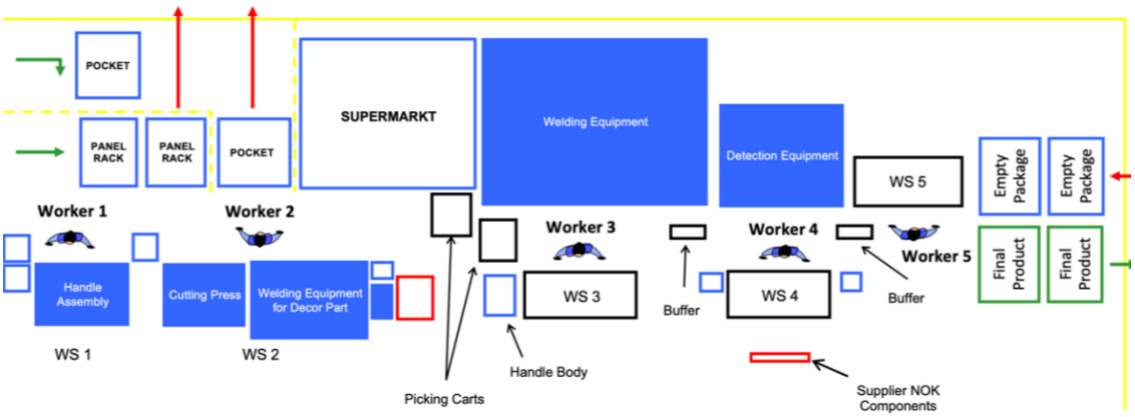


Figure 10 – Line layout, example 2

In such examples, performing the *ALB* is determining the handover tasks – meaning the last task the operator performs – from the first operator to the second, from the second to the third, and so on. A line with two required workers and five tasks only has four possible handover tasks and four possible solutions. A line with three workers and five tasks now has three possible handover tasks from the first operator to the second and the other three from the second to the third, resulting in six possible solutions. The number of combinations only increases as the number of workers and tasks grow and, although time-consuming and resource-demanding to test all possible solutions, it is moderately simple to find a feasible one.

3.3 Development of the *ALB DSS*

The goal for this assembly line balancing decision support system was to create a powerful and user-friendly solution that empowers process engineers to find high-quality solutions within a short timeframe. The *DSS*' rapid response capability enables users to make better decisions by providing access to a larger range of solutions and allowing them to test different parameters to find the ideal scenario. It is also crucial for the *DSS* to be adaptable, enabling process engineers to fine-tune it according to the specific circumstances of their assembly systems. This section presents the ambition for the *DSS*' functionalities and how it aims to assist users in finding the desired solutions effectively.

The *ALB DSS* is intended to give several options to the process engineer balancing the assembly lines. These options are also required to have a tool capable of resolving several types of problems with varying sizes. These options are in Figure 11.

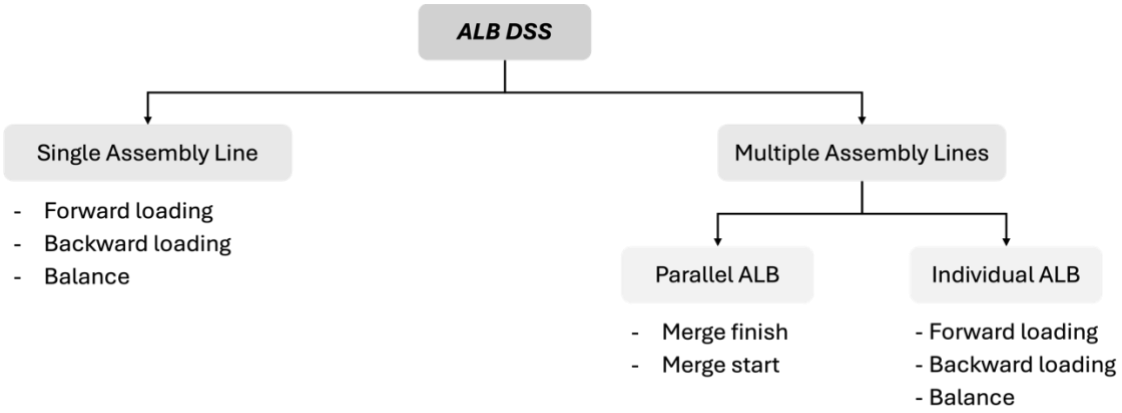


Figure 11 – Variants of the *ALB DSS*

The objectives made available for the *ALB DSS* were determined by the needs of the process engineers at Simoldes. The main necessity is naturally the balancing of task times across all workers in the assembly line. The possibility to push all the tasks to the front/rear of the line, releasing *CT* for the last/first worker is also a useful option to have, as it can be used to integrate their *CT* with other tasks such as refilling the racks within the assembly system, restocking items of large consumption such as clips and foams, reworking some items, or perform other tasks specific to the assembly line or system. These types of tasks are not considered in the current iteration of the *ALB DSS* as they are not occurring every cycle, and managing the necessity for the execution of them may be unpredictable. To conclude, enabling users to forward or backward load the assembly line provides the flexibility to manipulate the solution and achieve desired outputs under specific circumstances. This capability is particularly useful in optimizing the assembly line to meet unique production goals or adapt to varying operational constraints.

When balancing a single assembly line, the *DSS* provides only the option to perform individual *ALB.* In an assembly system, with multiple parallel assembly lines, the user is able to choose between individual or parallel *ALB*.

For the individual *ALB*, each line is balanced independently and the results for each are presented as their own, not compared and analyzed with the other lines in order to find improvement opportunities. The *DSS* then provides the user with the option to forward load the assembly lines (assigning tasks from the first workstation to the last), backward load (from the last to the first), and to balance (balancing the *CT* between all the workstations).

When parallel *ALB* is selected, all lines are balanced and then merged if possible. The shared workstation can be either created at the front or rear of the line (merge at start or finish) with the ultimate goal of reducing the number of required workers in an assembly system.

To help the understanding of each type of solution, the following set of tasks in Table 7, with sequential precedences, can be considered. The following sections explore each of the objectives the *ALB DSS* provides to the user and describe their application and functionality in a real-world scenario.

#	Task description	*Task Times*
T1	Grab the panel and evaluate the visual quality	10
T2	Place the barcode label on the technical side of the panel	5
T3	Place the panel on the assembly base	5
T4	Assemble the light bar on the panel	15
T5	Clip the connectors for cables of the lightbar	20
T6	Place the panel on the assembly base	5
T7	Assemble the armrest	15
T8	Assemble the pocket	15
T9	Assemble the insert	15
T10	Place the panel on the detection equipment	5
T11	Assemble 4 plastic clips	10
T12	Apply the protection film	10
T13	Remove the panel from the equipment	5
T14	Perform the final visual inspection	5
T15	Package according to packaging guidelines	10

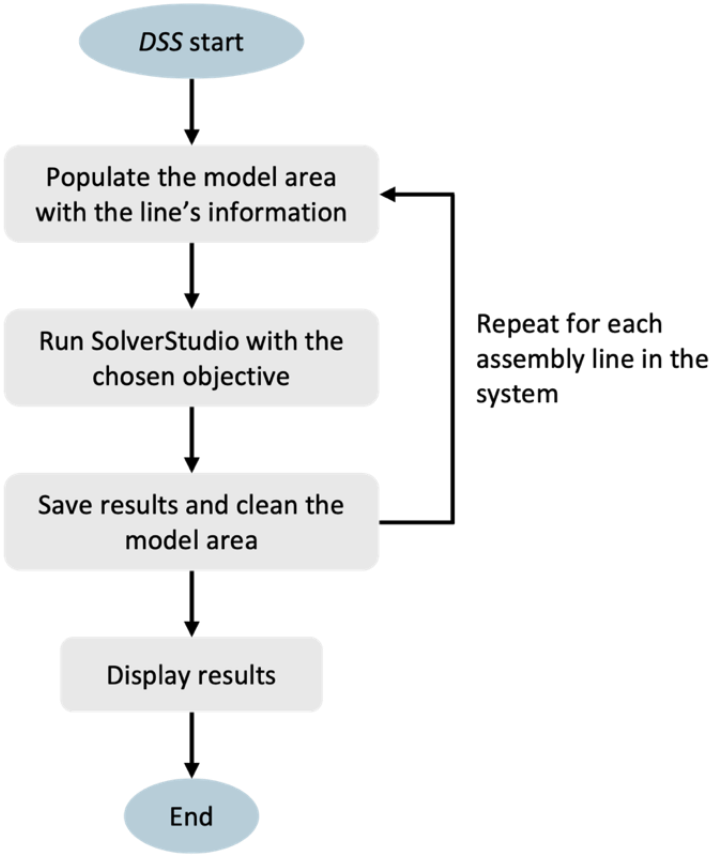
Table 7 – Example task list

This assembly line has a planned *CT* of 60 seconds and the sum of all the task times is 150 seconds. This means the assembly line requires at least three workers, as two workers without any idle time would only be able to perform a combined cumulative *CT* of 120 seconds to operate within the imposed cycle.

3.3.1 Individual *ALB* Variant

The individual variant treats each assembly line as an independent unit, even in systems with multiple lines. The mathematical model is applied separately to each line. The only shared information across all lines is the planned *CT*.

Flowchart 1 illustrates the sequence of steps that the *ALB DSS* follows during an individual line balancing operation.



Flowchart 1 – Individual *ALB* Variant

3.3.1.1 Forward Loading Objective

The forward loading objective intends to focus on the assembly line from start to finish, following the task sequence since the first step to the final assembled product. To achieve this goal, the task assignment has the goal of minimizing the idle time for the first workstation of the line, finding the task combination that achieves this without breaking the precedence rules. When the *CT* of the first workstation is saturated and it does not fit anymore tasks, the following workstation is opened and the process is repeated, until all tasks are assigned.

The utility for this objective does not come from the literature, but from the experience of the process engineers responsible for the *ALB*. The ability of freeing the *CT* of the last workstation can have benefits such as providing a goal to task optimization in order to close it, freeing one worker. It also opens the possibility to relocate the worker assigned to the last workstation to other required tasks in the assembly system or in the line, in case it cannot be automatically done by the *DSS*.

Returning to the task list provided as an example, the result from the *ALB DSS* when performing the individual balancing with the forward loading objective looks similar to the one presented in Table 8.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''	20''	5''										*60''*
WS2							15''	15''	15''	5''	10''					*60''*
WS3												10''	5''	5''	10''	*30''*

Table 8 – Solution with forward loading variant

Workstations 1 and 2 are fully saturated, with no available idle time. In contrast, Workstation 3 remains open, as it has not reached its 60-second limit. The worker assigned to Workstation 3 still has available time within the cycle and can be assigned additional tasks until the limit is reached.

3.3.1.2 Backward Loading Objective

The backward loading objective is very similar to its counterpart, the forward loading objective. In this case, however, the task assignment is done in a reverse order, from the final assembled product to the first task in the assembly line. Instead of freeing *CT* in the last workstation of the line, the backward loading objective frees the first workstation.

In terms of real-world applications, the backward loading objective is once again very similar to the forward loading objective. Regarding the assembly line, type of product, layout constraints

or logical sequence of tasks, it may be more beneficial to keep the last workstations full and give the first one more agility. The possibility of solving a single instance with both objectives gives the user more freedom in obtaining the intended solution for the assembly line.

Considering the task list example once again, the expected solution looks like the one presented in Table 9.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''												*35''*
WS2					20''	5''	15''	15''								*55''*
WS3									15''	5''	10''	10''	5''	5''	10''	*60''*

Table 9 – Solution with backward loading variant

For the same task list, this example reveals how a different objective can return completely different solutions. The task arrangement does not allow to fully occupy the *CT* for workstation 2, leaving it with 5 seconds of idle time. Although it is a small inactive time, the idle time of workstation 1 reduces, as it has to perform the task time that did not fit into workstation 2. This might compromise the execution of additional tasks for the worker assigned to workstation 1.

3.3.1.3 Balance Objective

The objective to balance the *CT* across all the workstations in an assembly line might be the most useful in terms of advantage to the process engineer, as it is the most reoccurring scenario and because it is the most difficult to achieve manually. In a large assembly line, with a big number of tasks and complex precedence relations, it may take several hours to achieve a reasonable balance between all the workers, without the guarantee that it is the optimal solution. The challenge of this task lies in the extremely large number of possible combinations, which makes it difficult to test them all during manual balancing.

The ability to automatically perform assembly line balancing with this objective allows the user to update task allocation to the workstations whenever a task time is modified or improved, a new task is added or removed, or when production conditions (operating time, production volume, *OEE*) change.

Taking the example used, the output of the *ALB DSS* should be the one presented in Table 10.

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''	20''											*55''*
WS2						5''	15''	15''	15''							*50''*
WS3										5''	10''	10''	5''	5''	10''	*45''*

Table 10 – Solution with balance variant

From all the solutions proposed up to now for the other objectives, the balance is the only one that provided a solution in which every workstation has idle time. For a perfect balance, each workstation would have a *CT* of 50 seconds and 10 seconds of idle time. However, the task duration and precedence relations do not allow for that scenario. In the presented case, despite having variations, the presented solution is the one that approximates the *CT* for the workstations the most.

The use of the balance objective provides the scenario for each the tasks are balanced between workstations as best as possible, making sure that they have enough idle time in each cycle to absorb the variation that occurs in every process. This ensures that, even when a task takes longer than its deterministic time used for the balancing, it can still fit inside the *CT* and not disturb the production flow of the line.

3.3.2 Parallel *ALB* Variant

The parallel variant of the problem is intended to find the possibility of allocating a single worker to perform the tasks of various workstations across different lines in the same assembly system, regarding this worker does not exceed the limit of the *CT*. The goal is to reduce on the number of workers involved in the assembly system and to reduce the idle time across all the workstations, improving the efficiency.

