applied sciences

MDPI

Article

Balancing Workload and Workforce Capacity in Lean Management: Application to Multi-Model Assembly Lines

Jordi Fortuny-Santos [1,*], Patxi Ruiz-de-Arbulo-López [2], Lluís Cuatrecasas-Arbós [3] and Jordi Fortuny-Profitós [4]

1 Department of Business Management, EPSEM School of Engineering, Universitat Politecnica de Catalunya, 08242 Manresa, Spain
2 Department of Business Management, University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain; patxi.ruizdearbulo@ehu.eus
3 Instituto Lean Management, 08172 Sant Cugat del Vallès, Spain; lluiscuat@gmail.com
4 Arcvi Big Data Agency, 25600 Balaguer, Spain; jordi@arcvi.io
* Correspondence: jordi.fortuny@upc.edu

Received: 4 November 2020; Accepted: 9 December 2020; Published: 10 December 2020

check for updates

Featured Application: Multi-model assembly line balancing.

Abstract: While multi-model assembly lines are used by advanced lean companies because of their flexibility (different models of a product are produced in small lots and reach the customers in a short lead time), most of the extant literature on how to staff assembly lines focuses either on single-model lines or on mixed-model lines. The literature on multi-model lines is scarce and results given by current methods may be of limited applicability. In consequence, we develop a procedure to staff multi-model assembly lines while taking into account the principles of lean manufacturing. As a first approach, we replace the concepts of operation time and desired cycle time by their reciprocal magnitudes workload and capacity, and we define the dimensionless term of unit workload (load/capacity ratio) in order to avoid magnitudes related to time such as cycle time because, in practice, they might not be known. Next, we develop the necessary equations to apply this framework to a multi-model line. Finally, a piece of software in Python is developed, taking advantage of Google's OR-Tools solver, to achieve an optimal multi-model line with a constant workforce and with each workstation performing the same tasks across all models. Several instances are tested to ensure the performance of this method.

Keywords: lean manufacturing; workforce; assembly line; OR-Tools; Python

1. Introduction

Designing efficient assembly processes requires computing the number of people necessary to accomplish the task and then assigning work elements to operators in an even manner. This is especially important in a lean management (LM) context, which requires products to be completed in a just-in-time fashion, without *muda* (Japanese for wasted resources in non-value-adding work), *muri* (people overburden) or *mura* (workload variation) [1]. Thus, this paper develops a procedure that balances the scheduled workload with the capacity of a process in order to compute the minimum number of operators to finish the job in the allowed time (e.g., a workday) and then it determines the time necessary to complete each model when more than one model is made during the same workday.

"Lean" was the word used by researchers at the Massachusetts Institute of Technology (MIT), in the 1980s, to describe the efficient plants of Japanese car manufacturers [2]. A fruit of their research was the book "The machine that changed the world" [3], which greatly contributed to the diffusion of LM among practitioners. LM has its roots in the Toyota Production System (TPS) [3], which was developed on the basis of two principles: respect for people (or employee engagement) and just-in-time (JIT) production [4]. For this reason, TPS, JIT and LM are frequently used as synonyms in the literature. For the purpose of this paper, LM can be defined as a management approach that tries to identify and remove all types of *muda*, *muri* and *mura* from the business processes in order to increase efficiency, cut costs and reduce order lead time. LM includes a variety of industrial practices such as single-piece flow (or, at least, small lot size), leveling of production (every day a stable mix of different models is made in order to meet customer demand) and setup time reduction techniques, which allow operators to quickly move from assembling one model to another [4–7].

An assembly line is a manufacturing process in which parts are added (or operations are performed) to a succession of products as these products visit workstations successively according to a previously defined sequence. Usually, workstations are aligned and placed close to one another. Products may be moved by a conveyor belt or another transportation system between stations. In an assembly line, workers (or even robots) are assigned specific operations, which they repeat before the product moves to the next station in the line. The assembly line devised by Henry Ford in the early days of the twentieth century was intended for one standardized product—the Ford Model T— but, currently, several models may be assembled on the same line. Consequently, assembly lines are classified either as "single-model", "multi-model" or "mixed-model" lines [8]. The single-model line is dedicated to the production of one single model of an end product (e.g., one type of car engine). In multi-model lines, different models of an end product are produced separately in batches. For example, in a day, first, 20 units of model A are assembled, one by one; later, as a batch, 30 units of model B are assembled on the same line, and after that, 15 units of model C are assembled, one after the other, on the same line too. The operation of a multi-model line frequently includes intermediate setup operations (e.g., time to change fixtures) between different models. For this reason, it is convenient to make the different models in batches in order to have small setup times compared to the lot processing time. In mixed-model lines, different models can be produced in any order [9] and the batch size can be as small as one. For example, one unit of A; after, one unit of B; and then two units of C. The sequence may be adjusted to market demand and/or it may try to smooth the demand for upstream components. Obviously, this manufacturing strategy can only be applied if setup times in between are not significant; otherwise, more time would be spent in unproductive setup than in assembly work.

We can find multi-model lines somewhere in the evolution from single-model assembly lines to mixed-model lines. A line that spends all year assembling the same model may evolve and spend several weeks on model X and then several more weeks on model Y. From the point of view of LM, large batches of X and Y are sources of *muda* such as overproduction and inventory, besides conveying a lot of unevenness. Since the evolution of assembly lines is based on the reduction in lot size, the reduction in time devoted to each model and the introduction of more models, the following step is to produce, each day, smaller batches of X and Y, and this is the situation that we analyze in the greatest part of this paper. The last step, so far, is the mixed-model line, where the batch size is one and the assembly line tries to meet the mixed-model demand that companies encounter every day.

Currently, in the dawn of Industry 4.0 [8,10], assembly lines continue to be widely used. They have many advantages but drawbacks too. Based on the principles of specialization and work division [11], the main benefit of assembly lines comes from the specialization of workers and machines. They perform repeatedly specific tasks that do not require the ability of a craftsman. Although little training is generally required—which offers greater employment opportunities—this system allows mass production of complicated goods such as cars or airplanes, while keeping labor costs low. Although the total assembly time may be long, the cycle time is usually short. For example, manufacturing a vehicle may take 10 h but we can see that every hour a vehicle leaves the assembly line [12]. These features are the basis of a highly efficient system with low costs per unit. In consequence, assembly lines have been adopted by lean companies because they have proved to be very efficient in the production of standardized products and even in low volume production of customized products [13]. In addition, one-piece flow avoids different types of *muda* such as dispensable transport, waiting, unnecessary movements, excessive work-in-process inventory and lack of quality, while saving floor space and allowing visual control [5,6,14]. Furthermore, since LM emphasizes the exigency to manufacture a variety of products in small batches to satisfy customer demand, in times of fierce competition, both mixed-model and multi-model assembly line problems become highly relevant for academia and industry [13].

Bukchin et al. [15] specified the main drawbacks of assembly lines:

(i) Low flexibility with respect to changes in demand. In addition, when models change, it is necessary to re-balance the line. Traditional lines were completely inflexible because they were designed for a single product but, currently, we may have multi-model and mixed-model lines as described in this paper. They allow production to meet demand.
(ii) Balance loss due to an imperfect balance of the line and stochastic task times, resulting in blockage (parts have to wait before a workstation) and starvation (workers have to wait) [11].
(iii) Poorly skilled people, poor work environment and poor quality. Luckily, LM gives a lot of importance to people engagement, which results in cross-training, teamwork and people committed to quality (or *jidoka* in Japanese) [4,5].
(iv) High costs of material handling (although this depends on the characteristics of the product) and high work-in-process, if lines are long. However, work-in-process is always smaller than in traditional batch-and-queue systems [12].

Additionally, repetitive motions, workers' postures (especially upper limbs), lifting efforts and the pace of work may become important threads for the safety and health of assembly workers [10,16], not to mention that such repetitive and monotonous work is not motivating, does not favor creativity and may result in alienation. Furthermore, especially in lean factories, continuous work and lack of buffers result in time pressure and stress [17]. On the other hand, empirical results show that lean practices create positive environments that offer improved job content, better quality of work and opportunities for participation and learning [17].

Most of the extant literature on assembly line balancing focuses either on single-model lines or on mixed-model lines. While the single-model assembly line balancing problem (SMALBP) was the subject of the earliest studies, currently, research focuses on mixed-model lines. Although multi-model assembly lines are also used in real companies [18], problems such as staffing and balancing multi-model lines have not been well studied. Having detected this gap between extant research and industry needs as an opportunity for research, this paper presents a procedure to staff multi-model assembly lines, based on the comparison between workload and capacity. The aim of our problem is to determine the number of workstations or operators so that, given N models, the work elements can be assigned to workstations (the number of stations being the same for all models), when batches of n_k units (the desired daily production is known) of each model $k = \{1. N\}$ are assembled.

