Journal of Applied Engineering and Technological Science
Vol 6(1) 2024: 1-20

ON THE ASSEMBLY LINE BALANCING PROBLEM: A SIMPLIFIED PERSPECTIVE WITH THE PRECEDENCE MATRIX

Magdy Helal[1]*, Kaushik Nag[2], Rifat Ozdemir[3]
College of Engineering and Technology, American University of the Middle East, Kuwait
magdy.helal@aum.edu.kw

Received: 8 July 2024, Revised: 7 October 2024, Accepted: 9 October 2024
**Corresponding Author*

ABSTRACT
The assembly line balancing problem (ALBP) has been an attractive research topic for decades, but the industrial application of the research findings remains limited. This can be attributed to the complexity of solution methods, restrictive assumptions, and the numerous variants of the problem in real-world settings. This article proposes using the precedence matrix as a foundation for developing efficient analysis frameworks for ALBP. We introduce algorithms for constructing the precedence matrix and using it to guide the development of feasible assembly line designs. Implemented in a spreadsheet model, the matrix framework provides a straightforward tool for organizing the balancing problem data, managing inputs to balancing heuristics, and ensuring feasible solutions. It is flexible and can be integrated with various balancing methods. In this work, we utilized simple priority rules for line balancing to demonstrate the potential of the matrix model. A performance comparison of the balancing rules showed that optimal solutions were achieved in over 60% of the test problems. The results demonstrate that the proposed matrix model is an efficient tool for production managers to evaluate potential line designs. The goal of this work is to help bridge the gap between research and practice in assembly line balancing.
Keywords: *Assembly Line Balancing, Precedence Matrix, Spreadsheets, Priority Rules*

1. Introduction

The assembly line is a flow production system where products are built on a series of workstations, with each station performing a partial set of tasks needed to complete the product. Assigning the tasks to the workstations is constrained by a set of precedence relations, which state the required technological order among the tasks. Each workstation is allocated a time window to perform its assigned tasks, that imposes an additional constraint. This time window, determined based on the required production rate, is commonly called the cycle time. Given the precedence constraints and the required cycle time, the goal is to distribute all the tasks to workstations such that the total time at each station does not exceed the cycle time, all precedence constraints are satisfied, and the number of needed workstations is minimized. This is known as the assembly line balancing problem (ALBP). It is a nonpolynomial (NP) hard combinatorial optimization problem (Ahmadi & van der Rhee, 2023; Lapierre et al., 2006) for which exact mathematical solution procedures become interactable as the number of assembly tasks increases.

The ALBP was characterized during the 1950s and has been a popular research topic since then. Assembly lines are used in mass production systems, e.g., for the assembly of consumer electronics, appliances and, most famous of all, the assembly of cars. In fact, the first assembly line was developed by Ford more than a century ago (Nicholas, 2018). Line balancing is crucial in the industrial sector as it directly impacts production efficiency, resources utilization and the overall operational performance of the organization (Hu et al., 2015; Rahman et al., 2023) and is key for sustainable competitiveness in a fast-paced industrial environment.

The literature on ALBP solution methods and system configurations is extensive. There is a good number of surveys on the line balancing, see for example Boysen et al. (2022), Jiao et al. (2021), Eghtesadifard et al. (2020), Chutima (2020), and Battaïa & Dolgui (2013). However, there exists a considerable gap between the academic discussions and practical applications in the line balancing area (Boysen et al., 2022; Katiraee et al., 2023; Pearce et al., 2019). Although many exact solution methods and advanced heuristic algorithms are available, simpler solution approaches are more commonly used in industry (Gonzales-Rodriguez, 2022). A review (Chutima, 2020) found that, between 2014 and 2018, only around 13% of research referenced real-world industry applications of line balancing techniques. It was similar situations during

1970s and 1980s (Boysen et al., 2022; Pearce et al., 2019). Based on their field experiences, Boysen et al. (2022) stated that only a small percentage of companies applied the advanced balancing procedures. They considered that an "*ongoing challenge*" for the line balancing research efforts. In addition, human workers and manual operations are common on assembly lines (Ozdemir et al., 2021; Şahin & Kellegöz, 2024; Sotskov, 2023) and process managers still relay on their experiences in making decisions regarding assembly line designs, which may sometimes lack proper validation (Manoria et al., 2012; Rahman et al., 2023).