The correct method to balance an assembly system with multiple parallel assembly lines depends on the types of tasks performed, amount of *CT* available per cycle, layout configuration and type of product assembled. Several methodologies have been analyzed and considered, but the one that better fits Simoldes' production model is the passive approach. This approach proposes balancing each line individually, then identifying workstations with available *CT* that can free a worker to perform tasks at a workstation in another line, all within the imposed CT limit. An additional proposal is to utilize the forward or backward loading objective to perform the initial individual balancing. This creates a last or first workstation with available *CT*, respectively. The example of the assembly line presented in the Section 3.2.1 shows two parallel lines with a worker performing the tasks for the first workstation of both. The assembly line layouts at Simoldes are generally compact to maximize floor usage and optimize the placement

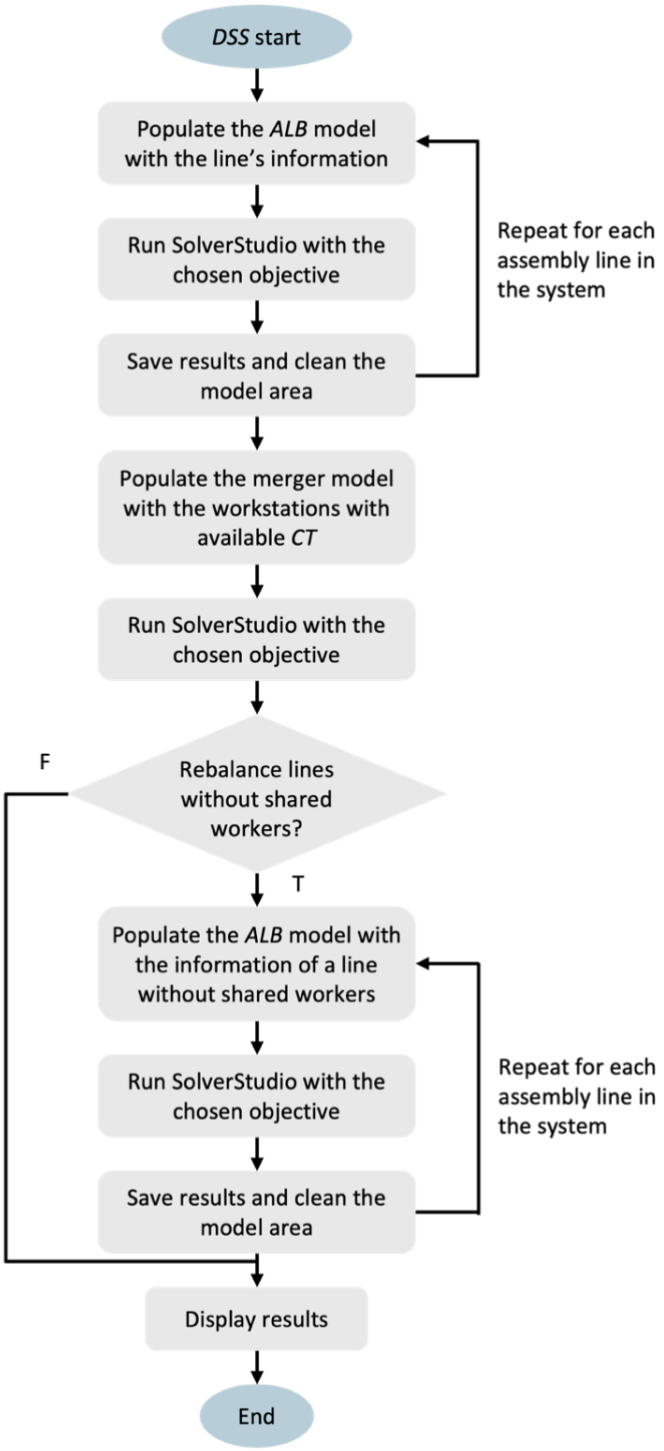
of assembly benches, component racks, and, in some cases, injection machines. Because of this, workers often find it easier to merge the first or last workstations.

After the individual balance of all the assembly lines in the system, either using the forward or backward loading goal, the solutions require post processing to merge the workstations in parallel lines. The *DSS* looks for the idle time in the first or last workstation of all the assembly lines in order to find the ones that can be combined. The assembly system has a maximum amount of six lines, as it provides a margin over the regular number of parallel lines in a system. The largest assembly systems usually have four lines for the production of a full set of door panels.

To find the best combination, the *DSS* can look at all the combinations and find the one that optimizes the system the most, minimizing the idle time in the workstations and reducing the number of workers in the assembly system.

Finally, after this stage, any line whose first or last workstation was not merged with another line can be rebalanced independently. This allows for a final balancing using the individual approach to achieve the balance objective for those specific lines. This option can be given to the user as it only requires additional computational time.

Flowchart 2 illustrates the sequence of steps during a parallel line balancing operation.



Flowchart 2 – Parallel *ALB* Variant

3.3.2.1 Merge Finish

After choosing the parallel balancing option, the user is able to choose between merging the first workstation of the lines or the last. In case the user chooses to merge the end of the lines, the last workstation of each line should have remaining *CT* to allow this modification. This implies using the minimizing objective when each assembly line is balanced individually.

After having all the assembly lines balanced, the *DSS* tests the different combinations until it reaches the best possibility to reduce number of workers and the amount of idle time in the assembly system.

To get a better understanding of how the *DSS* solves this type of problem, an assembly system with three assembly lines can be considered. Each assembly line has the same task list as the one presented earlier as an example. All the assembly lines work under the 60 seconds of planned *CT,* and all require at least 3 workstations, but the number of workers can be lower as one can perform the tasks of more than one workstation. Table 11 to Table 15 represent the different steps that are followed by the *DSS* to transform an instance into a balanced solution.

Step 1 – Individual balancing for all the assembly lines.

Firstly, each assembly line is balanced individually using the minimization objective.

Line 1	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''	20''	5''										*60''*
WS2							15''	15''	15''	5''	10''					*60''*
WS3												10''	5''	5''	10''	*30''*

Line 2	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	***∑CT***
WS4	10''	5''	5''	15''	20''	5''										*60''*
WS5							15''	15''	15''	5''	10''					*60''*
WS6												10''	5''	5''	10''	*30''*

Line 3	T31	T32	T33	T34	T35	T36	T37	T38	T39	T40	T41	T42	T43	T44	T45	∑CT
WS7	10''	5''	5''	15''	20''	5''										*60''*
WS8							15''	15''	15''	5''	10''					*60''*
WS9												10''	5''	5''	10''	*30''*

Table 11 - Solution for the merge finish variant, after step 1

Table 11 shows the individual task allocation for all the workstations in the three assembly lines. As the task list is the same for all lines, the allocation is the same, although this is not the norm in real assembly lines. The last workstation of each line (W3, W6, and W9) still have *CT* available in the cycle of 60 seconds. The worker assigned to each workstation is shown in Table 12.

Workstation	Worker
WS1	Worker 1
WS2	Worker 2
WS3	Worker 3
WS4	Worker 4
WS5	Worker 5
WS6	Worker 6
WS7	Worker 7
WS8	Worker 8
WS9	Worker 9

Table 12 – Workers assigned to workstations W1 to W9, after step 1

Step 2 – Merge workstations if the combined *CT* is not superior to the planned *CT*.

In the current case, there are three workstations with 30 seconds of idle time in a cycle of 60 seconds (Table 13). Two of them can be merged into a single worker, while the other maintains its idle time. As all the assembly lines have the same amount of idle time, all the combinations result in an equally efficient assembly system. In the current example, the workstations chosen for the merger are highlighted in green.

Line 1	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	∑CT
WS1	10''	5''	5''	15''	20''	5''										60''
WS2							15''	15''	15''	5''	10''					60''
WS3												10''	5''	5''	10''	30''

Line 2	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	∑CT
WS4	10''	5''	5''	15''	20''	5''										60''
WS5							15''	15''	15''	5''	10''					60''
WS6												10''	5''	5''	10''	30''

Line 3	T31	T32	T33	T34	T35	T36	T37	T38	T39	T40	T41	T42	T43	T44	T45	∑CT
WS7	10''	5''	5''	15''	20''	5''										60''
WS8							15''	15''	15''	5''	10''					60''
WS9												10''	5''	5''	10''	30''

Table 13 – Solution for the merge finish variant, after step 2

The chosen workstations to be merged were the first combination tested, W3 and W6, as all combinations would provide the same result. Each line maintains the same number of workstations, but the worker shares the cycle between the two (Table 14).

Workstation	Worker
WS1	Worker 1
WS2	Worker 2
WS3	**Worker 3**
WS4	Worker 4
WS5	Worker 5
WS6	**Worker 3**
WS7	Worker 7
WS8	Worker 8
WS9	Worker 9

Table 14 – Workers assigned to workstations W1 to W9, after step 2

Step 3 (optional) – Apply the Balance variant on the line(s) that did not suffer changes.

In the lines of the assembly system where no modifications were performed after the initial individual balancing using the minimization objective, the line can be balanced once again using the balance objective. This option can be optional, as the only requirements in terms of the *DSS* is the additional computational time.

The final balancing of the assembly system with three parallel assembly lines would look like the solution presented in Table 15, and the worker assignment to the workstations is presented in Table 14.

Line 1	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''	20''	5''										*60''*
WS2							15''	15''	15''	5''	10''					*60''*
WS3												10''	5''	5''	10''	*30''*

Line 2	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	***∑CT***
WS4	10''	5''	5''	15''	20''	5''										*60''*
WS5							15''	15''	15''	5''	10''					*60''*
WS6												10''	5''	5''	10''	*30''*

Line 3	T31	T32	T33	T34	T35	T36	T37	T38	T39	T40	T41	T42	T43	T44	T45	***∑CT***
WS7	10''	5''	5''	15''	20''											*55''*
WS8						5''	15''	15''	15''							*50''*
WS9										5''	10''	10''	5''	5''	10''	*45''*

Table 15 – Final solution for the assembly system with merge finish variant

3.3.2.2 Merge Start

This variant is very close to the previous one in terms of steps. Firstly, the lines are individually balanced, this time with the maximization objective to free *CT* in the first worker of the line.

Then, each line is combined in order to find the scenario that optimizes the efficiency of the system. Finally, the unmerged lines can be optionally rearranged using the balance objective.

In the case of assembly system with three parallel lines presented in the previous topic, a combined solution cannot be achieved. The initial individual balancing returns the output present in Table 16.

Line 1	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''												*35''*
WS2					20''	5''	15''	15''								*55''*
WS3									15''	5''	10''	10''	5''	5''	10''	*60''*

Line 2	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''												*35''*
WS2					20''	5''	15''	15''								*55''*
WS3									15''	5''	10''	10''	5''	5''	10''	*60''*

Line 3	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	***∑CT***
WS1	10''	5''	5''	15''												*35''*
WS2					20''	5''	15''	15''								*55''*
WS3									15''	5''	10''	10''	5''	5''	10''	*60''*

Table 16 - Solution for the merge start variant, after step 1

After analyzing the result after the initial individual balancing, it is evident that it is not possible to merge the first workstations of the three parallel lines of the system. The sum of the *CT* of any two of the three workstations results in a value bigger than the allowed 60 seconds. For this reason, as it is not possible to merge any of the first workstations of the line, the *DSS* rearranges the three lines using the balance objective, as seen in Table 17.

Line 1	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	∑CT
WS1	10''	5''	5''	15''	20''											*55''*
WS2						5''	15''	15''	15''							*50''*
WS3										5''	10''	10''	5''	5''	10''	*45''*

Line 2	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	∑CT
WS4	10''	5''	5''	15''	20''											*55''*
WS5						5''	15''	15''	15''							*50''*
WS6										5''	10''	10''	5''	5''	10''	*45''*

Line 3	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	∑CT
WS7	10''	5''	5''	15''	20''											*55''*
WS8						5''	15''	15''	15''							*50''*
WS9										5''	10''	10''	5''	5''	10''	*45''*

Table 17 - Final solution for the assembly system with merge start variant

3.3.3 Other Balancing Objectives

Assembly line balancing can be achieved by selecting the objective that best suits the assembly line or system. For this reason, additional balancing objectives can be added to the *ALB DSS* in the future if the need arises once it starts being used in the field.

The ability to tune the objective functions can also be useful to the user, providing the opportunity to find the best balance for the assembly system and achieve optimal results.

3.4 *ALB DSS*

Given the challenges and significant potential for improvement in the time-consuming and labor-intensive task of balancing assembly lines, the need for a *DSS* to automate these processes became increasingly evident. Recognizing the importance of maintaining familiar practices for the intended users, efforts were made to minimize disruption while integrating this new system. To address concerns about user acceptance and ease of use, the *DSS* was developed in Excel, a platform that users are already comfortable with. It automates several daily tasks, incorporating input validation to ensure data accuracy. The input data entered in Excel is seamlessly transmitted to SolverStudio, which then uses the PuLP library to process and solve the problem. This integration allows SolverStudio to interpret the data from Excel, apply the mathematical model, and generate an optimized solution. Furthermore, Visual Basic for Applications (*VBA*) code is employed to guide the user through the various steps, offering options for both the solution method and the desired outcome properties. Finally, the results and solutions are clearly presented within Excel, ensuring that users can easily interpret and utilize the output.

This section discusses the Assembly Line Balancing *DSS* developed for Simoldes' assembly lines. Section 3.4.1 covers the mathematical formulation behind the inner workings of the *DSS* that is able to construct solutions. Section 3.4.2 goes over how the SolverStudio works and how it is capable of building solutions using PuLP. Section 3.4.3 addresses the post processing mechanism that merges workstations over several assembly lines for the parallel balancing option. Section 3.4.4 examines the user interface and how the process engineer can manipulate the *DSS* to get the desired output, that is then covered in Section 3.4.5. Finally, Section 3.4.6 explores the future of the *ALB DSS*, discussing potential improvements and additional functionalities that can enhance its overall value and practicality for users.