The extant literature offers methods such as solving a single "combined" model, which may result in inapplicable solutions. Furthermore, contrary to the common approach, which postulates that the cycle time (how often the process generates a piece) is the same for all models and, in consequence, it is already known, we assume that the cycle time for each model is unknown, as we have encountered in a real situation. To be coherent with previous assumptions, we consider that the amount of time the line will spend assembling each model is not known either. In addition, solving each model as a separate problem may result in different numbers of workstations, whereas companies may prefer a constant number of operators because it is not convenient to vary the number of operators every time that the line has to assemble a different model [19], especially if these changes take place during the same shift.

Additionally, we take into account that, despite the huge amount of research on assembly lines, the number of companies that use mathematical methods to balance their lines is very low. Practitioners seem to be reluctant to methods that require computational complexity [9] and thus many companies use only trial and error methods [13,20] such as playing with the sequence using an electronic spreadsheet until the result is satisfactory [21]. Namely, LM relies on visual aids such as stacked bar charts called *yamazumi* charts [22]. In this paper, we present a method that was conceived as a simple tool for a real company and thus it can be implemented on an electronic spreadsheet for the sake of simplicity. After these steps, tasks are not allocated to workstations yet. The line may be then balanced, for each model, using exact, heuristic or graphic single-model assembly line balancing procedures. Finally, although LM relies on multi-skilled operators, we have developed a small computer program using Google's OR-Tools solver in order to find a solution where each workstation performs the same tasks for all models, at the expense of a number of operators above the minimum workforce. We hope our method helps to bridge the gap between theory and practice and provide better options, results and performance measures that lead to better decisions.

2. Relevant Literature on Multi-Model Assembly Lines

It was not until the 1950s that researchers became interested in assembly lines [23]. Since then, a large number of procedures have been developed to solve the assembly line balancing problem (ALBP) and its different extensions. As it is difficult to use exact methods with large problems because computers take too long to find the optimal solution [23], research has focused on finding efficient heuristic and metaheuristic algorithms [24,25].

Even though much work has been conducted on the single-model and mixed-model line balancing problems, little has been reported in the literature in respect to multi-model lines. The extant literature on balancing multi-model lines is scarce and the majority of the papers are not recent, as shown in different reviews [9,13,18,25].

The multi-model assembly line balancing problem was formulated as a linear integer programming problem [26] but, despite its theoretical interest, this formulation was not practical due to its computational requirements [20]. Alternatively, others suggest solving a single-model problem with the help of a combined precedence diagram, which transforms different models into a single combined model [27]. However, much inefficiency may arise when the solution of the combined model is applied to the real line [27]. Thomopoulos [28,29] used a combined precedence diagram to model a mixed-model line and he showed that his method can be applied to "batched" (multi-model) lines [29]. To solve the problem, the station cycle time is set to the shift time and task times are replaced by the total time required to complete this task in a given model mix that takes into account the amount of units of each model that have to be assembled [29]. The resulting problem is then solved as a SMALBP. In addition, Thomopoulos [29] mentioned, as advantages of multi-model lines, that companies can make batches of products with different cycle times for each model "without major shifts, if any, in station assignments" when the line changes from one model to another. Furthermore, he affirmed that it is a bad practice to change the assignment of tasks from one station to another when the line moves from one model to another model [29]. We are going to take into account these key ideas in this paper in order to staff and balance multi-model lines with a constant number of operators, without switching tasks from model to model.

Other papers suggest that multi-model lines, especially for large lots, can be balanced separately for each model [13,30–32]. Another option is to study each model in an independent way. Then, common configurations are selected as feasible configurations for all models [33]. This method is limited to few models and cases with long setup times. Roser [34], besides suggesting balancing the line separately for each model, also proposed either balancing the line for only the most frequent model or computing, for each task, a weighted average performance time taking into account the quantities to be produced. As a drawback, when models are studied separately, each resulting line may need a different number of operators [13].

Although metaheuristic techniques are mainly applied to mixed-model lines, a few works use them in multi-model lines [35,36]. Finally, unlike our paper, which focuses on computing the number of workstations of a generic, single-piece flow, manual, multi-model assembly line, some researchers optimized specific scenarios. Among them, we can mention the study of a line with common tasks (for several models) with the same performance time [37] or the study of a multi-model (or multi-product) line as if it was a single-product line, without focusing on the number of stations [38].

3. Materials and Methods: Workforce Capacity and Workload

3.1. A Few Definitions

In LM, the first step to balance a single-model assembly line is to compute customer *takt* time (TT) as in Equation (1). *Takt* time is the ratio of the effective working time (or net available time AT) to the number of units (q) to be produced during that time [5,39]. This conveys the lean principle that tasks on the shop floor are synchronized with a beat set by the customer: the *takt* time is the maximum amount of time that can elapse between two consecutive units in order to meet demand.

$$TT = \frac{AT}{q} \tag{1}$$

In this paper, the term "cycle time of a process" (such as an assembly line) refers to the time elapsed between two consecutive unit completions (at the end of the line) and "cycle time of a workstation" is the amount of time elapsed between two consecutive units exiting the station. Since the cycle time of a serial line process equals the longest cycle time of the workstations [12], the longest station time (sum of the task times assigned to the station) of the assembly line cannot be above the *takt* time [39] or desired cycle time. In practice, in order to compensate unexpected stoppages, the desired cycle time may be set below the *takt* time.

When the *takt* time (or desired cycle time) is known, the theoretical minimum number of workstations (w) is simply computed according to Equation (2) [5], where OT is the total operation time per workpiece (the sum of task times t_i).

$$w = \frac{OT}{TT} = \frac{\sum_i t_i}{TT} \tag{2}$$

Finally, tasks are assigned to workstations so that the resulting station times are never above the *takt* time. As an elemental task cannot be split between two workstations, the resulting number of workstations may be greater than w.

Since we assume that, in a multi-model line, the desired number of units of each model is known, because it is related to customer demand, but the time the line will be devoted to each model is unknown, in the following sections, we report a new framework to staff production systems that relies on the concepts of workload and capacity. When an operator, a workstation or an assembly line are assigned a job to do, we can say that they are given a certain workload. We can define the workload (e.g., of a workstation) as the amount of work assigned to such workstation per time unit (e.g., production in parts per day or man-hours per day). The capacity of a workstation is defined as the amount of work that a workstation can process per time unit [34]. In this paper, without loss of generality, we assume that a workstation is operated by one person at a time. Equation (3) describes the capacity that a given workstation requires to perform operation i on model k (CP_{ik}) in terms of how many times an operation that requires t_{ik} time units can be repeated in the available time (AT).

$$CP_{ik} = \frac{AT}{t_{ik}} \tag{3}$$

While extant line balancing methods are aimed at achieving similar station times for all workstations with values close to, but always below, the desired cycle time, in our approach, since the cycle time is not known, we balance capacity against workload in search of similar load/capacity ratios for each workstation, with station workload being less than or equal to station capacity. For this reason, the load/capacity ratio is defined as a dimensionless magnitude that we name unit workload of operation i on model k (Wu_{ik}) or, in consequence, the unit workload that bears a certain workstation that has to perform task i on model k. This is the ratio of the desired production of model k (q_k) to the capacity CP_{ik} of the workstation, both q_k and CP_{ik} being measured in parts per day (Equation (4)).

In the TPS, since its conception in the 1950s, the term *heijunka* (translated as production leveling) refers to distributing the production of different models evenly over a time period (e.g., a day) and entails, beyond the traditional line balancing approach, the concept of leveling or equating the work load to be performed to the capacity of operators or machines to complete that work [40]. Hence, the concepts that we operationalize in this paper stem from the very heart of LM. In Equation (4), workload becomes related to capacity and time units are avoided. If we examine Equations (1) and (3) together, we realize that load (q) and *takt* time (or desired cycle time) are reciprocal concepts as well as capacity and operation time (or station time). For this reason, an analysis based on load and capacity is equivalent to the traditional study based on operation time and cycle time [41].

$$Wu_{ik} = \frac{q_k}{CP_{ik}} \tag{4}$$

In Equation (5), the time that a given workstation spends (e.g., during the day or available time AT) performing task i in the production of model k (TP_{ik})—simply computed as q_k times the operation time t_{ik}—becomes related to capacity (CP_{ik}), by means of Equation (3), and especially to the unit workload of the workstation (Wu_{ik}), thanks to Equation (4). This way, temporal magnitudes may be regained when necessary. In Equation (5), Wu_{ik} can be isolated and then described as a fraction of the available time (Equation (6)).

$$TP_{ik} = t_{ik} \cdot q_k = \frac{AT}{CP_{ik}} . q_k = AT \cdot Wu_{ik} \tag{5}$$

$$Wu_{ik} = \frac{TP_{ik}}{AT} \tag{6}$$

According to its definition, when $Wu_{ik} \leq 1$, an operator is enough to complete a given job within the time allotted. If $Wu_{ik} < 1$, there is some idle time, and more operations could be assigned to the operator. Alternatively, if $Wu_{ik} > 1$, more people are necessary to complete the task. This approach is not only valid for assembly lines. Regardless of the type of layout or manufacturing strategy, the value of Wu_{ik} (rounded up to the next integer, if part-time work is not considered) gives the theoretical minimum number of operators (w) to complete the scheduled work.