Pearce et al. (2019) argued that the limited implementation of balancing algorithms in industry could be attributed to management's approval of good feasible solutions instead of seeking optimal ones due to computational difficulties, not knowing of theoretical algorithmic methods, financial resource limitations, and, sometimes, organizational resistance to change. The numerous possibilities in the assembly line settings and the restrictive assumptions in the solution methods may make it harder to fit theoretical solutions to the real-world applications (Katiraee et al., 2023; Pearce et al., 2019). The previously mentioned have counted dozens of variants of the ALBP, categorized by line layout, product models, task times variability, problem objectives, modes of operations, and other considerations.

Limited industrial applications will further widen the gap between research and practice. The line balancing problem will continue to get more complex with the advances in manufacturing technologies and the introduction of more constraints and objectives (Ahmadi & van der Rhee, 2023; Chutima, 2022; Schlueter & Ostermeier, 2022). We argue that there is a real need to develop simpler, more practical analysis frameworks to support practitioners in designing assembly lines. In line with that, this article proposes using the precedence matrix as a foundation for developing intuitive approaches to studying the ALBP. The precedence matrix is a storage structure of precedence relations data in a way that facilitates the estimation of many problem-related indices and features. We introduce algorithms for constructing the precedence matrix and using it to guide the balancing heuristics in developing feasible task assignments to the workstations. The matrix model can be adopted in the development of balancing heuristics to simplify the process and manage the inputs to the execution of the heuristics. We also describe a spreadsheet implementation of the precedence matrix framework, integrated with selected balancing rules, to demonstrate its potential.

2. Literature Review

2.1 The Assembly Line Balancing Problem

The ALBP is traditionally categorized into two basic types. Type I seeks to determine the minimum number of needed workstations to perform the assembly job, given the required production rate (cycle time). Type II seeks to determine the highest production rate (minimum cycle time) to perform the assembly job on a given set of workstations. Type I is more common in research, and it is the basic formulation, whereas Type II problems can be approached by using Type I solution methods in an iterative fashion (Bao et al., 2023; Fathi et al., 2018; Lapierre et al., 2006). Type I problem implies that a new assembly line is to be designed to produce a new product, whereas Type II assumes an existing line that needs to be modified to accommodate production or design changes. The basic objective function of the problem is defined as follows, where m is the number of workstations, C cycle time, t_i processing time of task i, and n the number of assembly tasks:

$$Min\ Z = m\,C - \sum_{i=1}^{n} t_i$$

Since the sum of processing times is constant, optimizing the objective function can be approached by minimizing the number of stations for a given cycle time (Type I) or minimizing the cycle time for a given number of stations (Type II). In both cases, optimizing the objective function is equivalent to minimizing total idle time on the workstations (Erel & Sarin, 1998). When the sum of task times at each station is equal, the line is considered balanced. The total time at each station must be less than or equal to the required cycle time. A balanced assignment of tasks to workstations eliminates bottlenecks and contributes to achieving smoother workflow and

higher efficiency, that leads to better cost performance and higher levels of leanness in the system (Nicholas, 2018).

Precedence relations are hard constraints that must be satisfied. They are the main characteristic of the ALBP. For n tasks in an assembly job, with r precedence relations among them, there are $n!/2r$ different task sequences that need to be evaluated, which is a prohibitive large number in most industrial problems (Erel & Sarin, 1998). Precedence relations can be represented graphically by the precedence diagram, which is a directed graph with nodes as tasks, and arrows as precedence relations. The precedence diagram was first described by Salveson (1955) and Jackson (1956) and used for manual solutions. While it is used for illustration purposes in textbooks and some research articles, solving the balancing problem directly on the diagram gets impractical as the problem size increases. Figure 1 shows the precedence diagram for a 12-task assembly job. Processing times of each task are shown on the top of each node.