3.4.1 Mathematical Formulation

It was necessary to develop a mathematical formulation to structure the problem and to understand how it could be solved. Formulating the problem mathematically breaks it down into variables, constraints, and objectives, which helps in understanding the complexities and interdependencies of different components in the assembly line. Assembly line balancing often involves multiple constraints and objectives (e.g., task precedence, station capacities). A mathematical formulation can handle these complexities systematically.

The mathematical model developed is an adaptation of existing models in the literature, such as the one established by Gökçen et al. (2006). However, modifications were necessary to adapt the model to this specific problem. Notably, an objective function for the balance variant was created, as none of the reviewed models, to our knowledge, included one.

The proposed mathematical formulation works under the following assumptions:

1) Single product – in cases where the product has different versions, the *ALB* should be conducted for the most complex version, while the simpler versions simply skip any unnecessary assembly steps.

2) Same planned *CT* for all the assembly lines in the assembly system.

3) Grouped task don't exist, so they'll be treated as individual. If more than one task needs to be performed by a single worker, they should be introduced as a single task that combines the others.

Before any mathematical formulation it is also important to develop an established and well based notation. This notation could be adopted from other authors in order to standardize it across the literature, however, as each mathematical formulation attacks the problem with its unique approach, it is hard to have a global notation that works equally well with all the mathematical formulations proposed.

For this reason, below is the notation for the assembly line balancing problem.

pCT	planned Cycle Time for all the assembly system
O	set of operators
o	operator, $o = 1, \dots, O$
T	set of tasks
t	task, $t = 1, \dots, T$
d_t	duration of task t
S	set of all precedence relations
$(r, s) \in S$	a precedence relation between two tasks, in which task s is a successor of task r
$w1, w2$	objective weights, where w_1 is the most important

The decision variables for the model are as follows.

x_{to}	1, if task t is assigned to worker w; 0, otherwise
y_o	1, if worker w has at least one task assigned; 0, otherwise
mCT	Maximum CT value among all the workers of an assembly line

As the *DSS* is capable of solving the same instance with several different objectives, the mathematical formulation contains variations for each one of these objectives. These are presented as follows.

$$\text{Minimize} \quad \sum_{o=1}^{WO} \sum_{t=1}^{T} x_{to} \cdot o \tag{1.1}$$

$$\text{Maximize} \quad \sum_{o=1}^{O} \sum_{t=1}^{T} x_{to} \cdot o \tag{1.2}$$

$$\text{Minimize} \quad w_1 \cdot \left(\sum_{o=1}^{O} y_w \right) + w_2 \cdot mCT \tag{1.3}$$

Objective 1.1 belongs to the Minimization variant and forces the entire task allocation to the beginning of the line. This objective tries to fill up the entire *CT* of the first worker, as the length of the task is multiplied by the index of the worker it has been assigned to.

Objective 1.2 belongs to the Maximization variant and takes a very similar approach to the previous one, maximizing instead of minimizing.

Objective 1.3 belongs to the Balance variant. This is a multi-objective function, primarily aiming to minimize the number of workers assigned at least one task. The secondary goal is to minimize the *mCT*, ensuring that the workers' cycle times are balanced across the assembly line. This approach naturally forces the cycle times to converge, as reducing one worker's *CT* would result in an increase for another.

To manage these competing objectives, the lexicographic method, a well-established technique in multi-objective linear programming (*MOLP*), was applied. This method prioritizes the objectives in a specific order, first optimizing the most important objective, then, after fixing its value, proceeding to optimize the next in importance. The process continues iteratively for each objective.

When multiple objectives need to be optimized simultaneously, *MOLP* ranks them according to priority. The optimization begins with the highest-priority objective, and once a solution is found, the focus shifts to the next objective, repeating the process until all objectives are addressed in the specified order.

In this particular case, it is possible to simplify the lexicographic approach by combining the two objectives using the weighted sum method. By assigning appropriate weights, *w1* and *w2*, to each objective, we ensure that the solution respects their order of importance. These weights are defined based on the expected magnitudes of the objective values, allowing the prioritization to be maintained while optimizing both objectives simultaneously. The formal definition of these weights is presented below.

$$Upper\ Bound\left(\sum_{o=1}^{O} y_o\right) = O$$

$$Upper\ Bound\ (mCT) = pCT$$

$$w_1 = pCT;\ w_2 = 1$$

With this approach, the solution process can be more efficient when solving only one combined objective.

The mathematical formulation is also subject to a set of constraints presented below.

Subject to:

$$\sum_{w=1}^{W} x_{tw} = 1, \qquad \forall t \in T \tag{2}$$

$$\sum_{w=1}^{W} x_{rw} \cdot w \leq \sum_{w=1}^{W} x_{sw} \cdot w, \qquad \forall (r,s) \in S \tag{3}$$

$$\sum_{t=1}^{T} x_{tw} \cdot d_t \leq pCT, \qquad \forall w \in W \tag{4}$$

$$mCT \geq \sum_{t=1}^{T} x_{tw} \cdot d_t, \qquad \forall w \in W \tag{5}$$

$$M \cdot y_w \geq \sum_{t=1}^{T} x_{tw} \cdot d_t, \qquad \forall w \in W, \qquad \forall t \in T \tag{6}$$

Restriction (2) guarantees that each task is assigned to exactly one worker. Restriction (3) enforces precedences relations, making sure that the precedent task is assigned to a worker with lower or equal index. To guarantee the planned *CT* is not exceeded by any worker, restriction (4) is required. Restrictions (5) and (6) are only required for the Balance variant. The first forces *mCT* to take the *CT* value of the worker with the smallest idle time and the second activates workers with at least one task assigned.

3.4.2 SolverStudio with PuLP

SolverStudio is an Excel add-in that enables the development of optimization models using various languages, such as PuLP. It combines the power of these modeling languages with the versatility, familiarity, and user-friendliness of Excel (Mason AJ, 2013).

A key aspect of SolverStudio's integration with Excel is the seamless, essentially "invisible" transfer of data between them. The mathematical model developed within SolverStudio seamlessly draws data from Excel worksheets, using defined ranges to specify the locations of each input parameter. Once the model is constructed in SolverStudio's text editor pane, it takes

over to find a solution. The results are then automatically returned to the Excel worksheet, highlighting the smooth integration between the two platforms, and demonstrating how easy and beneficial it is to alternate between them (Mason AJ, 2013).

PuLP addresses linear programming models by working with their mathematical formulation. These models consist of a set of linear relationships, typically comprising an objective function, which is either maximized or minimized, a set of constraints that the solution must satisfy to be feasible, and decision variables that define the choices available to the solver. PuLP then proceeds with a series of steps, such as translating the model into a format that the solver can process and systematically exploring points within the feasible region of solutions until it identifies the one that best meets the model's objectives (Tsurkov, 2001).

A contributing factor in the decision to choose SolverStudio with PuLP was that the installation and setup can typically be completed without requiring administrator privileges (Mason AJ, 2013). This ease of installation makes it possible to prepare all the computers of the intended users and helps ensure a smooth, problem-free setup.

3.4.3 Workstation Merger Algorithm

To achieve the capability of merging parallel assembly lines, and creating shared workers that perform tasks in workstations spread among two or more lines, it was necessary to develop a purpose-built mechanism. Taking a similar approach to the *ALB* itself, this assignment problem was mapped through a mathematical formulation that was then coded into SolverStudio and ran alongside the balancing algorithm. The literature review did not identify any existing methods suitable for addressing this specific assignment. As a result, a custom model was developed to meet the unique requirements of the problem. This tailored approach ensures that the model directly addresses the specific challenges presented, given the lack of readily available solutions in the existing body of research.

In this context, a shared worker is a worker that performs tasks in multiple workstations across more than one assembly line. The use of a shared worker in substitution of regular workers in the assembly system can happen when the combined *CT* of these workers does not exceed the planned *CT*.

The merger algorithm is employed only when parallel *ALB* is being conducted and always begins with a base solution. If the user selects the option to merge workers at the start of the line, the baseline is the assembly lines balanced individually using the backward loading variant. Conversely, if the user opts to merge at the end of the line, the forward loading variant is used. This approach is required by the structure of Simoldes' assembly lines, where the only feasible locations for merging workstations are typically at the extremities of the lines due to layout constraints. To prepare the model area, the workstation with the longest idle time on each assembly line is selected. These workstations serve as the first workstations for the merge start variant and as the last workstations for the merge finish variant.

An additional feature of the DSS is the ability to rebalance lines that have no shared workers assigned after the merger algorithm is applied. This adds significant value, as after each assembly line is individually balanced (either forward or backward loaded depending on the user's selection) some lines may still require adjustment. The rebalancing functionality re-runs the ALB model for these lines, ensuring the entire system is optimized for efficiency and better reflects the conditions of real-world assembly systems.

The notation for the *ALB* mathematical formulation has been extended to accommodate this new problem, as seen below.

pCT	planned Cycle Time for all the assembly system
SW	set of shared workers
s	shared worker, $s = 1, \dots, SW$
WS	set of workstations
p	workstation, $p = 1, \dots, WS$
d_p	total duration of tasks performed at workstation p

The decision variable for the model is as follows.

z_{ps}	1, if workstation p is assigned to shared worker s; 0, otherwise

The objective of the model is to maximize the number of shared workers by minimizing the product of the task durations by the index of the worker they are assigned to. This is presented below.

$$\text{Minimize} \quad \sum_{s=1}^{SW} \sum_{p=1}^{WS} z_{ps} \cdot s \tag{1}$$

The set of constraints is presented below.

Subject to:

$$\sum_{s=1}^{SW} z_{ps} = 1, \qquad \forall p \in WS \tag{2}$$

$$\sum_{p=1}^{WS} z_{ps} \cdot d_p \leq pCT, \qquad \forall s \in SW \tag{3}$$

Restriction (2) guarantees that each workstation is assigned to exactly one shared worker. Restriction (3) enforces the planned *CT* limit for all the shared workers.

3.4.4 User Interface in Excel with *VBA*

The primary goal of the developed *DSS* was to create a practical instrument that the process engineering team could effectively use, rather than just a program capable of assembly line balancing but not practical for everyday application. Therefore, Excel and its functionalities were chosen as the foundation for the user interface and data validation. Excel is not only extremely versatile but is also the tool predominantly used for most tasks in the department, making convenience and familiarity key factors in this decision.

The user interface developed for the *ALB DSS* effectively utilizes Excel's features and integrates *VBA* to facilitate communication with the user and automate the initiation of the mathematical model implemented in SolverStudio. Data input for assembly conditions (such as production targets, *OEE*, operating time, and more), the task list, and the machine list are also managed through cells, as handling the high volume of data in any other way would be challenging, given *VBA*'s limited functionalities and features.

The *ALB DSS* opens to the main worksheet of the program, where all the information of the assembly system is to be introduced in the '`Input Window`' section of the page, as Figure 12 shows.

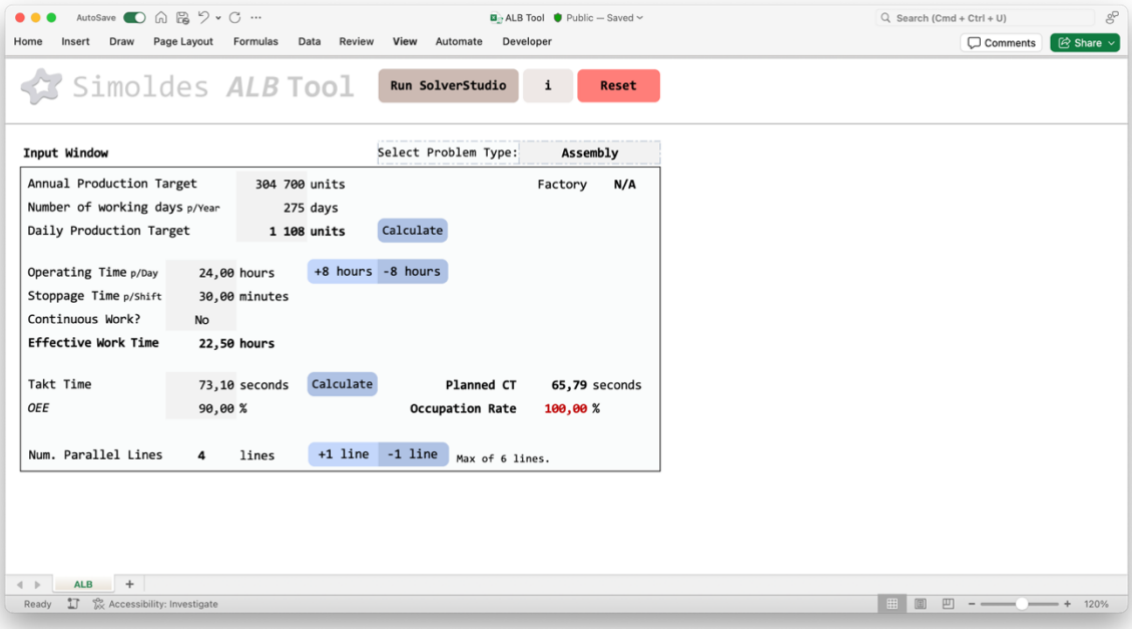


Figure 12 – Input Window (*ALB DSS*)

This section requires the user to fill in key information about the assembly system. Firstly, the daily production target can be introduced manually, or it can be achieved by introducing the annual production target, the number of working days per year, and then pressing the associated 'Calculate' button.

Information about the available work time is also entered in this section. The operating time is defined in hours, which can be manually entered or adjusted using the '+8 hours' and '-8 hours' buttons, which add or remove the equivalent of a work shift, respectively. Information about stoppage time in minutes can also be added, along with an indication of whether the work is continuous. With this information, the effective work time can be calculated and displayed in either hours or minutes. To switch between units, the desired one can be selected by clicking on the cell displaying the unit being utilized at the moment.

The Takt Time can be introduced manually or calculated automatically by pressing the associated 'Calculate' button, which calculates this value by using the daily production target and effective work time introduced previously. The *OEE* is also introduced in this section, which then allows the calculation of the planned *CT* for the assembly system and the occupation rate.