3.2. From Single-Model to Multi-Model Production

As stated above, Equations (1) and (2) show the first steps to compute the necessary workforce for a single-model assembly line. If k models are considered (in different single-model lines), Equations (1) and (2) can be combined to create Equation (7), which determines the theoretical minimum number of workstations (w_k) for each model or line.

$$w_k = q_k \cdot \frac{OT_k}{AT} \tag{7}$$

Since the aim of this paper is to apply our framework to determine the minimum number of operators in situations that involve assembling multiple models with a constant workforce, w_k becomes w, a common value for all models (Equation (8)). In consequence, AT has to be replaced by the time the system will theoretically spend assembling the q_k units of a specific model (TP_k) (Equation (8)). TP_k becomes isolated in Equation (9) but it will be better described in Equation (14) because the second factor in Equation (9) represents that work elements can be split between operators, which is not the common case.

$$w = q_k \cdot \frac{OT_k}{TP_k} \tag{8}$$

$$TP_k = q_k \cdot \frac{OT_k}{w} = q_k \cdot \frac{\sum_i t_{ik}}{w} \tag{9}$$

3.3. Computing the Required Workforce

Now the necessary concepts have been defined, this section describes the following steps to find w and TP_k. The process is illuminated with a numerical example. Let us consider an assembly process—although the example could be extended to other types of operations—where only two operations, X and Y, exist (and X precedes Y). Several units of two different models (A and B) are assembled each day. In Table 1, the amount of work (or workload) corresponding to the daily production (q_{ik}) is measured in man-hours per day for each operation and model. Notice that if the load q_k is measured in parts, its value is common across the assembly process, while if it is measured in man-hours, it may be different for each operation and q_{ik} must be used instead of q_k. Let us assume that the available time is one eight-hour shift and that each operator's labor capacity is eight man-hours per shift. Equation (4) is used to compute unit workload per operation and model (Wu_{ik}).

Table 1. Unit workload per operation (Wu_{ik}) and per model (TWu_k) and total load (TWu).

Model (k)	Operation X		Operation Y		Total Unit Workload per Model $TWu_k = \Sigma Wu_{ik}$
	Load (Man-Hours) q_{ik}	Unit Workload $Wu_{ik} = q_{ik}/CP_{ik}$	Load (Man-Hours) q_{ik}	Unit Workload $Wu_{ik} = q_{ik}/CP_{ik}$	
Model A (1)	48	48/8 = 6	16	16/8 = 2	6 + 2 = 8
Model B (2)	8	8/8 = 1	24	24/8 = 3	1 + 3 = 4
Total multi-model unit workload $TWu = \Sigma TWu_k =$					12

Each worker's capacity (CP_{ik}) is 8 man-hours per day.

Next, the total unit workload per model (TWu_k) and the total multi-model unit workload of the system (TWu) are computed according to Equations (10) and (11). This way, the unit workloads of different processes are added to compute the unit workload of a multi-model system. Since the total multi-model unit workload (TWu) in Table 1 is twelve (twelve times what a person can perform in a day), the minimum number of employees to complete the task in the allowed time is twelve (Equation (12)). The ceiling function · is used to return the smallest integer greater than or equal to TWu if TWu is not a whole number.

$$TWu_k = \sum_i Wu_{ik} \tag{10}$$

$$TWu = \sum_k TWu_k \tag{11}$$

$$w = TWu = \sum_k \sum_i Wu_{ik} \tag{12}$$

Now, our procedure redistributes the workforce capacity in order to complete the jobs in eight hours assuming a constant team of 12 cross-trained workers. Asterisks represent new values after w has been found. We assume that the system is in steady state so all operators can start working immediately with no wait.

The plant just needs a fraction of the day to complete model A since its new labor capacity (CP^*_k = 12) is greater than any model's workload (TWu_k). This way, capacity is adjusted to the desired workload. Equation (13) is derived taking into account that the time devoted to a model (TP_k) is inversely proportional to the available capacity (CP^*_k) and Equation (13) admits that the time devoted to each model has to be proportional to its relative workload. Since the job to perform has not changed, the real workload is not CP^*_k but it remains constant ($TWu^*_k = TWu_k$) within the scheduled time TP_k. In our numerical example, according to Equation (14), a crew of twelve operators will be producing model A during 8/12 (TWu_A/TWu) of the available time or TP_A = 5 h and 20 min. Throughout the rest of the day, the plant will be producing model B.

$$TWu^*_k{\cdot}AT = CP^*_k{\cdot}TP_k \tag{13}$$

$$TP_k = \frac{TWu_k}{TWu} \cdot AT \tag{14}$$

Next, we want to assign some of the 12 operators to operation X (w_{XA}) and the rest to Y (w_{YA}). The values of the unit workload ($Wu^*{}_{i\mathrm{A}}$) have to be computed again (Equation (15)) because the time devoted to model A is TP_{A} instead of AT.

$$w_{ik} = Wu^*_{ik} = \frac{TWu}{TWu_k} \cdot Wu_{ik} = TWu \cdot \frac{Wu_{ik}}{TWu_k} \tag{15}$$

Since the unit workload ($Wu^*{}_{i\mathrm{A}}$) is 9 for product A (according to Equation (15)), operation X requires nine operators (w_{XA} = 9) during TP_{A}, and three operators will be assigned to operation Y. If $Wu^*{}_{ik}$ is not a whole number, one of the operators in charge of one operation might help with the other operation for a while, as described by Monden [5]. Otherwise, the company should consider several alternatives: (i) that task would be assigned to the fastest workers; (ii) overtime; (iii) as lean manufacturing is based on continuous improvement and problem solving [33], this would be an opportunity for re-designing that task; and (iv) another operator might be added. If Equation (15) is now applied to model B, three people are assigned to operation X and nine to operation Y. As the number of operators performing X or Y depends on the model, at least six cross-trained people would be necessary to complete the task with a constant workforce of 12 people.

Before attempting the design of a multi-model assembly line in Section 4, let us rewrite Equation (15) in terms of capacities and total unit workload of the system (Equations (4) and (10)), as shown in Equation (16). Then, the capacity provided by the operators is adjusted to the workload (Equation (17)), with the same number of operators (w) for all models, as desired [41].

$$Wu^*_{ik} = Wu_{ik} \cdot \frac{TWu}{TWu_k} = TWu \cdot \frac{\frac{q_k}{CP_{ik}}}{\sum_i \frac{q_k}{CP_{ik}}} = TWu \cdot \frac{\frac{1}{CP_{ik}}}{\sum_i \frac{1}{CP_{ik}}} = \frac{TWu}{\sum_i \frac{1}{CP_{ik}}} \cdot \frac{1}{CP_{ik}} \tag{16}$$

$$w = CP^*_k = \sum_i Wu^*_{ik} = \sum_i \left(TWu \cdot \frac{Wu_{ik}}{TWu_k} \right) = TWu \cdot \frac{\sum_i Wu_{ik}}{TWu_k} = TWu \ \forall k \tag{17}$$

Taking into account the changes in $Wu^*{}_{ik}$, Equation (4) may be rewritten as Equation (18). This allows computing the amount of units of model k that can go through operation i per time unit ($q^*{}_k$). We can rename $q^*{}_k$ as P_{ik} (for daily production rate) and then isolate it in Equation (18) and combine it with Equations (4) and (15) in order to relate the daily production capacity of that model (P_{ik}) with its scheduled production (q_k) (Equation (19)). Equation (19) is the expression of a correctly balanced system, with a common rate for all operations, because it does not depend on i.

$$Wu^*_{ik} = \frac{q^*_k}{CP_{ik}} \tag{18}$$

$$P_{ik} = Wu^*_{ik} \cdot CP_{ik} = TWu \cdot \frac{Wu_{ik}}{TWu_k} \cdot \frac{q_k}{Wu_{ik}} = \frac{TWu}{TWu_k} \cdot q_k \ \forall i \tag{19}$$

4. Staffing a Multi-Model Assembly Line

4.1. Problem Statement

In this section, we show how our method based in workforce capacity and workload, described in Section 3, is applied to staffing a multi-model assembly line. Without loss of generality, the procedure is illustrated with a small example in order to avoid large amounts of numbers. However, the method was designed to compute the necessary workforce for the multi-model assembly lines of a production plant that assembled cell phones—one million units per year—computer monitors and DVD players.