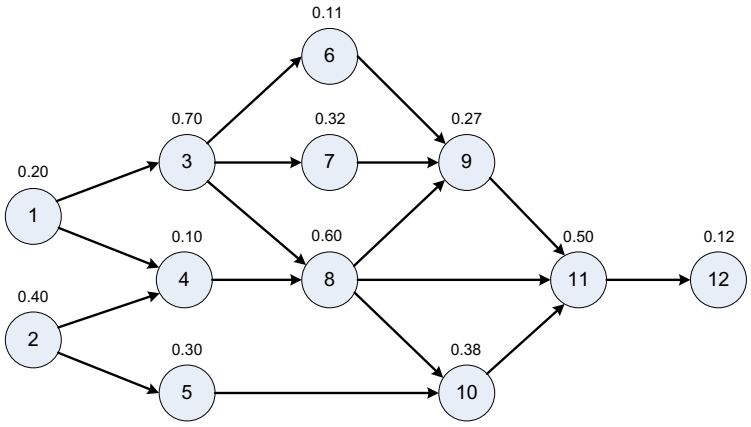
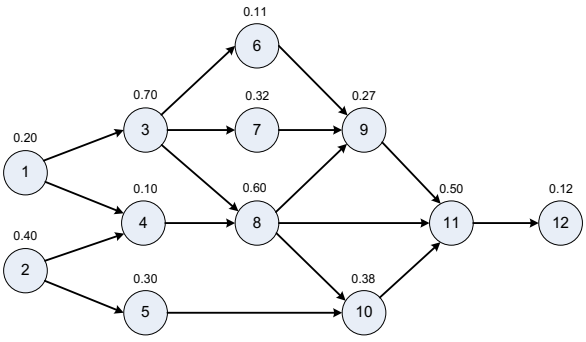


Fig. 1. Precedence diagram for a 12-task assembly job

2.2 The Precedence Matrix

Alternatively, the precedence relations can be stored in matrix format. Organizing the precedence relations in a matrix format was utilized by some researchers to develop balancing heuristics. Hoffmann's heuristic (Hoffmann, 1963) used a matrix that contained an entry of 1 if there was an immediate precedence relation between two tasks, and 0 otherwise. It was a square matrix of zeros and ones, with the number of rows and columns equal to the number of tasks. Only immediate relations (e.g., task 4 immediately follows task 1 in Figure 1) were indicated on the matrix while non-immediate relations (e.g., task 8 follows task 1 through task 4 in Figure 1) were identified logically. In Hoffmann's heuristic, the sum of a column i indicated the number of predecessors of task i. Columns with a sum of zero indicated tasks without predecessors and thence top candidates for assignment to the current workstation. Tasks were assigned to workstations one at a time going with the columns' sums in ascending order, with the matrix updated after each task assignment. The matrix simplified the solution process, but the algorithm tended to concentrate idle times at the later stations (Talbot et al., 1986).



Task	1	2	3	4	5	6	7	8	9	10	11	12
1		0	1	1	0	1	1	1	1	1	1	1
2	0		0	1	1	0	0	1	1	1	1	1
3	-1	0		0	0	1	1	1	1	1	1	1
4	-1	-1	0		0	0	0	1	1	1	1	1
5	0	-1	0	0		0	0	0	0	1	1	1
6	-1	0	-1	0	0		0	0	1	0	1	1
7	-1	0	-1	0	0	0		0	1	0	1	1
8	-1	-1	-1	-1	0	0	0		1	1	1	1
9	-1	-1	-1	-1	0	-1	-1	-1		0	1	1
10	-1	-1	-1	-1	-1	0	0	-1	0		1	1
11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1		1
12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	

Fig. 2. Precedence diagram (left) and precedence matrix for a 12-task assembly job

In (Moodie, 1964), the precedence matrix was used to represent both task predecessors and followers. Unlike Hoffmann's, Moodie's matrix included all immediate and non-immediate relations. For each row *i* and column *j*, an entry of +1 indicated that task *i* precedes task *j*, an entry of -1 indicated that *i* follows *j*, while an entry of 0 indicated no relation. Figure 2 shows Moodie's matrix for the 12-task diagram from Figure 1. Clearly, the entries below the shaded principal diagonal are mirror images of the entries above, but with opposite signs. This is redundant information.

A variant of Moodie's matrix was also described in which the matrix was split into an immediate followers' matrix and an immediate predecessor's matrix. These were not square matrices; each had columns equal to the maximum number of immediate followers or immediate predecessors. The two sub-matrices needed less computer memory than a single complete matrix. Moodie (1964) described this matrix as a more effective way to summarize precedence relations compared to the precedence diagram, although according to him, the matrix was constructed from the diagram. He compared popular balancing heuristics at that time including Hofmann's matrix heuristic, which was the only one used the matrix explicitly as a core component of the heuristic. Talbot & Patterson (1984) used the contents of the immediate predecessors' matrix to provide inputs for feasibility checking in an integer programming formulation, but no description was given of how it was incorporated. Rashid et al. (2012) also described the matrix as an effective way to represent precedence constraints where they used a square matrix as in Moodie's but only immediate predecessors were included on one side of the diagonal. However, no specific use was reported.