The production conditions section is finalized with the introduction of the number of parallel lines in the system. This number can be modified by the '+1 line' and '-1 line' buttons, which adds and removes one assembly line, respectively. This number cannot be edited manually because the use of the buttons also makes available the tasks list for the other parallel lines or removes them if the line is also removed. This value can range from one to a maximum of six parallel lines.

Before advancing further in the program by pressing the 'Run SolverStudio' button in the top ribbon, a set of data validations is performed to ensure that all values are in the correct format and within the allowed ranges. The validations check the daily production target, the operating and stoppage time, the planned *CT*, and the number of lines.

An information window is also available to the user by pressing the 'i' button, which displays a set of instructions on how to use the *DSS* and an overview of the required steps to achieve a solution (Figure 13).

In this screen, there are three additional options available. These are, from left to right, the possibility of populating the main screen with an example instance that has production conditions and task lists already available for easy testing of the *ALB DSS* functionalities, the shortcut to the Task Library, and to the Machine Library. These two libraries are mentioned later.

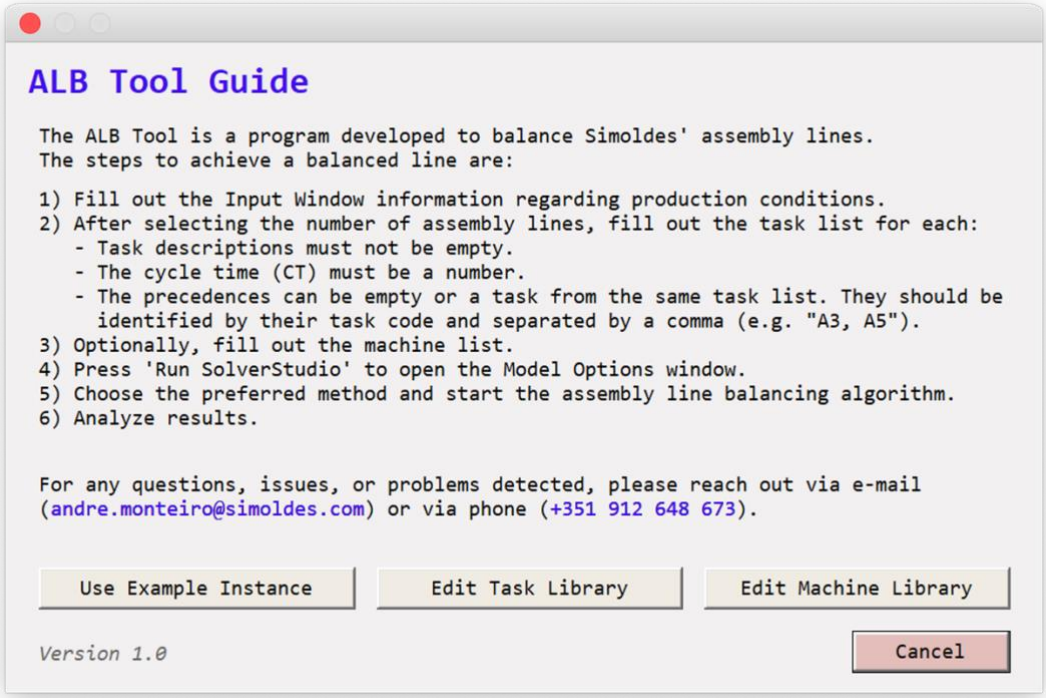


Figure 13 – Information screen (*ALB DSS*)

The 'Reset' button, also available at the top ribbon of the *DSS*, cleans up the entire area of input data. Not only for the production conditions sections, but also for the task list and machine list. The user is warned about the potential of lost progress before the action is executed.

The next section is dedicated to the task lists. The user is presented with a table for each of the open assembly lines in the assembly system. Figure 14 displays the case of two assembly lines.

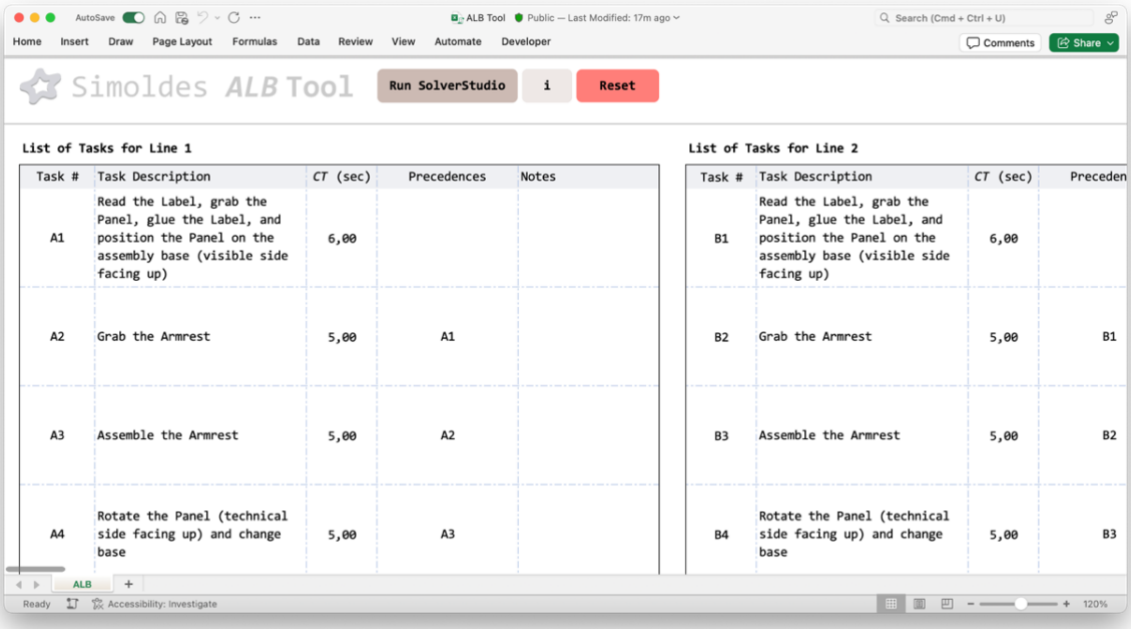


Figure 14 – Task List (*ALB DSS*)

This section requires the user to provide information about each task to be performed in every active assembly line within the system. The information required for each task is the description, the task time, and the precedent task. There is also an optional notes field. As the mentioned information is introduced, the task automatically receives a code, presented as the 'Task #'. This is the code that should be used when introducing the precedences for a task.

As mentioned previously, before starting the program, all the data introduced is checked. For a task, the description must be a text and not empty, the task time must be a positive number, the precedences must be the task code of the precedent task, and separated by a comma and a space in case there is more than one. An additional validation involves checking whether the *CT* for tasks does not exceed the planned *CT*.

To add a new task after all the available spaces have been used, the user can press the '+ Add New Task' button to reveal another line. An empty line in the task list can also be hidden by the 'Hide Line' button in case it is empty. There is also an indication in the bottom right corner of how many tasks have been introduced to the assembly line.

The task list is also equipped with the feature of a task library, where the user can search for common tasks in assembly lines, such as mounting certain standard clips or foams, and introduce them quickly in the task list, saving time and effort. This library can be opened by pressing the 'Task Library' button. Each assembly line has its own button, but they all display a similar screen, such as the one displayed in Figure 15.

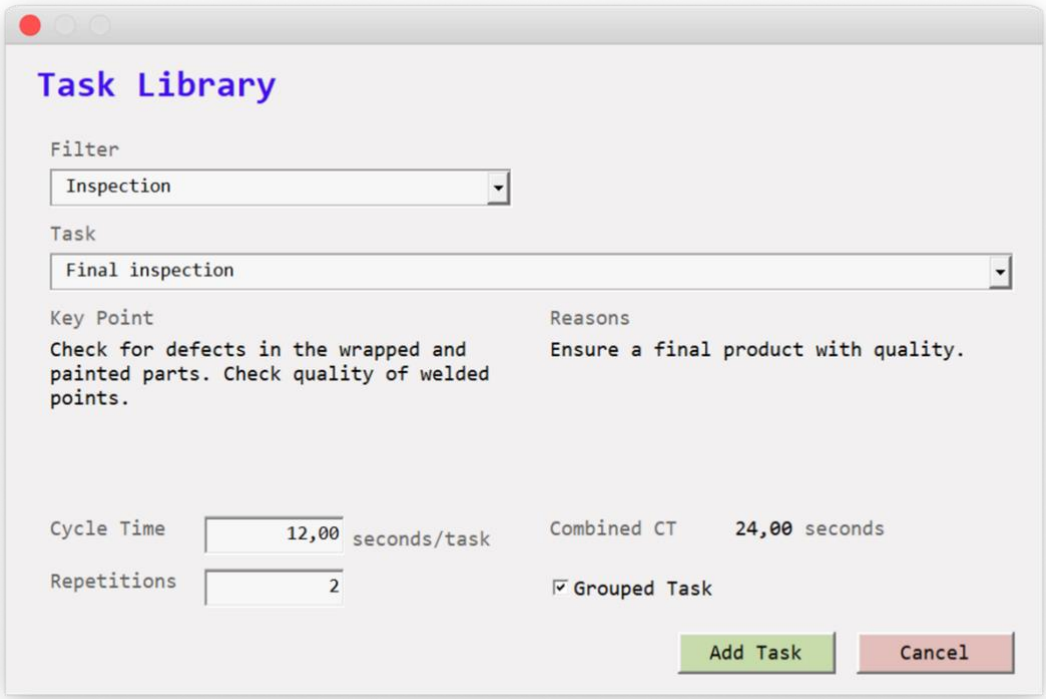


Figure 15 – Task Library screen (*ALB DSS*)

The search for a task can be performed using filters, allowing the user to search for tasks ranging from small component assembly to screwing and final inspections. After selecting a filter, which is optional, the task selection list is populated with matching tasks. When a task is chosen, the corresponding fields are automatically filled with the relevant information. The default number of repetitions is set to one, and the task time is automatically populated but can be manually adjusted to better reflect the desired task duration.

The grouped task option can be selected for two or more repetitions of the same task, allowing the user to introduce all repetitions as a single entry, rather than listing the same task multiple times. Once the user is satisfied with the task details, they can add it to the task list by pressing 'Add Task'. Before this action is executed, a set of data validations is performed to ensure the correct values are entered. These validations ensure that the task time or combined task time is a numeric value and not empty. Similarly, the number of repetitions must be a number greater than or equal to one for ungrouped tasks and greater than or equal to two for grouped tasks.

In the first version of the *ALB DSS*, the introduction of machine information does not impact the balancing of the assembly system. However, this feature intends to gather additional information about potential bottlenecks in the assembly lines and the required operating times for these machines. Although this feature is purely informational, this section has been

designed and functions in a manner similar to the task list. This section can be seen in Figure 16.

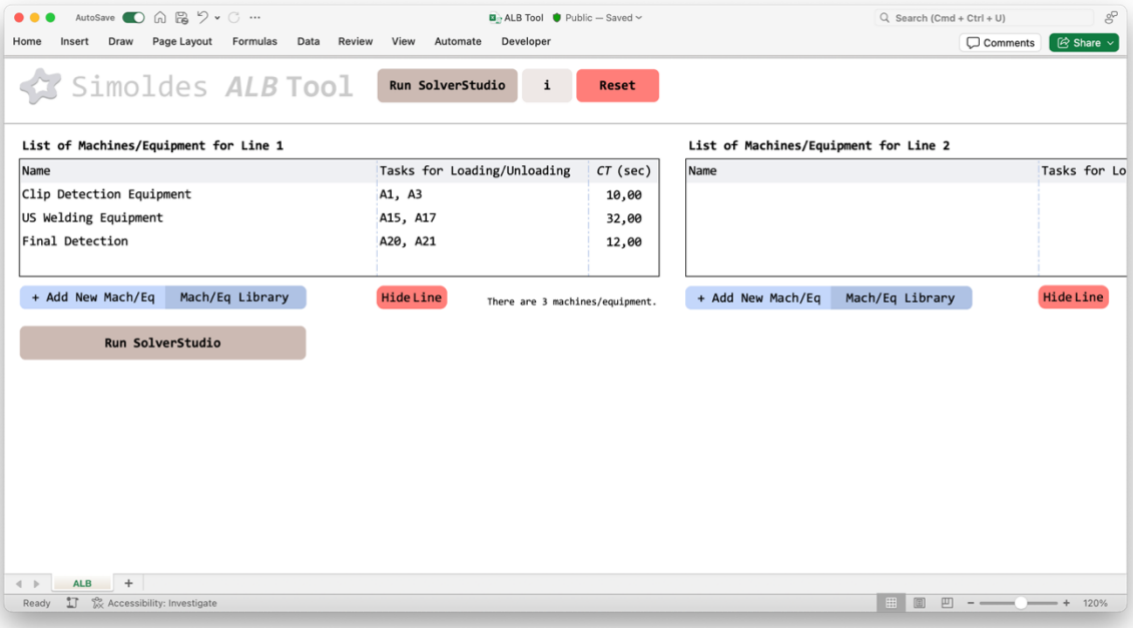


Figure 16 – Machine/Equipment List (*ALB DSS*)

The information required is the name or designation of the equipment, the task codes that are required for the loading and unloading of the equipment and the *CT* in seconds of its operation time.

To add another machine to the assembly line, the user can press the '`+ Add New Mach/Eq`' button. To access the machine library, the user should press the '`Mach/Eq Library`' button, shown in Figure 17. To remove an empty line, the user can press the '`Hide Line`' button. At the bottom right corner there is an indication of how many machines have been added to the line.