With a turnover of EUR 800 million, the plant provided employment to around 900 employees and it implemented LM to improve its operational performance.

Let us assume that the company wants to assemble three different models (Alpha, Beta and Gamma) of a certain device, such as a cell phone, in a multi-model assembly line. To approach LM, the company wants to maintain a daily production schedule [42]. Although production is arranged into model-specific lots, this approach reduces inventories, adds flexibility and focuses on process flow. The estimated monthly demand (Table 2) is divided by the number of working days in the month (22) so that production of each model is repeated once a day in the amounts shown in Table 2.

Table 2. Daily schedule.

Model (k)		Demand per Month (Units/Month)	Daily Production (q_k) (Units/Day)	Production Proportion ($\omega_k = q_k/\Sigma q_k$)
1	Alpha	30,800	1400	4/7
2	Beta	15,400	700	2/7
3	Gamma	7700	350	1/7
	Total	53,900	2450	1

Single tasks or work elements to assemble each model are identified by a number in Table 3. Task times are known constants, but they may have different values for different models (and some models may skip some tasks). Usually, time studies with a stopwatch are used to obtain these values [43]. Stopwatches used in work studies enable timing in different formats such as decimal minutes or decimal hours, but we consider that seconds are precise enough for our study. Anyway, this does not affect either the generalization or the performance of the methods described in this paper. This paper assumes that resources are homogeneous and thus the duration of tasks does not depend on the stations to which they are assigned and any task can be carried out at any station [24].

Table 3. Task times for each model t_{ik} (s/unit).

Task	Alpha	Beta	Gamma
1	6.0	5.0	8.0
2	7.0	11.0	13.0
3	4.0	6.0	-
4	5.0	5.0	5.0
5	5.0	3.0	2.0
6	2.0	1.0	4.0
7	4.0	1.0	3.0
8	7.0	4.0	13.0
9	3.0	4.0	1.0
10	1.0	1.0	1.0
11	10.0	11.0	8.0
12	1.0	-	3.0
Sum	55.0	52.0	61.0

4.2. Solution for a Combined Equivalent Model

In order to find a solution that could be later compared to our method, we created a combined precedence diagram with all the precedence relationships for Alpha, Beta and Gamma (Figure 1) [27]. This procedure transforms the different models into a combined model that we named "Omega". Taking into account the values of ω_k in Table 2, a weighted average task time (Equation (20)) was computed, for each work element (t_i), for the product Omega [11,34]. Resulting values are in Figure 1.

$$t_i = \frac{\sum_{k=1}^{3}(t_{ik} \cdot \omega_k)}{\sum_{k=1}^{3}\omega_k} \tag{20}$$

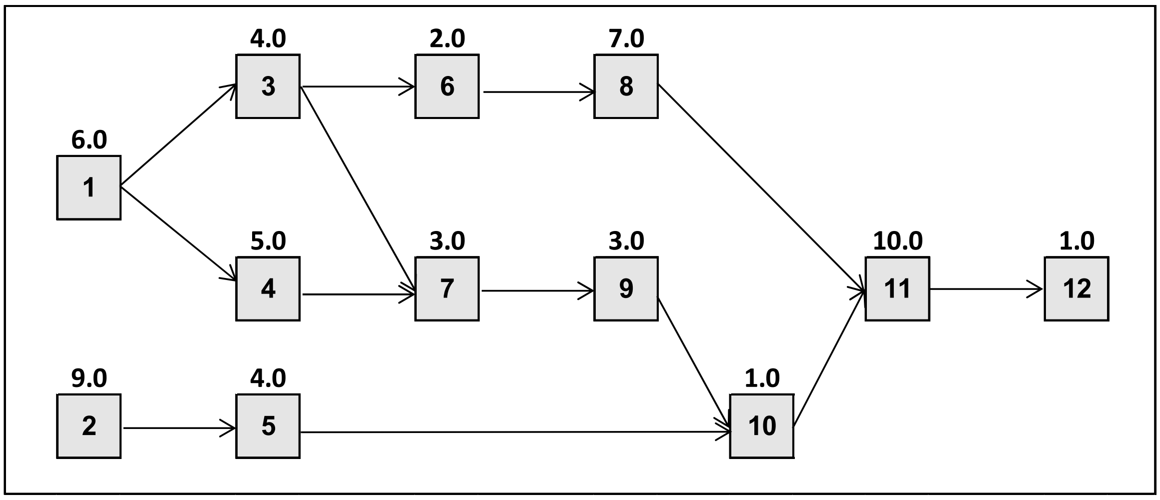

Figure 1. Precedence relationships between tasks for "combined" model Omega. Values above the squares are task times (in seconds).

According to Section 3.1, a common *takt* time (11.76 s) and the theoretical minimum number of workstations (4.7, rounded up to 5) for model Omega were computed with Equations (1) and (2). Integer linear programming (ILP) was used to find the optimal solution of the SMALBP for model Omega [44]. Table 4 shows that the optimal solution requires six workstations and involves eleven seconds of idle time per part, but this result might not be feasible in practice because, for model Gamma, some work elements would exceed the *takt* time.

Table 4. Tasks assigned to workstations.

Workstation	**Tasks**	**Station Time (*)**
(*j*)	**(*i*)**	**(s)**
1	1–4	11.0
2	2	9.0
3	3–5–6	10.0
4	7–8	10.0
5	9–10	4.0
6	11–12	11.0

* For model Omega.

4.3. Computing Manpower through Workload and Capacity

In this section, the problem presented in Section 4.1 is going to be solved with our method. The necessary equations have been implemented in Tables 3 and 5, Tables 6–8 on an electronic spreadsheet. In Table 5, we compute (with Equation (3)) how many times each work element *i* for each model *k* could be repeated in a day (i.e., CP_{ik}). We assume that the daily available time (*AT*) is 8 h (or 28,800 s). Changeover time is omitted because, in lean companies, setup times are usually small [45]. Otherwise, all sorts of non-productive time should be deducted from the available time.

Table 5. Capacity and unit workload for each task and each model.

Model (*k*): Daily Volume q_i	**Alpha (1) 1400 Units**		**Beta (2) 700 Units**		**Gamma (3) 350 Units**	
Task (*i*)	**Capacity (Units/Day) CP_{ik}**	**Unit Load Wu_{ik}**	**Capacity (Units/Day) CP_{ik}**	**Unit Load Wu_{ik}**	**Capacity (Units/Day) WP_{ik}**	**Unit Load Wu_{ik}**
1	4800.0	0.292	5760.0	0.122	3600.0	0.097
2	4114.3	0.340	2618.2	0.267	2215.4	0.158
3	7200.0	0.194	4800.0	0.146	-	0.000
4	5760.0	0.243	5760.0	0.122	5760.0	0.061
5	5760.0	0.243	9600.0	0.073	14,400.0	0.024
6	14,400.0	0.097	28,800.0	0.024	7200.0	0.049
7	7200.0	0.194	28,800.0	0.024	9600.0	0.036
8	4114.3	0.340	7200.0	0.097	2215.4	0.158
9	9600.0	0.146	7200.0	0.097	28,800.0	0.012
10	28,800.0	0.049	28,800.0	0.024	28,800.0	0.012
11	2880.0	0.486	2618.2	0.267	3600.0	0.097
12	28,800.0	0.049	-	0.000	9600.0	0.036
$TWuk = \Sigma Wu_{ik}$		2.674		1.264		0.741
Total multi-model unit load (TWu) = ΣTWu_k = 4.679						

Table 6. Reciprocal values of the required capacity per task and model $1/CP_{ik}$.

Task (*i*)	**Alpha**	**Beta**	**Gamma**
1	0.00021	0.00017	0.00028
2	0.00024	0.00038	0.00045
3	0.00014	0.00021	0
4	0.00017	0.00017	0.00017
5	0.00017	0.00010	0.00007
6	0.00007	0.00003	0.00014
7	0.00014	0.00003	0.00010
8	0.00024	0.00014	0.00045
9	0.00010	0.00014	0.00003
10	0.00003	0.00003	0.00003
11	0.00035	0.00038	0.00028
12	0.00003	0	0.00010
$\Sigma(1/CP_{ik})$	0.00191	0.00181	0.00212
$TWu/\Sigma(1/CP_{ik})$ (units/day)	2450.00	2591.35	2209.02

Values of $1/CP_{ik}$ in days/unit.