In (Chiang, 1998), the immediate predecessors' matrix was used to check feasibility in a tabu-search heuristic, where a shortest-rout algorithm (Warshall's algorithm) was employed to find the transitive closure matrix through which the non-immediate predecessors were identified. Warshall's algorithm (Warshall, 1962) finds the shortest routes between nodes in directed networks. It was called in each iteration of the tabu-search to identify all predecessors of the current task, to ensure the feasibility of moving it between stations. This added a second layer of search to the tabu-search' search process and made it computationally more demanding.

In a similar approach, Nearchou (2007) proposed an evolutionary algorithm and used the immediate predecessor's matrix for feasibility checking. Nearchou described the construction of the matrix using a simple algorithm and then utilized Warshall's algorithm to establish the non-immediate predecessors. After each trial sequence of tasks was created it was checked against the precedence data to remove infeasibility. Hamta et al. (2013) followed Nearchou's matrix and simplified the matrix construction to use a simpler looping algorithm. The precedence data was also used to test the trial solutions in a particle swarm algorithm for solving the ALBP with sequence dependent setup times.

In summary, the precedence matrix has been recognized as an effective structure to store precedence relations in the ALBP. However, it is not often mentioned explicitly in literature. Based on our review, we identified only Nearchou (2007) and Hamta et al. (2013) who provided descriptions of how to construct the matrix, based on our review.

2.3 Solving the ALBP

Although all versions of the ALBP are NP-complete (Boysen et al., 2022), the problem has attracted researchers since its first definition, and dozens of exact and heuristic solution methods have been introduced. This included optimization methods such as integer programming, dynamic programming and branch and bound algorithms, and various implementations of meta-heuristics such as tabu search, simulated annealing, and genetic algorithm, and many others. A recent comprehensive review can be found in Boysen et al. (2022). In addition, several constructive procedures and many simpler priority rules have been utilized to solve the ALBP by generating feasible line designs efficiently. The priority rules were recognized for efficiency and flexibility. Several surveys of priority rules have been published over the years, see for example Fathi et al. (2018), Moreira et al. (2012), Capacho Betancourt (2008), Scholl & Voß (1997), and Talbot et al. (1986).

There are many line balancing priority rules. These simple heuristics rank assembly tasks in ascending or descending order of some parameter values, then assign the tasks to the

workstations, one at a time, starting with the top-ranked task. The priority rule methods are computationally efficient but can only check limited numbers of potential solutions. Among the most popular and successful of them is the Ranked Positional Weight (RPW) which was proposed by Helgeson & Birnie (1961). RPW assigns each task a weight equal to its processing time plus the processing times of all its follower tasks. Tasks are then assigned to the earliest workstation with sufficient time to accommodate them, given all predecessors of the task have already been assigned to the current or earlier stations. For their simplicity and efficiency, priority rules remained interesting, and several researchers have built on them to develop more structured and efficient heuristics. Boysen et al. (2022) identified "*a stream of research*" that developed priority rules-based procedures to obtain good feasible solution efficiently for specially characterized line balancing problems. RPW was frequently utilized in these efforts, as can be seen in, for example, Capacho Betancourt (2008), Pearce et al. (2019), and Çelik & Arslankaya (2023).

We utilize popular balancing rules in this work, incorporating them into the proposed matrix-based framework. Priority rules develop quick feasible solutions, even for very large problems ((Boysen et al., 2022), which aligns with our vision to provide practitioners with simple and effective analysis tools.

3. Objectives

In this work, we propose using the precedence matrix as a foundation for developing efficient line balancing frameworks for the simple ALBP. We introduce algorithms to construct the precedence matrix in a spreadsheet model, from the core problem data, namely the tasks, their processing times, and the immediate precedence relations among them. The matrix is then integrated with the selected balancing heuristic. The precedence matrix provides an intuitive and effective structure for storing ALBP data, facilitating the estimation of various problem-related parameters and indices, which contributes to developing balancing heuristics, and it supports efficient executing of the procedures and ensure solution feasibility.