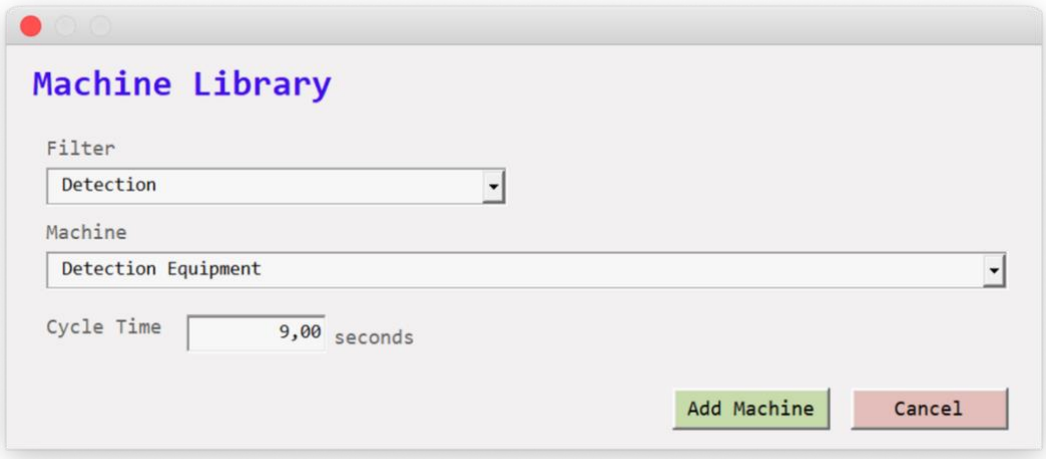


Figure 17 – Machine Library screen (*ALB DSS*)

Once again, the machine library works in a very similar manner to the task library. The objective is a facilitated insertion of standard equipment to the machine list. The user can select a filter to narrow the search, such as a welding machine, wrapping machine, or detection equipment. After this, the machine selection list is populated with the machines and equipment that match the filter. After one has been selected, the determined time for that machine is automatically inserted, although it can be adjusted by the user. Finally, to add the machine to the list, the user only needs to press the 'Add Machine' button, regarding the machine time is valid.

When all the information is introduced into the task and machine list, the *ALB DSS* is able to perform the automatic line balancing. To start the program, the user should press the 'Run SolverStudio' button. This action validates once again all the inputs and ensure no incorrect values, empty lines in the task lists, invalid inputs, or empty assembly lines. If everything is correct, the program displays the screen shown in Figure 18.

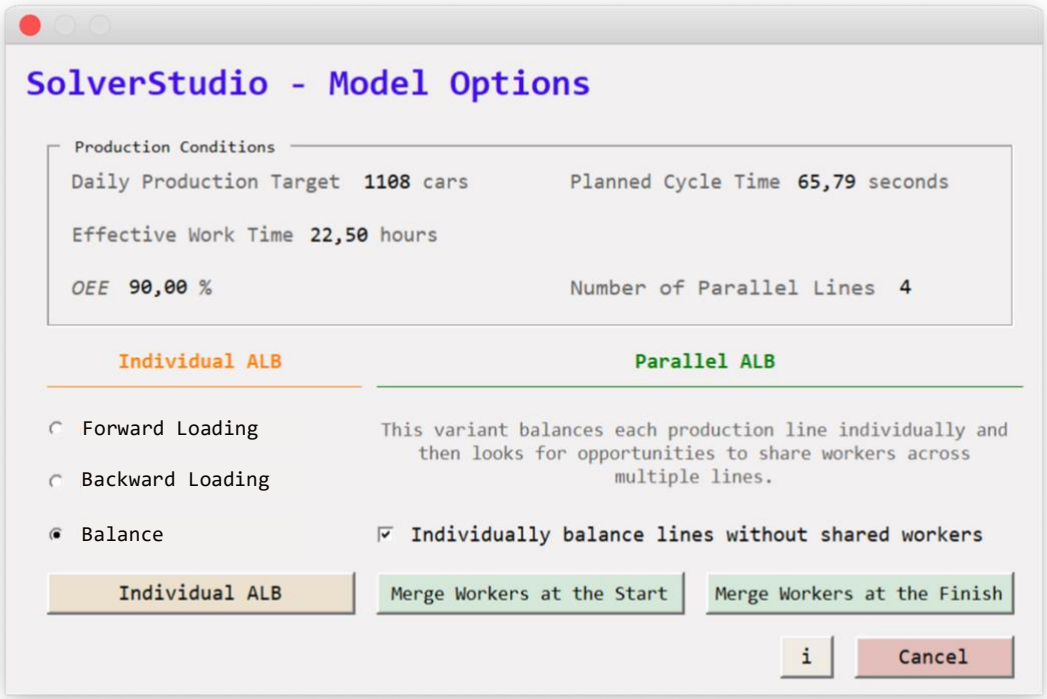


Figure 18 – Model Options screen (*ALB DSS*)

Before running the mathematical model to find a solution for the assembly system, the user is presented with a summary of the assembly conditions and the options for assembly line balancing. For individual balancing, the user can choose between forward loading, backward loading, or balance, with the latter being the default selection. For parallel assembly line balancing, the user can choose to either merge at the start or at the end of the assembly line.

The final step is to press the button with the desired objective to initiate the model.

To edit the Task and Machine library, the user should open the information window available by pressing the '`i`' button in the top ribbon of the *ALB DSS* and choose the desired one. After this, the user is forwarded to the Task Library sheet (Figure 19) or to the Machine Library sheet (Figure 20). After arriving at these sheets, the libraries can be manually edited, adding, or removing items from them. All the fields are open, except the filter which is restricted to a predetermined selection.

The user can then return to the main screen by pressing the '`Return`' button.

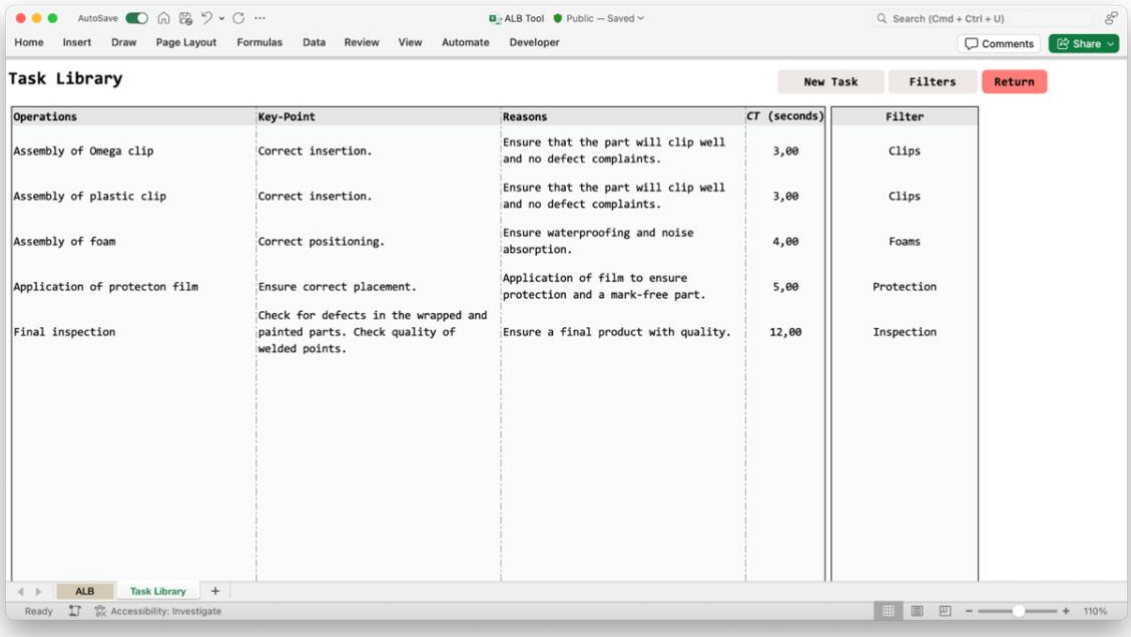


Figure 19 – Task Library sheet (*ALB DSS*)

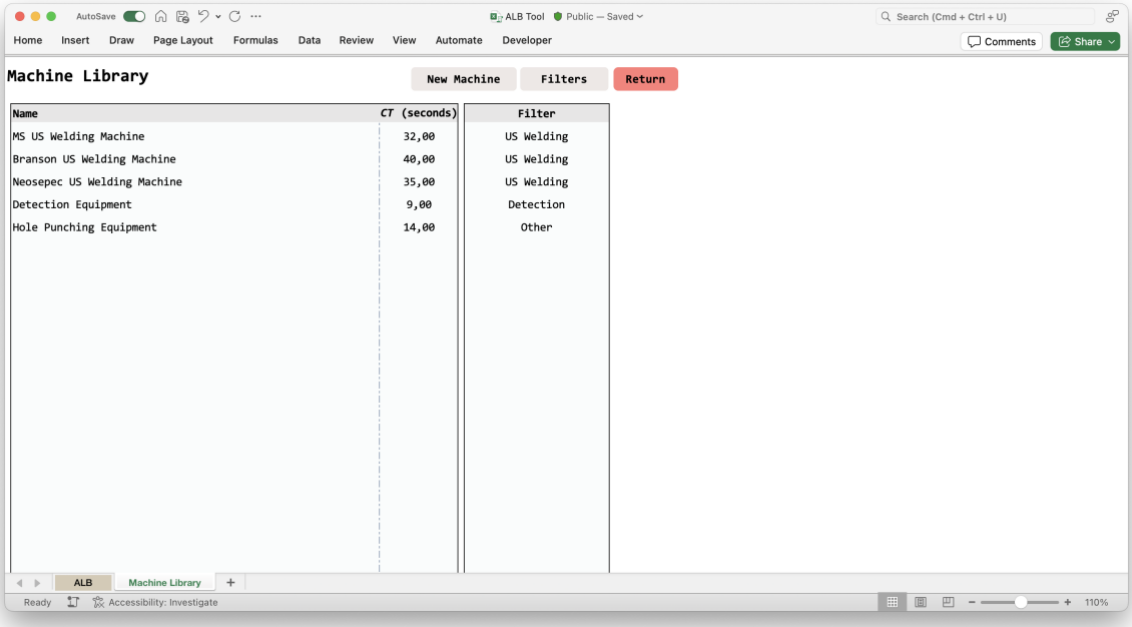


Figure 20 – Machine Library sheet (*ALB DSS*)

3.4.5 *DSS* Outputs

After performing the *ALB* operation, the user is forwarded to the output sheet of the *DSS*, where the solution for the assembly system, regarding the inputs and objectives chosen, is presented. This sheet, as seen in Figure 21, starts by displaying the production conditions, mirroring the main worksheet of the *DSS*. It is also possible to check the variant run by SolverStudio, representing the objective chosen for the algorithm.

Immediately after this, the user can consult the '`Worker Assignment`' table, which indicates, for each assembly line in the assembly system, the workers required identified by their codes and the presence of shared workers, performing tasks in more than one assembly line. To the right of each worker is the cumulative *CT* for all the tasks they perform in said assembly line.

The last information presented as a result of the *ALB* is the task list introduced by the user, this time assigned to the workers who performs each task. These lists are organized by worker and not by task sequence, so all the tasks assigned to one worker appear together (Figure 22).

At the top ribbon of this sheet, the user also has available the option to reset the *DSS*, returning to initial conditions and deleting all the information inputted, or to return to the main sheet, in case it is desired to introduce some changes and test out a different scenario or objective.

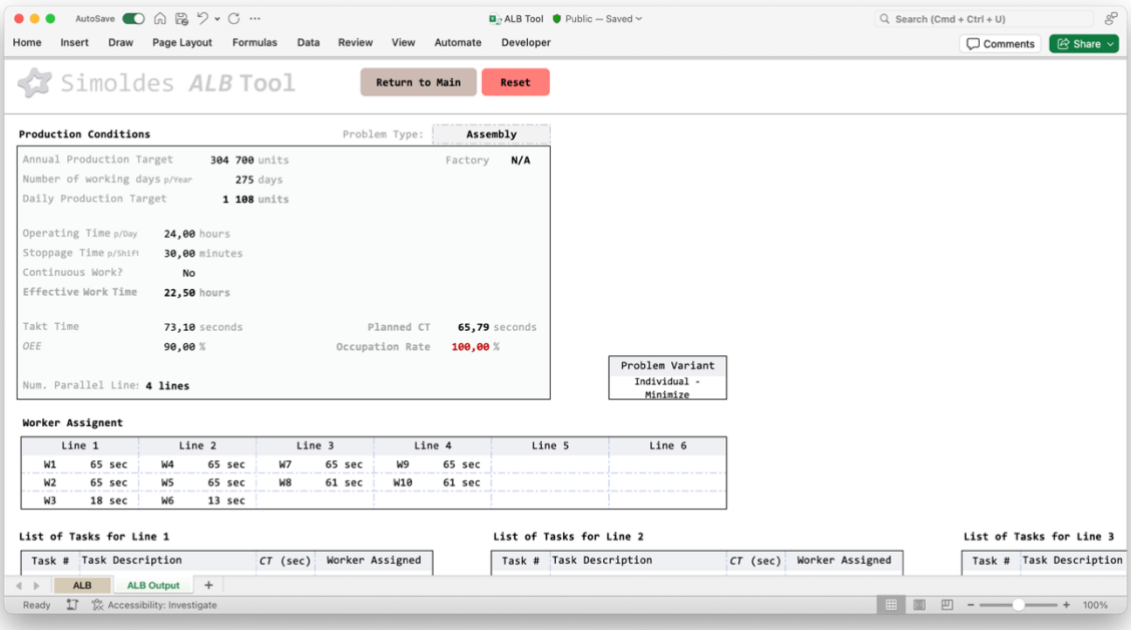


Figure 21 – *ALB* Output, 1 (*ALB DSS*)