In Table 5, each task's unit workload (Wu_{ik}) is computed according to Equation (4). Unit loads for the operations of a product are summed (Equation (10)) in order to know the theoretical unit workload (TWu_k) of each model. These values, according to Equation (12), would be the minimum number of workstations in separated single-model lines. Finally, we add these quantities to obtain the unit workload of the line (TWu) (Equation (11)). Since TWu is 4.679, the multi-model line requires, at least, five workstations (Equation (12)) and this value has been computed without resorting to the concept of cycle time.

In previous lines, each task's unit workload (Wu_{ik}) was computed as if each model was made in a different line during an eight-hour shift, but now it is necessary to compute their new values (Wu^*_{ik}) for the multi-model line. They are calculated with Equation (16), for each task and model. On the spreadsheet, this has been conducted in accordance with the following steps: first, the reciprocal values of the capacity ($1/CP_{ik}$) are computed (Table 6); next, the unit workload of the line (TWu) is divided by the sum of the reciprocal values of the capacity for each model. As it will be proven in Equation

(21), these values represent the daily capacity of the line for each model (e.g., the five-station line can assemble up to 2209 units of model Gamma in a day). Finally, the daily capacity is multiplied by the reciprocal values of the capacity in order to compute the new unit workloads (Wu^*_{ik}) (which can be found in Table 7). For each model, the sum of its operations' loads provides the total workload (TWu_k) (see Table 7, bottom row), which remains unchanged.

Table 7. New unit workload (Wu^*_{ik}) per task and model.

Task (*i*)	Alpha	Beta	Gamma
1	0.51042	0.44989	0.61362
2	0.59549	0.98975	0.99713
3	0.34028	0.53986	0
4	0.42535	0.44989	0.38351
5	0.42535	0.26993	0.15340
6	0.17014	0.08998	0.30681
7	0.34028	0.08998	0.23011
8	0.59549	0.35991	0.99713
9	0.25521	0.35991	0.07670
10	0.08507	0.08998	0.07670
11	0.85069	0.98975	0.61362
12	0.08507	0	0.23011
$TWu_k = \Sigma Wu^*_{ik}$	4.679	4.679	4.679

Table 8. Expected performance of the multi-model line.

Model (*k*)	Unit Load TWu_k	Production Time (min/Day) TP_k	Production Rate (Units/Day) P_{ik}	Daily Output (Units) P_k
Alpha	2.674	274.0	2450.00	1400
Beta	1.264	130.0	2591.35	700
Gamma	0.741	76.0	2209.02	350
Total	4.679	480.0	-	2450

Equation (14) is used to estimate (in Table 8) the lapse of time the multi-model line has to be devoted to each model (TP_k) based on data from Table 5. Again, these values have been found without reference to any cycle time. The daily production rate (P_{ik}) or daily capacity is also calculated for each model with Equation (19). The resulting quantities (see Table 8) equal the values in the bottom row in Table 6, as Equation (21) confirms. Equation (21) is Equation (19) but Wu^*_{ik} has been replaced by its expression in Equation (16).

$$P_{ik} = Wu^*_{ik} \cdot CP_{ik} = \frac{TWu}{\sum_i \frac{1}{CP_{ik}}} \cdot \frac{1}{CP_{ik}} \cdot CP_{ik} = \frac{TWu}{\sum_i \frac{1}{CP_{ik}}} \tag{21}$$

For each model, the daily output of the line (P_k) has to be proportional to the assembly rate (P_{ik}—which is common for all i) and to the production time (TP_k). These relationships are modeled with Equation (22). Resulting values of P_k can be found in Table 8. They match each model's scheduled production (q_k) in Table 2.

$$P_k = P_{ik} \frac{TP_k}{AT} \tag{22}$$

4.4. Balancing the Multi-Model Line

The last step to design a line is the assignment of tasks to workstations [46]. Each model (k) can be considered as a separate SMALBP because the time devoted to each product (TP_k) is now known (as well as the minimum number of workstations). Therefore, each model can be solved using *yamazumi* charts, heuristics or exact methods. In our example, in order not to miss the optimal solution, ILP was used, model by model. Expression (23) is the multi-model objective function and Equations (24)–(27) show linear constraints. Each variable x_{ij} (or x_{hj}) is 1 if task i is assigned to workstation j. Otherwise, x_{ij} is zero. Twelve tasks ($|I| = 12$) and up to six ($|J| = 6$) possible workstations are considered in the model because the required number of workstations might not be the theoretical minimum value. If we compare Equation (25) with Baybars' model [44], workload values ($Wu*_{ik}$) found in previous sections are used instead of task times and the capacity of each workstation (E_j) replaces the cycle time of the line. E_j may be either zero or one (one operator), depending on whether the station is necessary or not in that line. Table 9 displays the optimal solution for each model. The result shows that, for every model, all work elements are assigned to five workstations or operators.

$$\text{Objective function}: \; Min \sum_j E_j \tag{23}$$

Subject to

$$\sum_j x_{ij} = 1 \; \forall i \tag{24}$$

$$\sum_i Wu^*_{ik} \cdot x_{ij} \leq E_j \; \forall j \tag{25}$$

$$\sum_j j \cdot x_{ij} \leq \sum_j j \cdot x_{hj} \; \forall h; \; \forall i \prec h \tag{26}$$

$$E_{j+1} \leq E_j \;\; \forall j \tag{27}$$

To measure the efficiency of a multi-model line, we develop Equation (28) in terms of load and capacity [34]. In our example, since the total multi-model unit workload of the system (TWu) is 4.68 and the capacity (ω) is 5, the efficiency is 0.936 or 93.6 percent.

$$Efficiency = \frac{TWu}{w} \cdot 100 \tag{28}$$

Table 9. Assignment of tasks to workstations.

Model (*k*):	**Alpha**		**Beta**		**Gamma**	
Workstation (j)	**Tasks (*i*)**	**Unit Load $\Sigma Wu*_{ik}$**	**Tasks (*i*)**	**Unit Load $\Sigma Wu*_{ik}$**	**Tasks (*i*)**	**Unit Load $\Sigma Wu*_{ik}$**
1	1–4	0.94	1–4	0.90	1–4	1.00
2	2–3	0.94	2	0.99	2	1.00
3	5–6–7	0.94	3–5–6–7	0.99	5–6–7	0.69
4	8–9–10	0.94	8–9–10	0.81	8	1.00
5	11–12	0.94	11	0.99	9–10–11–12	1.00

4.5. Optimizing the Assignation

The results in Section 4.4 are very satisfactory and our procedure outperforms the method described in Section 4.2. However, balancing each model in an independent way might lead to slightly different number of operators per model or to workstations performing different tasks depending on the model (as is the case for tasks 3, 9 and 10), while in multi-model lines, tasks common to more than one model should be performed by the same operator [30], according to the principles of specialization, learning and avoiding the cost of duplicating equipment [11]. In addition, this makes it easier to move from assembling a model to assembling another model. Luckily, in LM, when workers are cross-trained, it is permissible to assign the same work elements to different workers for different models [11].

In order to find an optimal solution with the minimum common number of workstations where the same tasks are assigned to the same workstations across the different models, as much as possible, while avoiding additional trial and error work, we developed a piece of software coded in Python 3.7 using the CP-SAT solver included in Google's OR-Tools library [47,48] since our problem can be written as a constraint programming model.