Selected priority rules for line balancing will be implemented into the spreadsheet model to demonstrate its usability and potential. Spreadsheets are inexpensive, powerful data manipulation tools that are widely available. Current programming languages (e.g., Python's Openpyxl and Pandas) integrate spreadsheet objects into their libraries. The proposed framework is designed to provide industry practitioners with a flexible tool to quickly investigate and develop feasible solutions. This aligns with the reported practice by production managers in developing line design solutions, as discussed in Section 1. Our goal is to help bridge the gap between research and practice. In the following sections, we describe the structure and application of the proposed matrix framework.

4. Research Methods

4.1 Constructing the Precedence Matrix

In an ALBP, the minimum required data must include the list of tasks, task processing times, and the immediate precedence relations among them. The precedence matrix for a set of n tasks, is the square $n \times n$ matrix, in which the entries; M_{ij} at the intersection of row *i* and column *j* are defined as:

$$M_{ij} = \begin{cases} 1 & if\ task\ i\ precedes\ task\ j \\ 0 & otherwise \end{cases}$$

To simplify, zero entries will be represented by empty matrix cells. The immediate precedence relation between tasks i and j $(i, j = 1,2,3, \dots, n)$ is represented by the 2-tuple (i, j), which indicates that task i precedes task j. Tasks must be named such that if task i precedes task j, then i is smaller than j. This ensures all entries are on one side of the principal diagonal line of the matrix. In Figure 3, the set of immediate precedence relations for the 12-task network of Figure 1 are listed on the left. The precedence matrix is shown to the right of Figure 3. The principal diagonal line is highlighted in black. Rows and columns are labeled with the task numbers. Row 1, for example, shows that task 1 is followed by tasks 3, 4, 6, 7, 8, 9, 10, 11, and 12. According to the precedence diagram, tasks 3 and 4 are immediate followers, while the rest are non-

immediate followers of task 1. The immediate followers are highlighted in yellow, while the non-immediate followers are in grey. The tasks' processing times are listed to the right and at the bottom of the matrix.

Algorithm 1 follows the work of Hamta et al. (2013) and Nearchou (2007) and uses the minimum problem data mentioned above, to construct the precedence matrix into a spreadsheet. It also identifies the immediate and non-immediate followers for each task.

Immediate Precedence Relations ((i, j): i preceds j)	Task	1	2	3	4	5	6	7	8	9	10	11	12	Time
(1, 3), (1, 4)	1			1	1		1	1	1	1	1	1	1	0.20
(2, 4), (2, 5)	2				1	1			1	1	1	1	1	0.40
(3, 6), (3, 7), (3, 8)	3						1	1	1	1	1	1	1	0.70
(4, 8)	4								1	1	1	1	1	0.10
(5, 10)	5										1	1	1	0.30
(6, 9)	6									1		1	1	0.11
(7, 9)	7									1		1	1	0.32
(8, 9), (8,10), (8,11)	8									1	1	1	1	0.60
(9, 11)	9											1	1	0.27
(10, 11)	10											1	1	0.38
(11, 12)	11												1	0.50
-	12													0.12
	Time	0.20	0.40	0.70	0.10	0.30	0.11	0.32	0.60	0.27	0.38	0.50	0.12	

Fig. 3. Immediate predecessors relations list (left) and the precedence matrix for the 12-task network in Figure 1

4.2 Using the Precedence Matrix

The precedence matrix, as described, provides efficient support to the analysts for calculating the various problem features and indices, in executing line balancing heuristics, and in checking the feasibility of task assignments to workstations. The following subsections illustrate using the presentence matrix to estimate problem related indices and parameters. For better presentation, selected balancing rules are implemented within the spreadsheet model, for each of them the rule index is defined in terms of the rows and columns of the matrix. The performance of the rules will be compared and evaluated using benchmarking ALBP data.

Algorithm 1: Constructing the Precedence Matrix

I. Finding the immediate followers of tasks

1. For each present 2-tuple (i, j) do:
   a. Set $M_{i,j}$ = 1 (add 1 at the intersection of row i and column j)
   b. Stamp the intersection cell as immediate relation cell
   c. Increment the counter of the immediate followers of i by 1
   d. Add j to the list of followers of i
2. Next 2-tuple