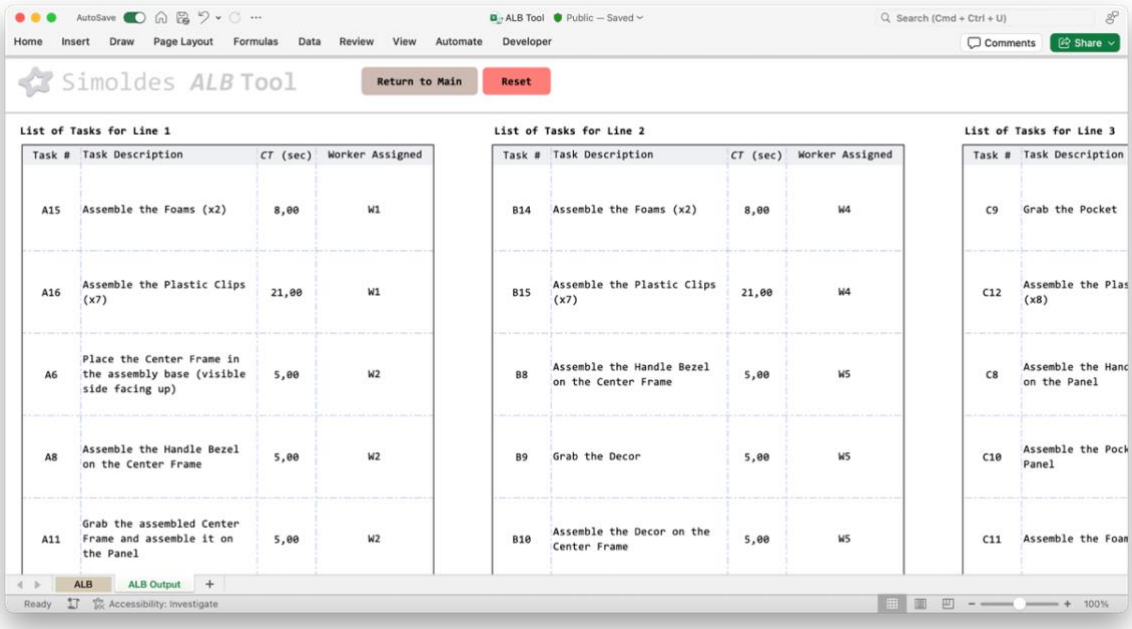


Figure 22 – *ALB* Output, 2 (*ALB DSS*)

4 Results

This section aims to apply the *ALB DSS* in a practical setting to evaluate its performance and the benefits it offers to both users and the organization. The instance used for testing the *DSS'* functionalities and outcomes is based on the example introduced in Section 3.1.1, which represents the assembly system responsible for producing a set of four door panels. The front panels are more complex than the rear ones, with the front left door panel requiring an additional task. The rear door panels are mirrored, meaning they likely result in similar task assignments for workers on their respective lines. The production target is 1108 cars per day, demanding a planned *CT* of 65,79 seconds, considering an *OEE* of 90% and an effective working time of 22,5 hours per day.

The ALB *DSS* was tested using each objective it provides to the user, allowing for a comparison of the results, practicality, and relevance of the solutions presented.

The first variant tested was the minimization with individual balancing. The results, presented in Table 18, show that the assembly system requires a total of ten workers (three for each of the front panels and two for each of the rear ones). As this variant aims to shift task assignments towards the start of the lines, reducing the *CT* for the last worker, it can be observed that the last worker on each line is consistently the least occupied. The difference in workload is significant for Lines 1 and 2, but less so for Lines 3 and 4. This is because the goal of minimizing the number of workers forced the last worker on Lines 3 and 4 to take on as many tasks as possible, thereby avoiding the need for a third worker on those lines. However, for Lines 1 and 2, the cycle time disparity is noticeable, meaning the last worker on those lines still has considerable available *CT* to either perform additional tasks on the assembly line or assist other elements of the assembly system.

Regarding task distribution for the assembly system under the minimization variant (also shown in Table 18), it is clear that the model strategically used precedence constraints to optimize the result, rather than following a strict sequential order of tasks. This approach respects the precedence rules but explores them more effectively than a manual balancing process typically would.

The time required to run this variant is approximately 19 seconds. During this time, the *ALB DSS* populates the model area for each assembly line and runs the SolverStudio model, repeating the process four times. Once this step is completed, the *DSS* decodes the results and displays them on the Output sheet.

Line 1		Line 2		Line 3		Line 4	
W1	65 sec	W4	65 sec	W7	65 sec	W9	65 sec
W2	63 sec	W5	58 sec	W8	61 sec	W10	61 sec
W3	20 sec	W6	20 sec				

W1	A1, A2, A3, A4, A5, A6, A9, A15, A16
W2	A7, A8, A10, A11, A12, A13, A14, A17, A18, A19, A20, A21, A22
W3	A23, A24, A25
W4	B1, B2, B3, B4, B5, B7, B9, B14, B15
W5	B6, B8, B10, B11, B12, B13, B16, B17, B18, B19, B20, B21
W6	B22, B23, B24
W7	C1, C2, C3, C4, C5, C6, C7, C9, C12
W8	C8, C10, C11, C13, C14, C15, C16, C17, C18, C19, C20
W9	D1, D2, D3, D4, D5, D6, D7, D9, D12
W10	D8, D10, D11, D13, D14, D15, D16, D17, D18, D19, D20

Table 18 – Results of the *ALB DSS* for forward loading variant

The maximization variant follows a similar approach to the previous one but yields opposite results, freeing *CT* for the first workers in the assembly line (Table 19). The number of required workers remains at ten, with the same distribution across the lines. However, while in the previous example the idle time for the last workers of Lines 1 and 2 was identical, this is not the case in the maximization variant. This difference is likely due to the precedence constraints, which, in this variant, result in W6 having more idle time than W4 in the previous case. Lines 3 and 4, on the other hand, achieve a balance in *CT*.

Once again, the task distribution illustrates that the assignment is driven by the optimization of precedence constraints to achieve the desired outcome.

The computational time for this variant is also approximately 18 seconds.

Line 1		Line 2		Line 3		Line 4	
W1	21 sec	W4	16 sec	W7	63 sec	W9	63 sec
W2	64 sec	W5	64 sec	W8	63 sec	W10	63 sec
W3	63 sec	W6	63 sec				

W1	A1, A2, A3, A4
W2	A5, A6, A7, A8, A9, A10, A11, A15, A16
W3	A12, A13, A14, A17, A18, A19, A20, A21, A22, A23, A24, A25
W4	B1, B2, B3
W5	B4, B5, B6, B7, B8, B9, B10, B14, B15
W6	B11, B12, B13, B16, B17, B18, B19, B20, B21, B22, B23, B24
W7	C1, C2, C3, C4, C5, C6, C11, C12
W8	C7, C8, C9, C10, C13, C14, C15, C16, C17, C18, C19, C20
W9	D1, D2, D3, D4, D5, D6, D11, D12
W10	D7, D8, D9, D10, D13, D14, D15, D16, D17, D18, D19, D20

Table 19 – Results of the *ALB DSS* for backward loading variant

The balance variant may be the most useful option in the *ALB DSS* due to its broad applicability and relevance to nearly all assembly lines. This variant attempts to minimize the number of workers required while also reducing the *CT* differences between workers on the same line, as shown in Table 20. In this case, the assembly system still requires ten workers. For Line 1, the *CT* difference is only 2 seconds, while for Line 2 it is 1 second. Lines 3 and 4 are perfectly balanced, with no *CT* differences between the two workers.

The computational time for this variant is approximately 100 seconds.

Line 1		Line 2		Line 3		Line 4	
W1	50 sec	W4	47 sec	W7	63 sec	W9	63 sec
W2	50 sec	W5	48 sec	W8	63 sec	W10	63 sec
W3	48 sec	W6	48 sec				

W1	A1, A2, A3, A4, A15, A16
W2	A5, A6, A7, A8, A9, A10, A11, A12, A13, A14
W3	A17, A18, A19, A20, A21, A22, A23, A24, A25
W4	B1, B2, B3, B4, B9, B15
W5	B5, B6, B7, B8, B10, B11, B12, B13, B14
W6	B16, B17, B18, B19, B20, B21, B22, B23, B24
W7	C1, C2, C3, C4, C5, C6, C11, C12
W8	C7, C8, C9, C10, C13, C14, C15, C16, C17, C18, C19, C20
W9	D1, D2, D3, D4, D5, D6, D11, D12
W10	D7, D8, D9, D10, D13, D14, D15, D16, D17, D18, D19, D20

Table 20 - Results of the *ALB DSS* for Balance variant

After analyzing the results of all individual variants, the focus shifts to the parallel variants.

The merge start variant first backward loads all lines to create idle time for the first worker of each line. It then explores the possibility of introducing a shared worker capable of performing tasks for two workstations on parallel lines, as demonstrated by worker SW1 in Lines 1 and 2 (Table 21). Following this, the model re-evaluates lines without shared workers, such as Lines 3 and 4, using the balance variant to optimize the assembly system, though this step is optional.

SW1 performs tasks for both Line 1 and Line 2, accumulating 21 seconds from Line 1 and 16 seconds from Line 2. This results in a total *CT* of 37 seconds for SW1, with approximately 28 seconds of idle time available for movement between the two workstations. With this adjustment, the assembly system requires only nine workers, saving one compared to the individual variants.

Task assignments remain similar to previous examples, with the notable addition of SW1 performing tasks for both lines.

The computational time for this variant is approximately 50 seconds if rebalancing lines without shared workers is included, and about 18 seconds if it is not. The process begins with balancing each line individually using either the minimization or maximization objective, similar to the individual variants. Next, the model assesses the possibility of merging workstations with shared workers. If the option is selected, the *DSS* then rebalances lines without shared workers using the balance variant, and the final results are presented to the user.

This solution also provides insights for factory layout. Since Lines 1 and 2 are connected through a shared worker performing initial tasks for each line, it is crucial that they are positioned close together. Adequate walking space between workstations and a minimal distance are necessary. This analysis must be conducted by the process engineer responsible, as the *DSS* does not have the capability to evaluate these layout considerations.

Line 1		Line 2		Line 3		Line 4	
SW1	21 sec	**SW1**	16 sec	W5	63 sec	W7	63 sec
W1	64 sec	W3	64 sec	W6	63 sec	W8	63 sec
W2	63 sec	W4	63 sec				

SW1	A1, A2, A3, A4, B1, B2, B3
W1	A5, A6, A7, A8, A9, A10, A11, A15, A16
W2	A12, A13, A14, A17, A18, A19, A20, A21, A22, A23, A24, A25
W3	B4, B5, B6, B7, B8, B9, B10, B14, B15
W4	B11, B12, B13, B16, B17, B18, B19, B20, B21, B22, B23, B24
W5	C1, C2, C3, C4, C5, C6, C11, C12
W6	C7, C8, C9, C10, C13, C14, C15, C16, C17, C18, C19, C20
W7	D1, D2, D3, D4, D5, D6, D11, D12
W8	D7, D8, D9, D10, D13, D14, D15, D16, D17, D18, D19, D20

Table 21 – Results of the *ALB DSS* for Merge Start variant

The merge finish variant, as shown in Table 22, is quite similar to the previous one. In this variant, the shared worker is positioned at the end of Lines 1 and 2, with a *CT* of 20 seconds for each line and approximately 35 seconds of idle time remaining in the cycle. Lines 3 and 4 maintain the same task distribution among their workers.

The computational time for this variant is comparable to the previous parallel variant, reaching around 50 seconds if the rebalance option is selected.

Regarding the factory layout, the conclusions are similar to those of the previous variant. However, in this case, the shared worker is located at the end of both lines. Consequently, it is essential that Lines 1 and 2 are positioned close together in the factory layout, with sufficient walking space and minimal distance between them.

Line 1		Line 2		Line 3		Line 4	
W1	65 sec	W3	65 sec	W5	63 sec	W7	63 sec
W2	63 sec	W4	58 sec	W6	63 sec	W8	63 sec
SW1	20 sec	**SW1**	20 sec				

SW1	A23, A24, A25, B22, B23, B24
W1	A1, A2, A3, A4, A5, A6, A9, A15, A16
W2	A7, A8, A10, A11, A12, A13, A14, A17, A18, A19, A20, A21, A22
W3	B1, B2, B3, B4, B5, B7, B9, B14, B15
W4	B6, B8, B10, B11, B12, B13, B16, B17, B18, B19, B20, B21
W5	C1, C2, C3, C4, C5, C6, C11, C12
W6	C7, C8, C9, C10, C13, C14, C15, C16, C17, C18, C19, C20
W7	D1, D2, D3, D4, D5, D6, D11, D12
W8	D7, D8, D9, D10, D13, D14, D15, D16, D17, D18, D19, D20

Table 22 – Results of the *ALB DSS* for Merge Finish variant

4.1 Comparison with Manual *ALB*

To compare the *ALB DSS* with manual balancing performed by a process engineer at Simoldes, the same example was tested using the balance objective for individual lines.

For the manual balancing, the method described in Section 3.2 was employed. This method involves using a table that automatically sums all tasks assigned to each worker. Tasks are then manually redistributed among workers to find an assignment that optimizes resource use, adheres to precedence constraints, and respects the planned *CT* limit. The results of the manual balancing for individual lines applying the balance objective are presented in Table 23.

Line 1		Line 2		Line 3		Line 4	
W1	49 sec	W4	49 sec	W7	63 sec	W9	63 sec
W2	51 sec	W5	46 sec	W8	63 sec	W10	63 sec
W3	48 sec	W6	48 sec				

W1	A1, A2, A3, A4, A5, A6, A7, A8, A15
W2	A9, A10, A11, A12, A13, A14, A16
W3	A17, A18, A19, A20, A21, A22, A23, A24, A25
W4	B1, B2, B3, B4, B5, B6, B7, B8, B14
W5	B9, B10, B11, B12, B13, B15
W6	B16, B17, B18, B19, B20, B21, B22, B23, B24
W7	C1, C2, C3, C4, C5, C6, C11, C12
W8	C7, C8, C9, C10, C13, C14, C15, C16, C17, C18, C19, C20
W9	D1, D2, D3, D4, D5, D6, D11, D12
W10	D7, D8, D9, D10, D13, D14, D15, D16, D17, D18, D19, D20

Table 23 - Results of the manual balancing using balance objective

The results achieved manually are largely similar to those produced by the *ALB DSS*, especially for Lines 3 and 4, with only minor differences in Lines 1 and 2. Despite these small variations, the assembly system still requires ten workers, the same as in the *DSS'* solution.