In order to analyze the $|K| = 3$ models at the same time, Equation (24) becomes Equation (29), where x_{ijk} are integer-Boolean variables that represent, when their value is 1, that task i is assigned to workstation j for model k. Otherwise, their value is zero. Equation (25) becomes Equation (30), where integer variables E_{jk} specify whether station j is necessary for model k (these variables can only take the value 0 or 1 since the capacity of a workstation is 1). Equation (26) becomes Equation (31) and Equation (27) becomes Equation (32).

$$\sum_j x_{ijk} = 1 \ \forall i \ \forall k \tag{29}$$

$$\sum_i Wu^*_{ik} \cdot x_{ijk} \leq E_{jk} \ \forall j \ \forall k \tag{30}$$

$$\sum_j \ j \cdot x_{ijk} \leq \sum_j \ j \cdot x_{hjk} \ \forall h; \ \forall i \ \prec \ h \ \forall k \tag{31}$$

$$E_{j+1\ k} \leq E_{jk} \ \forall j \ \forall k \tag{32}$$

In the computer program, the constraints described in these equations, as well as the objective function, have been coded following the solver's requirements. The objective function (Expression (33)) rewards performing the same task in the same station across the different models and penalizes the number of stations used in each model.

$$\text{Objective function}: \ Min \ \beta_1 \sum_j \sum_k E_{jk} - \beta_2 \sum_i B_i \tag{33}$$

β_1 is the weight assigned to the number of stations in the line. β_2 is the weight associated with the number of "triplets"—a task is performed in the same workstation for the three models. Binary variables (B_i) indicate whether task i is assigned to the same station for all models. In order to define B_i using linear constraints, we introduce integer-Boolean variables b_{ij} that take the value of 1 only if task i is assigned to workstation j for all models (Equations (34) and (35)).

$$\sum_k x_{ijk} \geq |K| \cdot b_{ij} \ \forall j \ \forall i \tag{34}$$

$$\sum_j b_{ij} = B_i \ \forall i \tag{35}$$

Together, β_1 and β_2 appraise how valuable it is to achieve one more "triplet" at the expense of operating an additional station. Managerial policies should decide the relative values of these coefficients.

To increase computational speed, the solver works over the integers. This means that the ILP must be defined using integers only. For this reason, the values of Wu^*_{ik} have been multiplied by a large integer (100,000) and rounded to the upper nearest whole number. The computer took 0.11 s to find the optimal solution (for $\beta_1 = \beta_2 = 1$), as shown in Table 10. If the changeover time is negligible, the assembly line could be operated as a mixed-model line and, according to the weights in Table 2, the line would assemble a basic sequence (four Alphas, two Betas, one Gamma) repeatedly in order to follow the principles of *heijunka* [40].

Table 10. Optimal assignment of tasks to workstations.

Model (k):	Alpha		Beta		Gamma	
Workstation (j)	Tasks (i)	Unit Load ΣWu^*_{ik}	Tasks (i)	Unit Load ΣWu^*_{ik}	Tasks (i)	Unit Load ΣWu^*_{ik}
1	2	0.60	2	0.99	2	1.00
2	1–4	0.94	1–4	0.90	1–4	0.65
3	3–5–6	0.94	3–5–6	0.90	5–6	0.46
4	8	0.60	8	0.36	8	1.00
5	7–9–10	0.68	7–9–10	0.54	7–9–10	0.38
6	11–12	0.94	11	0.99	11–12	0.85

5. Results of Computational Experiences

In order to evaluate its performance, our methodology and our computer program were used to solve different instances of multi-model/mixed-model assembly lines that we found in the literature. Valid examples for our purposes are those that include task times, precedencies, scheduled production and available time, especially if the authors were interested in assigning tasks to constant workstations—although, in most cases, their objective functions are different from ours. First, original task times are transformed into unit workloads following the algorithm described in Section 4 on an electronic spreadsheet. Then, our computer program captures input data from the spreadsheet and finds the optimal solution according to our objective function.

The first instance [11] includes 15 tasks and 3 models. Four specific tasks (2, 6, 8, 13) should be assigned to the same workstation for each model. The problem was solved on a PC with an Intel®Core™ i7-8700 processor at 3.2 GHz with 8 GB of RAM for $\beta_1 = \beta_2 = 1$ and the computer needed 0.06 s to find the optimal solution (Table 11). This instance was solved again for $\beta_1 = 1$ and $\beta_2 = 4$ (Table 11) with the same computer processor (CPU) time.

Table 11. Optimal assignment of tasks to workstations to the instance from [11]. Figures represent tasks (i).

Workstation (j)	Model 1 ($\beta_2 = 1$)	Model 2 ($\beta_2 = 1$)	Model 3 ($\beta_2 = 1$)	Models 1, 2 and 3 ($\beta_2 = 4$)
1	1–2	1–2–5	1–2–3	1–2
2	3–4–9	3–4–9	4–5–9	3–4
3	5–6–7	6–7	6–7	6–7
4	8–10–12	8–10–12	8–10–12	5–10
5	11–13–14–15	11–13–14–15	11–13–14–15	8–9–12–14
6	-	-	-	11–13–15

In the second instance, Thomopoulos presents a line with 19 tasks where three models are to be assembled for a production schedule of 120, 60 and 40 units of each model, respectively, during each shift [29]. The available time is not mentioned in the paper but Thomopoulos assumes that the line has three workstations. For a theoretical minimum number of workstations equal to three and taking into account the amount of assembly work to be performed, the available time must be at least 414 min (this value is also mentioned in [29]). For this reason, we have repeated the same problem for values of assembly time between 414 and 480 min (one shift) for $\beta_1 = \beta_2 = 1$, as shown in Table 12.

Table 12. Results of the computational experience with an instance from [29].

Available Time (min)	Theoretical Minimum Number of Workstations	Number of Workstations in Multi-Model Line	Tasks Assigned to Constant Workstations	Efficiency	Average (*) CPU Time (s)
480	2.59	3	19	0.86	0.09
458	2.71	3	19	0.90	0.09
436	2.85	3	18 (Except task 18)	0.95	0.12
414	3.00	4	19	0.75	0.14

* After 10 replications of each case. CPU time stands for computer microprocessor time.

The third instance involves 35 tasks and 6 models [49]. The problem was solved twenty times for $\beta_1 = 1$ and $\beta_2 = 7$ and the solver found different solutions (one example is Table 13) but all solutions were of the same quality because all required 14 workstations and in all cases, 34 of the 35 tasks could be assigned to the same workstations for the six models. Task 3240 has incompatible different precedencies for different models, and, in consequence, it behaves as two different tasks. Thus, we can consider that task 3240 was correctly assigned too.

Table 13. A solution to problem from [49]. Figures represent tasks.

Workstation	Model 1, 2, 5 and 6	Model 3 and 4
1	3230-3255-3265-3285	3230-3240-3255-3265-3285
2	3245-3290-3335	3245-3290-3335
3	3235-3305-3360	3235-3305-3360
4	3275-3295-3300	3275-3295-3300
5	3310-3315-3340	3310-3315-3340
6	3250-3280-3365	3250-3280-3365
7	3330-3370	3330-3370
8	3345	3345
9	3320-3375-3380	3320-3375-3380
10	3385-3405	3385-3405
11	3325-3390-3395	3325-3390-3395
12	3350	3350
13	3240-3270-3355	3270-3355
14	3400	3400

6. Discussion

In our experiment in Section 4, we compare the solution of a multi-model assembly line balancing problem that we can achieve through our method with the solution reached with the creation of a combined model analyzed as a single-model line. We find that the classical approach of a combined model results in an inapplicable multi-model line. Furthermore, the results in Table 9 are better than those achieved with a combined model in Table 4 because fewer workstations are necessary and there is less idle time. The method computes the theoretical minimum number of workstations and the fraction of the production time that the line should be devoted to each model in order to achieve the desired output. As a drawback, this procedure computes the required capacity, but it does not assign tasks to workstations. This can be performed, for each model, using well-known heuristics or linear programming but by always replacing task times with workload and replacing cycle time with station capacity (typically, one operator). To go one step further, we have coded a piece of software in Python, which takes advantage of Google's solver, in order to assign tasks to workstations, with the same number of workstations for each model and even the same tasks performed at each station for each model, which gives stability to the line. The approach based on workload allows studying the three lines together, as shown in Section 4.5, leading to the solution in Table 10. This solution presents a line operated by a constant number of operators where tasks are consistently assigned to the same workstations across the different models. The resulting line can produce what was expected; it is flexible (different models, different quantities) and the resulting task distribution is directly applicable, with balanced workstations. All these characteristics agree with the principles of LM: JIT production, respect for people and reduced waste of resources.

In our first additional instance, the original paper [11] was able to find, through a heuristic method (based on a procedure to balance robotic lines), a line operated by five people. For $\beta_1 = \beta_2 = 1$, our software found an optimal solution with five stations, where thirteen (out of fifteen) tasks—except tasks 3 and 5—were assigned to constant workstations for all the models. The solutions found in the original paper [11], although its objective function had a different purpose, correctly allocated tasks 2, 6, 8 and 13 to five workstations. Execution time cannot be compared since the paper searched for another objective. With $\beta_1 = 1$ and $\beta_2 = 4$, we found an optimal solution where all tasks were assigned to constant workstations at the expense of adding a sixth workstation. In consequence, we correctly solved this instance with five or six operators.