II. Finding the non-immediate followers of tasks

3. For each task i = 1 to n do:
   a. If the number of immediate followers of i > 0 then
      i. For each present 2-tuple (j, k) do:
         1. For each task j in (j, k) where $j \neq i$ do:
            a. For each task f in the list of followers of i do:
               i. If $f = j$ then
                  1. Set $M_{i,k}$ = 1 (add 1 at the intersection of row i and column k)
                  2. Stamp intersection cell as non-immediate relation cell
            b. Next f
         2. Next j

ii. Next 2-tuple
4. Next $\boldsymbol{i}$
5. End

4.2.1 Assessing Difficulty of the ALBP

Common indicators of the ALBP difficulty include the order strength, flexibility ratio, and the WEST ratio. The order strength (OS) is the ratio of the number of precedence relations among the tasks to the maximum possible number of relations. If r is the number of existing precedence relations among n tasks, then $OS = 2r/(2(n+1))$. The value $n(n-1)/2$ is the maximum number of possible relations among n tasks. The value of OS affects the computational time to solve the problem; a value close to one implies a highly constrained problem and fewer number of alternative solutions (Chutima, 2022; Erel & Sarin, 1998; Mastor, 1970; Talbot et al., 1986). It is easier to see the meaning of OS with the precedence matrix, where each entry of 1 represents a precedence relation. Using Figure 3, OS is the sum of all entries in the matrix, divided by half the matrix size (above the principal diagonal) excluding the n cells on the principal diagonal, which have no entries. The matrix size, as a square matrix is $n\ x\ n = n^2$. Then, OS can be given as follows in terms of the rows and columns of the matrix:

$$OS = \frac{2(\sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij})}{n(n-1)}$$

For the example in Figure 3, OS = 0.697 which is relatively high. The flexibility ratio (FR) is the complement of the order strength, calculated as $FR = 1 - OS$. It uses the number of missing precedence relations. It indicates the volume of feasible sequences that can be generated for a problem. Larger FR values indicate less precedence constraints and more alternative solutions to evaluate.

Another complexity index was described by (Mastor, 1970) which is the average number of tasks per workstation, denoted by the WEST ratio. We will call it Task-to-Station Ratio (TSR) in this article. When an assembly line uses more workstations, the number of tasks per station (TSR) is small and there are fewer combinations of the tasks to assign to stations. In this case, balancing heuristics can be faster due to the limited search space, but solutions are more likely to include significant idle times. On the other hand, with smaller numbers of workstations, TSR increases and there will be more possible combinations of tasks and alternative solutions. In that case, the balancing heuristics will require longer computational times. TSR requires the number of stations to be known in advance, which makes it more relevant to the Type II problems. The lower bound on the number of workstations can be used as an estimate of the line length (Driscoll & Thilakawardana, 2001) to use TSR in Type I problems. A lower bound on the number of the workstations; m_{min} is the integer value equal to or greater than the quotient of the total processing time and the cycle time, C (Capacho Betancourt, 2008), given by:

$$m_{min} = \left\lceil \frac{\sum_{i=1}^{n} t_i}{C} \right\rceil$$

TSR can then be estimated as follows:

$$TSR = \frac{n}{\left\lceil \frac{\sum_{i=1}^{n} t_i}{C} \right\rceil}$$

4.2.2 Line Balancing with The Priority Rules

To provide a working prototype of the proposed framework, selected priority rules are implemented in the spreadsheet model. The more popular rules according to available reviews (Capacho Betancourt, 2008; Fathi et al., 2018; Moreira et al., 2012; Scholl & Voß, 1997; Talbot et al., 1986) are included, which are defined below. Using priority rules is meant for simplicity of presentation. Besides, the matrix-based analysis framework with priority rules can be an attractive tool to practitioners, given the discussion in Section 1, where managers in practice would be

satisfied with quick feasible solutions. It should be noted that other, more structured balancing heuristics can be integrated with the matrix model well.

We utilize the matrix framework to perform a comparison of the priority rules. Ten rules are used; the popular RPW-based rules (we call it maximum positional weight for naming consistency), the number of followers-based rules, in addition to priority rules using the earliest and latest stations that a task can be assigned to and the task slack, are included. In the following list of priority rules, their definitions are made in terms of the rows and columns of the precedence matrix, where i refers to the tasks listed in the rows of the matrix and j refers to the tasks listed in the columns, where i and $j = 1,2,\dots,n$.