Balancing Lines 1 and 2 manually took approximately 20 minutes, 15 minutes for Line 1 and an additional 5 minutes for Line 2. This time was needed to manually shift tasks between workers to explore different possibilities, ensuring that the task combinations for each worker remained within the *CT* limit and respected the precedence constraints. For Line 1, the manual balancing resulted in a difference of 3 seconds between the largest and smallest *CT*, compared to the 2 second difference achieved by the *ALB DSS*. For Line 2, the manual approach delivered a result with a 3 second *CT* difference, while the *ALB DSS* produced a 1 second difference.

For Lines 3 and 4, the manual balancing produced identical results to those of the *ALB DSS*, with the same idle time for all workers and identical task assignments. Despite having fewer tasks, these lines posed a greater challenge due to the constraints of task duration, precedence, and disposition. The total task times for these lines left very little idle time for the workers, making the problem more constrained and difficult to manage. Any slight increase in one worker's *CT* would require a third worker, which was undesirable. Since Lines 3 and 4 share the same task list, the balance achieved for one could be directly applied to the other, eliminating the need for a separate balancing process. However, due to the complexity of these lines, it took almost 30 minutes to arrive at a solution.

The lengthy time required to balance what seems like simple lines with a small number of tasks can be attributed to the need to carefully consider all constraints and limits at each iteration. Every change in the balancing must be reviewed to ensure no errors are made, and infeasible scenarios often require reworking when constraints are violated. Furthermore, manual

balancing is influenced by human factors, such as overlooking feasible solutions or getting stuck on a particular approach. Taking breaks to reset and refocus is sometimes necessary to view the problem from a different perspective, but this further increases the time required to reach a solution, which is something the *ALB DSS* eliminates.

The comparison between manual balancing and using the *ALB DSS* highlights the advantages of the latter. While the results are largely similar, the *ALB DSS* requires only approximately 3 minutes for the lengthier variant to generate a solution, whereas the manual process took nearly 50 minutes. This significant time difference allows the user of the *ALB DSS* to review the results, check for mistakes, and propose improvements if needed.

The main advantage of the *ALB DSS* is not necessarily in the results it produces, as these are always better but comparable to those achieved manually, but rather in the speed and efficiency with which it generates solutions. The *DSS'* flexibility and versatility are evident, and its performance becomes even more advantageous as the complexity of the problem increases, either in the number of tasks, the complexity of precedence relations, or the number of workers required to stay within the planned *CT*. The *DSS* effectively manages this increased complexity by utilizing a linear model, which simplifies the relationships between variables and allows for efficient computation.

Another key advantage of the *ALB DSS* is its ability to quickly adjust to changes in input data. Adding or removing tasks, changing task times, or modifying precedence constraints would typically require a complete reevaluation of task allocation and possibly a full rebalancing of the assembly lines. The *DSS'* quick response time enables the user to test different scenarios in a reasonable timeframe, facilitating well-informed decisions regarding the number of workers needed, task assignments, and factory layout.

5 Conclusion

As markets evolve and constantly change, companies must adapt to these variations to remain competitive and achieve long-term success. This requires agility and the ability to respond promptly to challenges that may necessitate adjustments to pre-established plans. In this context, eliminating inefficiencies that slow down processes and optimizing repetitive, time-consuming tasks become crucial for maintaining operational excellence.

This project was developed with that objective in mind, following a proposal from Simoldes to create a *DSS* capable of automating the assembly line balancing process. The goal was to streamline decision-making for process engineers by generating optimized solutions in a short timeframe, saving valuable time and resources. The *DSS* was designed to be flexible, adapting to different production conditions, and providing support throughout all stages of the production lifecycle, from initial quotations to serial production. Recognizing this improvement opportunity led to the central question: How does the development of an *ALB DSS* accelerate the dimensioning and balancing of production lines and facilitate decision-making?

The project was initiated with the goal of developing a *DSS* that would bring significant value to its users and the organization as a whole. From the outset, it was important to create something practical and easy to implement in real-world scenarios, while also providing robust solutions. This required a balance between simplicity and complexity, ensuring the *DSS* could produce clear, actionable outputs. Additional variations of the problem were also explored, including parallel lines with shared workers, further enhancing the *DSS'* adaptability to real assembly line conditions at Simoldes.

To develop this automated *DSS* for balancing assembly lines and optimizing their layout, worker allocation, and task assignment, a linear model was constructed. This model includes a set of objective functions, which the user can select based on the desired output. Additionally, a series of constraints were defined to guide and limit the solutions produced by the model. These constraints address task assignment, precedence relationships, and cycle time limits for the

assembly system. Furthermore, a second linear model was created to manage parallel lines, assigning workstations on parallel lines to shared workers. This simpler model focused primarily on cycle time constraints. Both models were processed using SolverStudio in Excel, keeping the *DSS* integrated within a familiar platform to facilitate user adoption and minimize resistance to new functionalities.

The results demonstrate that the *ALB DSS* has significant potential to improve assembly line balancing. Not only is the computational time drastically reduced, from 50 minutes of manual balancing to just 3 minutes in the most time-consuming variant for a medium-sized problem with four parallel lines, but the *DSS* also offers deeper analysis of precedence constraints, which are difficult to manage manually. Additionally, the *DSS* reduces the probability of errors and makes it easier to modify assembly systems in response to changes in task lists, precedences, task durations, or production conditions, as the line can be rebalanced almost instantly. The ability to select from multiple objectives also allows users to test various scenarios, resulting in more informed decisions regarding task allocation and factory layouts.

This project encountered several challenges along the way, including a lack of documentation from previous projects regarding task lists, durations, and precedence constraints. As a result, much of the data used to build the models was drawn from ongoing or current projects in production. Another challenge was developing an intuitive and interactive program that would allow users to manage complex task lists and multiple balancing objectives while maintaining a user-friendly interface.

Although future improvements have already been identified, the *DSS* is now at a stage where it is fully functional and capable of providing significant benefits to the process department where it was developed. The next steps involve testing the *DSS* within the department, expanding its use across Simoldes' factories, and continuously refining its features to enhance performance.

5.1 Future of the *ALB DSS*

The *DSS* developed for this project is still in an early phase and serves primarily as a proof of concept. Its purpose is to introduce these new functionalities and methods to the process engineers who will work with it and determine if further development of the *DSS* would be beneficial to the department and the company.

If the *ALB DSS* proves to be valuable to its users and there is interest in its further development, several improvements and additional functionalities have already been identified.

The first proposal refers to a previously discussed topic. Due to the limited time available for the project, a decision was made to focus on advancing the parallel line problem rather than also addressing grouped tasks and mixed-model assembly lines. Developing solutions for these new challenges will require further research into the methods used and their integration with parallel lines.

Another proposed improvement is the inclusion of walking times and transfer times between workstations. For a worker assigned tasks on more than one line, it may be important to account for the *CT* spent while moving between workstations. These times will vary depending on the distance between workstations and the layout, which may increase movement time even for short distances. Regarding transfer times, when tasks at one workstation are completed and the tasks at the next station begin, a transfer activity typically occurs. This may be as simple as placing a door panel on a transfer base and then picking it up to resume the assembly steps. Although these tasks are simple and quick, they add to the *CT* of the workstations and are often not included in the established task list for the assembly line. Where they appear in the overall sequence also depends on the line balancing, as they are only required between workstations. To achieve this improvement, movement times between workstations could be deducted from the available *CT* of the worker who operates in two or more workstations, also considering the return time to the original position. For transfer times, a predetermined duration could be reserved in the workstation for the pickup and handoff activities.

Other improvements can be suggested not with the aim of creating more possible solutions or giving more freedom to the user, but to facilitate the *DSS'* use and enhance the overall experience. One proposal focuses on improving the machine and task library, which was already implemented in the current version of the *ALB DSS*. To further increase the productivity of the process engineers using this *DSS*, a larger selection of standard tasks (such as the time required to assemble a specific type of plastic clip) can be measured and added. The same applies to the machines and peripherals of the line. Although these are not necessary when the goal is simply balancing the assembly line, they may become useful when analyzing the assembly system as a whole. Implementing a relational database, such as Access, could also enhance the *DSS'* versatility and usability for users.

The majority of resources dedicated to developing this project were focused on how the user would input all the required information into the *DSS* and, sequentially, how it, through its various mechanisms and algorithms, would transform that input and solve it. The presentation of the solution, the information that could be gathered from it, and additional insights were not explored as deeply as they could have been. Therefore, a future update of the *DSS* could enhance the final interaction between the system and the user, presenting the results more effectively, allowing for an easier understanding of the final task allocation, and improving knowledge of what can be refined or changed. Additionally, new graphs could be introduced to show *CT* imbalances between workstations, the bottleneck of the assembly line, and the potential reduction of *CT* for a workstation compared to the *CT* of the equipment it requires.

Additionally, regarding the presentation of the solution, the automatic completion of work instructions can be implemented. While the information provided may not be sufficient to fully complete the work instruction, simply generating the template and inputting the task title and *CT* will save considerable time for the operator and reduce the chance of errors. The missing information can then be manually entered.

Another proposed improvement is the ability to save the input page for later use. This feature aims to prevent the user from having to re-enter all the information from scratch each time they need to resume work on a previous project.

References

Aghajani, M., Ghodsi, R., & Javadi, B. (2014). Balancing of robotic mixed-model two-sided assembly line with robot setup times. International Journal of Advanced Manufacturing Technology, 74(5–8), 1005–1016. https://doi.org/10.1007/s00170-014-5945-x

Aguilar, H., García-Villoria, A., & Pastor, R. (2024). Heuristic and metaheuristic procedures for the Parallel Assembly Lines Balancing Problem with multi-line workstations and buffer sizing. Computers and Operations Research, 166. https://doi.org/10.1016/j.cor.2024.106596

Akpinar, S., Mirac Bayhan, G., & Baykasoglu, A. (2013). Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Applied Soft Computing Journal, 13(1), 574–589. https://doi.org/10.1016/j.asoc.2012.07.024

Almeida Simaria, A. S. (2006). Assembly line balancing - new perspectives and procedures.

Amen, M. (2000). An exact method for cost-oriented assembly line balancing. International Journal of Production Economics, 64, 187–195.

Amer, Y., & Hajiabolhasani, Z. (2013). Improving Injection Moulding Processes Using Experimental Design. https://www.researchgate.net/publication/269097699

Anderson, E. J., & Ferris, M. C. (1994). Genetic Algorithms for Combinatorial Optimization: The Assemble Line Balancing Problem. ORSA Journal on Computing, 6(2), 161–173. https://doi.org/10.1287/ijoc.6.2.161

Antani, K. R., Pearce, B., Kurz, M. E., Mears, L., Funk, K., & Mayorga, M. E. (2013). Manual Precedence Mapping And Application Of A Novel Precedence Relationship Learning Technique To Real-World Automotive Assembly Line Balancing. http://www.asme.org/about-asme/terms-of-use

Ashby, M. F. (2011). Processes and Process Selection. Materials Selection in Mechanical Design, 367–414. https://doi.org/10.1016/B978-1-85617-663-7.00013-8

Baybars, İ. (1986). A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. Management Science, 32(8), 909–932.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research, 168(3), 694–715. https://doi.org/10.1016/j.ejor.2004.07.023

Belkharroubi, L., & Yahyaoui, K. (2022). Solving the mixed-model assembly line balancing problem type-I using a Hybrid Reactive GRASP. Production and Manufacturing Research, 10(1), 108–131. https://doi.org/10.1080/21693277.2022.2065380

Bell Rae, J., & K. Binder, A. (2024). Automotive Industry. Britannica. https://www.britannica.com/technology/automotive-industry/Europe-after-World-War-II

Borba, L., & Ritt, M. (2014). A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. Computers and Operations Research, 45, 87–96. https://doi.org/10.1016/j.cor.2013.12.002

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. European Journal of Operational Research, 183(2), 674–693. https://doi.org/10.1016/j.ejor.2006.10.010

Boysen, N., Schulze, P., & Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? In European Journal of Operational Research (Vol. 301, Issue 3, pp. 797–814). Elsevier B.V. https://doi.org/10.1016/j.ejor.2021.11.043

Business Research Insights. (2024). Automotive Market Size, Share, Growth, and Global Industry Analysis, By Type (Passenger Vehicle and Commercial Vehicle), By Application (Personal Use, Municipal Use, and Business Use), Regional Insights, and Forecast To 2031. Business Research Insights. https://www.businessresearchinsights.com/market-reports/automotive-market-102183

Caixeta, M. C. B. F., & Fabricio, M. M. (2018). Métodos e instrumentos de apoio ao codesign no processo de projeto de edifícios. Ambiente Construído, 18(1), 111–131. https://doi.org/10.1590/s1678-86212018000100212

Capacho, L., Pastor, R., Dolgui, A., & Guschinskaya, O. (2009). An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem. Journal of Heuristics, 15(2), 109–132. https://doi.org/10.1007/s10732-007-9063-x

Chica, M., Cordón, Ó., Damas, S., & Bautista, J. (2010). Multiobjective constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. Information Sciences, 180(18), 3465–3487. https://doi.org/10.1016/j.ins.2010.05.033

Developing a line balancing tool for reconfigurable manufacturing systems A tool to support investment decisions. (n.d.).

Dolgui, A., & Proth, J. M. (2013). Assembly line balancing: Conventional methods and extensions. IFAC Proceedings Volumes (IFAC-PapersOnline), 46(9), 43–48. https://doi.org/10.3182/20130619-3-RU-3018.00644

Ebnesajjad, S. (2015). Injection Molding. Fluoroplastics, 236–281. https://doi.org/10.1016/B978-1-4557-3197-8.00010-9

Eghtesadifard, M., Khalifeh, M., & Khorram, M. (2020). A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017. Computers and Industrial Engineering, 139. https://doi.org/10.1016/j.cie.2019.106182

Erel, E. (n.d.). Binary Integer Formulation For Mixed-Model Assembly Line Balancing Problem.