In our second additional instance, we played with an instance taken from Thomopoulos' paper [29]. We were able to determine the optimal lines for each value of the available time in the shift (see Table 12) and we repeated each experiment ten times. We were surprised by the speed of the solver (values below 1 s) and noticed that the time increased when the available time in the shift decreased (it was more difficult to allocate tasks to workstations and reach an optimal solution). Although Thomopoulos assumes that "an operator who is assigned an element on one model may not be assigned that same element on other models" [29], our solutions were able to assign all nineteen tasks to the same workers except one task on one occasion (when the available time was 436 min). This would be possible with a fourth operator (thus allowing a greater value for β_2) at the expense of losing efficiency. That fourth operator is always necessary when the available time drops to 414 min and then all 19 tasks can be correctly assigned again. Since Thomopoulos' objective function was to minimize the balance delay among stations, no further comparison is possible between solutions.

With respect to our third additional instance, since 13 of the 35 tasks must be assigned to the same workstation for all the models due to their common equipment [49], it is a good candidate for our computer program (the code is available at Supplementary Materials: https://github.com/jordifortuny/assembly_line). Since the original paper does not mention the available time during the workday, we solved the problem for an eight-hour shift (28,800 s). The CPU time was two minutes and the solution involved 10 workstations for each model (since the theoretical minimum number of workstations or total multi-model unit workload *TWu* is 8.755, the efficiency of the resulting line is 0.97). We did not incentivize the program ($\beta_1 = \beta_2 = 1$) to assign more tasks to the same workstations for each model and, as a result, our program found an optimal solution with 31 tasks (out of 35) assigned to the same stations for all six models. Cevikcan et al. used 12 workers to complete the task (in fact, they designed a u-shaped cell instead of a straight line) while correctly assigning the required 13 operations [49]. To assess the validity of our algorithm, we tested our program under more stringent conditions and reduced the available time. We found out that there were no feasible solutions with twelve operators under 25,000 s, and, finally, we set the available time to 21,500 s (because the theoretical minimum number of workstations was still twelve) and $\beta_2 = 7$ (greater than the number of models in order to reward tasks assigned to the same workstations for all the models). In a series of 20 repetitions, the computer needed between 7 min 39 s and 48 min 26 s to reach the optimal solutions. Since this instance is based on a real assembly line, we consider that these values may be acceptable for real applications of the same size. We used 14 workers (Table 13) but we were able to assign all 35 tasks in a consistent manner, as explained in Section 5.

7. Conclusions

In this paper, a novel approach to compute the necessary workforce in manual assembly systems was developed and then it was applied to a multi-model assembly line following the principles of LM.

Multi-model lines have not been researched much and there is not a general procedure for multi-model lines as those developed for the SMALBP.

Our method is based on the fact that verifying the balance between workload and resource capacity is necessary to assure that a production process is feasible in the considered time lapse. For this reason, we define the concepts of unit workload and capacity in order to avoid using time magnitudes (such as cycle time) since we assume that they are unknown. In Section 3, we derived the necessary equations to compute the required workforce without resorting to temporal magnitudes and used them to solve a small example. The same framework can be applied to other manufacturing strategies and thus, in Section 4, the method was applied to a multi-model assembly line.

While many approaches in the assembly line balancing literature consider either a cycle time for each model [50] or a common cycle time as given, we state that there are practical cases in which a company will only have a daily schedule based on market demand. Therefore, computing an average cycle time may lead to inapplicable solutions. In our method, the need for operators is modeled in terms of workload and capacity, not on assembly times. The practical consequence is that it is not necessary to know any cycle time to solve the problem. Thus, this study furnishes practitioners with a better understanding of the relations between several variables to design, implement and control processes. In addition, it can be easily implemented (e.g., Tables 4–8 on an electronic spreadsheet) and quickly modified when the production mix changes. Then, the line can be balanced for each model using either well-known heuristics for single-model lines to find a feasible solution or linear programming as we did in this paper to be sure that we found the optimal solution. In all cases, the unknown cycle time is not necessary.

Finally, our procedure was completed by means of a computer program coded in Python with the help of the OR-Tools library (which results in few lines of code, mainly devoted to manage data input and output). This allowed us to study all assembly lines together, based on linear constraints, in order to find an optimal solution—not just a feasible one—which minimized the number of workstations while aiming at a common number of workstations and even with a constant assignation of tasks to workstations, according to our interest. Besides not needing a previously defined cycle time, the result can be directly applied because real tasks are used instead of "combined" tasks.

Using our algorithms will allow operations managers to quickly staff and balance (or re-balance) their multi-model assembly in order to guarantee a smooth operation (with few changes, if any, in the number of operators or in the tasks assigned to each operator during the workday). This will also ensure the production, in small batches, of the scheduled quantities and models to satisfy market demand. Results show that our approach is a suitable option for balancing multi-model lines, without great needs of assets (where light objects such as cell phones are assembled), and with labor (the number of workstations) being the only resource considered, and for considering the setup cost as a fixed cost because lot sizes are based on demand, not on cost minimization. These premises also constitute the limitations of our work and could be the basis of further research. Since we have been able to use our transformed data (Wu_{ik}) in methods originally intended for assembly times and cycle times (such as solving the SMALBP with linear programming), it would be interesting to study whether our approach can be used in combination with other extant techniques.

Although our practical experience is related to manual assembly lines, a new trend in manufacturing, as part of the so-called Fourth Industrial Revolution or Industry 4.0 [51], is the development of hybrid assembly systems where robots (called *cobots*—meaning collaborative robots) work alongside people, in contrast with traditional arrangements where robots work in isolation for safety reasons. In the ALBP recent literature, there is an upward trend on robotic lines [52]. However, the research on multi-model assembly lines with robots is scarce [53]. In further research, our methodology could be extended to robotic lines. Currently, we assume that any worker can perform any task—which is based on human flexibility and on the cross-training practice in LM—while research on robotic lines requires selecting what tasks can be assigned to people and what tasks can be assigned to robots. In addition, when several types of robots are available, it is necessary to assign the most efficient type for each task to the corresponding workstation [54], while we might say that the number of "robot" types allocated to the line that we solved in this paper is one because all operators are equal [24].

Supplementary Materials: The computer code is available online at https://github.com/jordifortuny/assembly_line.

Author Contributions: Conceptualization, P.R.-d.-A.-L., J.F.-S. and L.C.-A.; methodology, P.R.-d.-A.-L., J.F.-S., L.C.-A. and J.F.-P.; software, J.F.-P.; validation, J.F.-P.; writing—original draft preparation, P.R.-d.-A.-L., J.F.-S, L.C.-A. and J.F.-P.; writing—review and editing, J.F.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mor, R.; Bhardwaj, A.; Singh, S.; Sachdeva, A. Productivity gains through standardization-of-work in a manufacturing company. *J. Manuf. Technol. Manag.* **2019**, *30*, 899–919. [CrossRef]
2. Krafcik, J.F. Triumph of the lean production system. *Sloan Manag. Rev.* **1988**, *30*, 41–52.
3. Womack, J.P.; Jones, D.T.; Roos, D. *The Machine That Changed the World*; Rawson Associates—Simon & Schuster: New York, NY, USA, 1990.
4. Sugimori, Y.; Kusunoki, K.; Cho, F.; Uchikawa, S. Toyota production system and Kanban system: Materialization of just-in-time and respect-for-human system. *Int. J. Prod. Res.* **1977**, *15*, 553–564. [CrossRef]
5. Monden, Y. *Toyota Production System: Practical Approach to Production Management*; Industrial Engineering and Management Press, Institute of Industrial Engineers: Norcross, GA, USA, 1983.
6. Shah, R.; Ward, P.T. Defining and developing measures of lean production. *J. Oper. Manag.* **2007**, *25*, 785–805. [CrossRef]
7. Cuatrecasas, L.; Fortuny-Santos, J.; Vintro, C. The operations-time chart: A graphical tool to evaluate the performance of production systems—From batch-and-queue to lean manufacturing. *Comput. Ind. Eng.* **2011**, *61*, 663–675. [CrossRef]
8. Gjeldum, N.; Salah, B.; Aljinovic, A.; Khan, S. Utilization of Industry 4.0 related equipment in assembly line balancing procedure. *Processes* **2020**, *8*, 864. [CrossRef]
9. Ghosh, S.; Gagnon, R.J. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *Int. J. Prod. Res.* **1989**, *27*, 637–670. [CrossRef]
10. Colim, A.; Faria, C.; Braga, A.C.; Sousa, N.; Rocha, L.; Carneiro, P.; Costa, N.; Arezes, P. Towards an ergonomic assessment framework for industrial assembly workstations—A case study. *Appl. Sci.* **2020**, *10*, 3048. [CrossRef]
11. Bukchin, J.; Dar-El, E.M.; Rubinovitz, J. Mixed model assembly line design in a make-to-order environment. *Comput. Ind. Eng.* **2002**, *41*, 405–421. [CrossRef]
12. Cuatrecasas-Arbós, L.; Fortuny-Santos, J.; Ruiz-de-Arbulo-López, P.; Vintró-Sanchez, C. Monitoring processes through inventory and manufacturing lead time. *Ind. Manag. Data Syst.* **2015**, *115*, 951–970. [CrossRef]
13. Boysen, N.; Fliedner, M.; Scholl, A. Assembly line balancing: Which model to use when? *Int. J. Prod. Econ.* **2008**, *111*, 509–528. [CrossRef]
14. Ohno, T. *Toyota Production System: Beyond Large-Scale Production*; Productivity Press: New York, NY, USA, 1988.
15. Bukchin, J.; Darel, E.; Rubinovitz, J. Team-oriented assembly systems design: A new approach. *Int. J. Prod. Econ.* **1997**, *51*, 47–57. [CrossRef]
16. Ben-Gal, I.; Bukchin, J. The ergonomic design of workstations using virtual manufacturing and response surface methodology. *IIE Trans.* **2002**, *34*, 375–391. [CrossRef]
17. Seppälä, P.; Klemola, S. How do employees perceive their organization and job when companies adopt principles of lean production? *Hum. Factors Ergon. Manuf.* **2004**, *14*, 157–180. [CrossRef]
18. Cevikcan, E. An optimization methodology for multi model walking-worker assembly systems: An application from busbar energy distribution systems. *Assem. Autom.* **2016**, *36*, 439–459. [CrossRef]

19. Chutima, P.; Suphapruksapongse, H. Practical assembly-line balancing in a monitor manufacturing company. *Thammasat Int. J. Sci. Technol.* **2004**, *9*, 62–70.
20. Erel, E.; Sarin, S.C. A survey on the assembly line procedures. *Prod. Plan. Control* **1998**, *9*, 414–434. [CrossRef]
21. Roser, C.H. Line Balancing Part 6—Tips and Tricks for Balancing. Available online: https://www.allaboutlean.com/line-balancing-6/ (accessed on 24 September 2020).
22. Nagi, M.; Chen, F.F.; Wan, H.-D. Throughput rate improvement in a multiproduct assembly line using lean and simulation modeling and analysis. *Procedia Manuf.* **2017**, *11*, 593–601. [CrossRef]
23. Agpak, K.; Gökçen, H. Assembly line balancing: Two resource constrained cases. *Int. J. Prod. Econ.* **2005**, *96*, 129–140. [CrossRef]
24. Corominas, A.; Pastor, R.; Plans, J. Balancing assembly line with skilled and unskilled workers. *Omega* **2008**, *36*, 1126–1132. [CrossRef]
25. Sivasankaran, P.; Shahabudeen, P. Literature review of assembly line balancing problems. *Int. J. Adv. Manuf. Technol.* **2014**, *73*, 1665–1694. [CrossRef]
26. Roberts, S.D.; Villa, C.D. On a multiproduct assembly line balancing problem. *AIIE Trans.* **1970**, *2*, 361–364. [CrossRef]
27. Becker, C.; Scholl, A. A survey on problems and methods in generalized assembly line balancing. *Eur. J. Oper. Res.* **2006**, *168*, 694–715. [CrossRef]
28. Thomopoulos, N.T. Line balancing-sequencing for mixed-model assembly. *Manag. Sci.* **1967**, *14*, B59–B75. [CrossRef]
29. Thomopoulos, N.T. Mixed model line balancing with smoothed station assignments. *Manag. Sci.* **1970**, *16*, 593–603. [CrossRef]
30. Buxey, G.M.; Slack, N.D.; Wild, R. Production flow line system design—A review. *AIIE Trans.* **1973**, *5*, 37–48. [CrossRef]
31. Eryuruk, S.H.; Kaloglu, F.; Baskak, M. Assembly line balancing in a clothing company. *Fibres Text. East. Eur.* **2008**, *16*, 93–98.
32. Al-Zubaidy, S.S.; Alrazaq, F.F.A. Multi-model production and assembly line balancing (Caravans production workshop). *J. Univ. Babylon Eng. Sci.* **2013**, *21*, 1301–1312.
33. Kabir, M.A.; Tabucanon, M.T. Batch-model assembly line balancing: A multi attribute decision making approach. *Int. J. Prod. Econ.* **1995**, *41*, 193–201. [CrossRef]
34. Roser, C.H. Line Balancing Part 2—Duration of Tasks. Available online: https://www.allaboutlean.com/line-balancing-2/ (accessed on 24 September 2020).
35. Pastor, R.; Andres, C.; Duran, A.; Perez, M. Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. *J. Oper. Res. Soc.* **2002**, *53*, 1317–1323. [CrossRef]
36. Qu, S.; Jiang, Z. A memetic algorithm approach for batch-model assembly line balancing problem of sub-block in shipbuilding. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2014**, *228*, 1290–1304. [CrossRef]
37. Berger, I.; Bourjolly, J.M.; Laporte, G. Branch-and-bound algorithms for the multi-product assembly line balancing problem. *Eur. J. Oper. Res.* **1992**, *58*, 215–222. [CrossRef]
38. Zhao, C.; Li, J. Analysis and improvement of multi-product assembly systems: An application study at a furniture manufacturing plant. *Int. J. Prod. Res.* **2014**, *52*, 6399–6413. [CrossRef]
39. Aqlan, F.; Al-Fandi, L. Prioritizing process improvement initiatives in manufacturing environments. *Int. J. Prod. Econ.* **2018**, *196*, 261–268. [CrossRef]
40. Coleman, J.B.; Vaghefi, R.A. Heijunka (?): A key to the Toyota Production System. *Prod. Invent. Manag. J.* **1994**, *35*, 31–35.
41. Cuatrecasas, L. *Ingeniería de Procesos y de Planta*; Profit Editorial: Barcelona, Spain, 2017.
42. Hirano, H. *JIT Implementation Manual*; CRC Press: Boca Raton, FL, USA, 2009; Volume 4.
43. Meyers, F.E.; Stewart, J.R. *Motion and Time Study for Lean Manufacturing*; Prentice Hall: Upper Saddle River, NJ, USA, 2002.
44. Baybars, I. A survey of exact algorithms for the simple assembly line balancing problem. *Manag. Sci.* **1986**, *32*, 909–932. [CrossRef]
45. Matsui, Y. An empirical analysis of just-in-time production in Japanese manufacturing companies. *Int. J. Prod. Econ.* **2007**, *108*, 153–164. [CrossRef]

46. Koo, P.-H. A new self-balancing assembly line based on collaborative ant behavior. *Appl. Sci.* **2020**, *10*, 6845. [CrossRef]
47. Faganello, R.; Amaral, V. Logic-based Benders decomposition for the heterogeneous fixed fleet vehicle routing problem with time windows. *Comput. Ind. Eng.* **2020**, *148*, 106641.
48. Valdron, M.; Pu, K.Q. Data Driven Relational Constraint Programming. In Proceedings of the IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI), Las Vegas, NV, USA, 11–13 August 2020; pp. 156–163. Available online: https://doi.ieeecomputersociety.org/10.1109/IRI49571.2020.00030 (accessed on 24 September 2020).
49. Cevikcan, E.; Durmusoglu, M.B.; Unal, M.E. A team-oriented design methodology for mixed model assembly systems. *Comput. Ind. Eng.* **2009**, *56*, 576–599. [CrossRef]
50. Erel, E.; Gokcen, H. Shortest-route formulation of mixed-model assembly line balancing problem. *Eur. J. Oper. Res.* **1999**, *116*, 194–204. [CrossRef]
51. Fortuny-Santos, J.; Ruiz-de-Arbulo López, P.; Luján-Blanco, I.; Chen, P.K. Assessing the synergies between lean manufacturing and Industry 4.0. *Dir. Organ.* **2020**, *71*, 71–86. [CrossRef]
52. Chutima, P. Research Trends and Outlooks in Assembly Line Balancing Problems. *Eng. J.* **2020**, *24*, 93–134. [CrossRef]
53. Christensen, M.K.; Janardhanan, M.N.; Nielsen, P. Heuristics for solving a multi-model robotic assembly line balancing problem. *Prod. Manuf. Res.* **2017**, *5*, 410–424. [CrossRef]
54. Rubinovitz, J.; Bukchin, J.; Lenz, E. RALB—A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines. *CIRP Ann. Manuf. Technol.* **1993**, *4*, 497–500. [CrossRef]