Esmaeilian, G. R., Mahmad, M. M. H., Sulaiman, S., & Ismail, N. (n.d.). Assembly Line and Balancing Assembly Line.

Fernandes, F. S. (2021). Grupo Simoldes: Uma história feita de mudanças. https://www.jornaldenegocios.pt/negocios-iniciativas/premios-exportacao---internaci/detalhe/uma-historia-feita-de-mudancas

Fernandes, J., Machado, C., & Amaral, L. (2020). Methodology Used for Determination of Critical Success Factors in Adopting the New General Data Protection Regulation in Higher Education Institutions. In Research Methodology in Management and Industrial Engineering.

Francis, L. F. (2016). Melt Processes. Materials Processing: A Unified Approach to Processing of Metals, Ceramics and Polymers, 105–249. https://doi.org/10.1016/B978-0-12-385132-

1.00003-3

Galizia, F. G., ElMaraghy, W., ElMaraghy, H., Bortolini, M., & Mora, C. (2019). The evolution of molds in manufacturing: From rigid to flexible. Procedia Manufacturing, 33, 319–326. https://doi.org/10.1016/j.promfg.2019.04.039

Gökçen, H., Ağpak, K., & Benzer, R. (2006). Balancing of parallel assembly lines. International Journal of Production Economics, 103(2), 600–609. https://doi.org/10.1016/j.ijpe.2005.12.001

Gökçen, H., Kara, Y., & Atasagun, Y. (2010). Integrated line balancing to attain Shojinka in a multiple straight line facility. International Journal of Computer Integrated Manufacturing, 23(5), 402–411. https://doi.org/10.1080/09511921003642162

Greene, J. P. (2021). Injection Molding. Automotive Plastics and Composites, 241–254. https://doi.org/10.1016/B978-0-12-818008-2.00019-2

Hackman, S. T., Magazine, M. J., & Wee, T. S. (1989). Fast, Effective Algorithms for Simple Assembly Line Balancing Problems on JSTOR. Operations Research, 916–924. https://www.jstor.org/stable/171473

Hamta, N., Fatemi Ghomi, S. M. T., Jolai, F., & Akbarpour Shirazi, M. (2013). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. International Journal of Production Economics, 141(1), 99–111. https://doi.org/10.1016/J.IJPE.2012.03.013

Haq, A. N., Rengarajan, K., & Jayaprakash, J. (2006). A hybrid genetic algorithm approach to mixed-model assembly line balancing. International Journal of Advanced Manufacturing Technology, 28(3–4), 337–341. https://doi.org/10.1007/s00170-004-2373-3

Hoffmann, T. R. (1963). Assembly line balancing with a precedence matrix. Management Science, 9(4), 551–562.

Hoffmann, T. R. (1992). Eureka: A hybrid system for assembly line balancing. Management Science, 38(1), 39–47.

Ismail, N., Esmaeilian, G., Hamedi, M., Sulaiman, S., Ismail, N., Esmaeilian, G. R., Hamedi, M., & Sulaiman, S. (n.d.). Balancing of parallel assembly lines with mixed-model product. https://www.researchgate.net/publication/267204458

Joaquim, D., & Borges Gouveia, J. (n.d.). o júri presidente Doutor José Carlos da Silva Neves professor catedrático da Universidade de Aveiro.

Kheirabadi, M., Keivanpour, S., Chinniah, Y. A., & Frayret, J. M. (2023). Human-robot collaboration in assembly line balancing problems: Review and research gaps. Computers and Industrial Engineering, 186. https://doi.org/10.1016/j.cie.2023.109737

Kheirabadi, M., Keivanpour, S., Chinniah, Y., & Frayret, J. M. (2022). A Review on Collaborative Robot Assembly Line Balancing Problems. IFAC-PapersOnLine, 55(10), 2779–2784. https://doi.org/10.1016/j.ifacol.2022.10.151

Kimt, Y. J., Kimlt, K., & Cho~, Y. (1998a). A Heuristic-Based Genetic Algorithm For Workload Smoothing In Assembly Lines. In Computers Ops Res (Vol. 25, Issue 2).

Klindworth, H., Otto, C., & Scholl, A. (2012). On a learning precedence graph concept for the automotive industry. European Journal of Operational Research, 217(2), 259–269. https://doi.org/10.1016/j.ejor.2011.09.024

Kryachek, V. M. (2004a). Injection Moulding (REVIEW). In Powder Metallurgy and Metal Ceramics (Vol. 43).

Leo, G. (2022, February 1). How to Make Injection Molds: The Complete Process from Start to Finish. MadeAria. https://www.madearia.com/blog/how-to-make-injection-molds-the-complete-process-from-start-to-finish/

Li, M., Tang, Q., Zheng, Q., Xia, X., & Floudas, C. A. (2017). Rules-based heuristic approach for the U-shaped assembly line balancing problem. Applied Mathematical Modelling, 48, 423–439. https://doi.org/10.1016/j.apm.2016.12.031

Li, Y., Liu, D., & Kucukkoc, I. (2023). Mixed-model assembly line balancing problem considering learning effect and uncertain demand. Journal of Computational and Applied Mathematics, 422. https://doi.org/10.1016/j.cam.2022.114823

Li, Z., Janardhanan, M. N., Tang, Q., & Ponnambalam, S. G. (2019). Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. Swarm and Evolutionary Computation, 50. https://doi.org/10.1016/j.swevo.2019.100567

Li, Z., Kucukkoc, I., & Nilakantan, J. M. (2017). Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. In Computers and Operations Research (Vol. 84, pp. 146–161). Elsevier Ltd. https://doi.org/10.1016/j.cor.2017.03.002

Mason, A. (2013). SolverStudio: A new tool for better optimisation and simulation modelling in Excel. INFORMS Trans.

Mason, A. J. (2013). SolverStudio: A New Tool for Better Optimisation and Simulation Modelling in Excel. INFORMS Transactions on Education, 14(1), 45–52. https://doi.org/10.1287/ited.2013.0112

Maynard, H., Stegemerten, G., & Schwab, J. (1948). Methods-time measurement. New York: McGraw-Hill.

Mcmullen, P. R., & Frazierb, G. v. (1997). international journal of production economics ELSEVIER A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. In Int. J. Production Economics (Vol. 51).

Meng, K., Tang, Q., Zhang, Z., & Yu, C. (2021). Solving multi-objective model of assembly line balancing considering preventive maintenance scenarios using heuristic and grey wolf optimizer algorithm. Engineering Applications of Artificial Intelligence, 100. https://doi.org/10.1016/j.engappai.2021.104183

Mozdgir, A., Mahdavi, I., Badeleh, I. S., & Solimanpur, M. (2013). Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. Mathematical and Computer Modelling, 57(1–2), 137–151. https://doi.org/10.1016/j.mcm.2011.06.056

Nearchou, A. C. (2008). Multi-objective balancing of assembly lines by population heuristics. International Journal of Production Research, 46(8), 2275–2297.

Nourmohammadi, A., Fathi, M., & Ng, A. H. C. (2019). Choosing efficient meta-heuristics to solve the assembly line balancing problem: A landscape analysis approach. Procedia CIRP, 81, 1248–1253. https://doi.org/10.1016/j.procir.2019.03.302

Oksuz, M. K., Buyukozkan, K., & Satoglu, S. I. (2017). U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics.

Computers and Industrial Engineering, 112, 246–263. https://doi.org/10.1016/j.cie.2017.08.030

Özcan, U. (2019). Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. International Journal of Production Economics, 213, 81–96. https://doi.org/10.1016/j.ijpe.2019.02.023

Panneerselvam, R. (2005). Production and operations management (2nd ed.). PHI Learning.

Pereira Coutinho, C. (2014). Metodologia de Investigação em Ciências Sociais e Humanas: teoria e prática. www.almedina.net

Pereira, J., & Álvarez-Miranda, E. (2018). An exact approach for the robust assembly line balancing problem. Omega (United Kingdom), 78, 85–98. https://doi.org/10.1016/j.omega.2017.08.020

R. Dittrich, K., Geppert, A., & C. Norrie, M. (2001). Advanced Information Systems Engineering.

Rekiek, B., Delchambre, A., Dolgui, A., & Bratcu, A. (2002). Assembly Line Design: A Survey. www.elsevier.com/locate/ifac

Saif, U., Guan, Z., Wang, B., Mirza, J., & Huang, S. (2014). A survey on assembly lines and its types. In Frontiers of Mechanical Engineering (Vol. 9, Issue 2, pp. 95–105). Higher Education Press. https://doi.org/10.1007/s11465-014-0302-1

Salveson, M. E. (1955). The Assembly Line Balancing Problem. Journal of Industrial Engineering, 6, 18–25. https://www.scirp.org/reference/ReferencesPapers?ReferenceID=1780891

Schmid, N. A., Montreuil, B., & Limère, V. (2022). A case study on the integration of assembly line balancing and feeding decisions. IFAC-PapersOnLine, 55(10), 109–114. https://doi.org/10.1016/j.ifacol.2022.09.376

Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 168(3), 666–693. https://doi.org/10.1016/j.ejor.2004.07.022

Scholl, A., & Boysen, N. (2009). Designing parallel assembly lines with split workplaces: Model and optimization procedure. International Journal of Production Economics, 119(1), 90–100. https://doi.org/10.1016/j.ijpe.2009.01.011

Scholl, A., Boysen, N., Fliedner, M., Schriften Zur Wirtschaftswissenschaft, J., Fakultät, W., Lorenz, H.-W., Scholl, A., Boysen, N., & Fliedner, M. (2009). The assembly line balancing and scheduling problem with sequence-dependent setup times: Problem extension, model formulation and efficient heuristics. www.jbe.uni-jena.de

Scholl, A., Fliedner, M., & Boysen, N. (2010). Absalom: Balancing assembly lines with assignment restrictions. European Journal of Operational Research, 200(3), 688–701. https://doi.org/10.1016/j.ejor.2009.01.049

Scholl, A., & Voß, S. (1997). Simple assembly line balancing - heuristic approaches. Journal of Heuristics, 2(3), 217–244. https://doi.org/10.1007/BF00127358

Sewell, E. C., & Jacobson, S. H. (2012a). A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS Journal on Computing, 23(3), 433–442.

Sewell, E. C., & Jacobson, S. H. (2012b). A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS Journal on Computing, 24(3), 433–442. https://doi.org/10.1287/ijoc.1110.0462

Shahabuddin, S. (2009). Forecasting automobile sales. Management Research News, 32(7), 670–682.

Sharma, R. (2012). Sales Forecast of an Automobile Industry. In International Journal of Computer Applications (Vol. 53, Issue 12).

Simoldes Plastics. (n.d.-a). LinkedIn: Simoldes Plastics. Retrieved January 18, 2024, from https://www.linkedin.com/company/grupo-simoldes-plastics/about/

Simoldes Plastics. (n.d.-b). Simoldes: INPLAS. Retrieved January 18, 2024, from https://www.simoldes.com/en/plastics-companies/inplas/

Simoldes Plastics. (n.d.-c). Simoldes: Plásticos. Retrieved January 18, 2024, from https://www.simoldes.com/plastics/inicio/

Simoldes Plastics. (n.d.-d). Simoldes: Sobre Nós. Retrieved January 18, 2024, from https://www.simoldes.com/sobre-nos-grupo-simoldes/

Simoldes Plastics. (2022). Process Engineering Integration Brochure [Unpublished Presentation].

Singh, G., & Verma, A. (2017). A Brief Review on injection moulding manufacturing process. In Materials Today: Proceedings (Vol. 4). www.sciencedirect.comwww.materialstoday.com/proceedings

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. In International Journal of Advanced Manufacturing Technology (Vol. 73, Issues 9–12, pp. 1665–1694). Springer-Verlag London Ltd. https://doi.org/10.1007/s00170-014-5944-y

Sternatz, J. (2014). Enhanced multi-Hoffmann heuristic for efficiently solving real-world assembly line balancing problems in automotive industry. European Journal of Operational Research, 235(3), 740–754. https://doi.org/10.1016/j.ejor.2013.11.005

Stockemer, D. (2018). Quantitative Methods for the Social Sciences: A Practical Introduction with Examples in SPSS and Stata. In Quantitative Methods for the Social Sciences: A Practical Introduction with Examples in SPSS and Stata. Springer International Publishing. https://doi.org/10.1007/978-3-319-99118-4

Tiacci, L. (2017). Mixed-model U-shaped assembly lines: Balancing and comparing with straight lines with buffers and parallel workstations. Journal of Manufacturing Systems, 45, 286–305. https://doi.org/10.1016/j.jmsy.2017.07.005

Tsurkov, V. (2001). Large-scale Optimization: Problems and Methods (1st ed.). Kluwer Academic Publishers.

Wicky, D. (2023). LinkedIn: Understanding The Role of Operations Department. https://www.linkedin.com/pulse/day-40-understanding-role-operations-department-wicky-david/

Wilson, J. M. (2014). Henry Ford vs. assembly line balancing. International Journal of Production Research, 52(3), 757–765. https://doi.org/10.1080/00207543.2013.836616

Xu, S., Shavarani, S. M., Ghadiri Nejad, M., Vizvari, B., & Toghraie, D. (2023). A novel competitive exact approach to solve assembly line balancing problems based on lexicographic order of vectors. Heliyon, 9(3). https://doi.org/10.1016/j.heliyon.2023.e13925

Ying, B., Hongshun, Z., & Liao, Z. (2009). Mixed-model assembly line balancing using the hybrid genetic algorithm. 2009 International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2009, 3, 242–245. https://doi.org/10.1109/ICMTMA.2009.591

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., & Ardakani, H. D. (2012). Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. Journal of Manufacturing Systems, 31(2), 139–151. https://doi.org/10.1016/j.jmsy.2011.10.002

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

P.PORTO

DECLARAÇÃO DE INTEGRIDADE

DECLARAÇÃO DE INTEGRIDADE

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Declaro que o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

ISEP, Porto, 13 de setembro de 2024