1. **Maximum Number of Followers (MaxF) rule** counts the numbers of followers of each task, ranks them in descending order, and then assigns tasks to the workstations starting with the maximum value. Let F_i be the set of followers of task i, then its number of followers; $|F_i|$, is the sum of all entries in row i in the matrix

$$|F_i| = \sum_{j=i+1}^{n} M_{ij}$$

2. **Maximum Number of Immediate Followers (MaxIF) rule** counts only the immediate followers of the task. In the matrix construction algorithm, the immediate and the non-immediate precedence relations are distinguished from each other. The number of immediate followers; $|IF_i|$ of i is the sum of all entries in row i if they are tagged as immediate in the matrix:

$$|IF_i| = \sum_{j=i+1}^{n} M_{ij} \quad if\ M_{ij}\ is\ tagged\ as\ immediate$$

3. **Maximum Number of Non-Immediate Followers (MaxNIF) rule** counts the non-immediate followers of the task. The number of non-immediate followers; $|NIF_i|$, of task i is the sum of all entries in row i, if they are tagged as non-immediate:

$$|NIF_i| = \sum_{j=i+1}^{n} M_{ij} \quad if\ M_{ij}\ is\ tagged\ as\ non-immediate$$

4. **Maximum Positional Weight (MaxPW) rule** calculates the PW of each task, ranks them in descending order, and then assigns tasks to workstations starting with the maximum PW value. If F_i is the set of followers of task i, then $PW_i = t_i + \sum_{h \in F_i} t_h$, where t_i is the process time of i. In the matrix format, PW_i is equal to the process time of task i plus the sum of products of row i and the row containing the tasks' process times:

$$PW_i = t_i + \sum_{j=i+1}^{n} \left(M_{ij} \times t_j\right)$$

5. **Maximum Average Positional Weight (MaxAPW) rule** calculates the PW of the task divided by the number of its followers plus one:

$$MaxAPW_i = \frac{t_i + \sum_{j=i+1}^{n}\left(M_{ij} \times t_j\right)}{1 + |F_i|} = \frac{t_i + \sum_{j=i+1}^{n}\left(M_{ij} \times t_j\right)}{1 + \sum_{j=i+1}^{n} M_{ij}}$$

6. **Maximum Positional Weight of Followers (MaxPWF) rule** calculates the PW for the task but without the task itself:

$$PWF_i = \sum_{j=i+1}^{n} \left(M_{ij} \times t_j\right)$$

7. **Maximum Average Positional Weight of Followers (MaxAPWF) rule** is similar to the previous rule but divided by the number of followers of the task:

$$MaxAPW_i = \frac{\sum_{j=i+1}^{n}(M_{ij} \times t_j)}{|F_i|} = \frac{\sum_{j=i+1}^{n}(M_{ij} \times t_j)}{\sum_{j=i+1}^{n} M_{ij}}$$

8. **Minimum Slack of Task (MinSLK) rule** calculates slack of each task, ranks the values in ascending order, and assigns the tasks to stations starting with the minimum value. The minimum slack task is the one having the least flexibility in the assignment to workstations and should be assigned first. Let L_i and E_i be the latest and earliest stations that the task can be assigned to, respectively, then the slack of task i is given by:

$$Slack_i = L_i - E_i .$$

9. **Earliest Station for Task (MinEi) rule** ranks the tasks in ascending order of E_i and assigns them to stations starting with the minimum value. For a task i , E_i is the first station to which the task can be assigned given all its predecessors have been assigned before it. Let P_i be the set of predecessors of task i, then we have

$$E_i = \left\lceil \frac{t_j + \sum_{h \in P_i} t_h}{C} \right\rceil$$

In the matrix structure, the predecessors of a task are those tasks having entries of 1 in the task's column. Using i for rows and j for columns, then E_i can be represented as follow:

$$E_j = \left\lceil \frac{t_j + \sum_{i=1}^{j-1}(M_{ij} \times t_i)}{C} \right\rceil$$

10. **Latest Station for Task (MinLi) rule** ranks tasks in ascending order of L_i and assigns them to stations starting with the minimum value. For task i, L_i is the last station the task can be assigned to, such that all its followers can be assigned after it. It assumes knowing the needed number of workstations, yet an estimate of the upper bound on the number of workstations can be used according to. In Scholl & Voß (1997) an upper bound m_{max} is given as follows:

$$m_{max} = Min\left\{n, \left\lfloor \frac{\sum_{i=1}^{n} t_i}{C + 1 - t_{max}} \right\rfloor + 1, \left\lfloor \frac{2 \times \sum_{i=1}^{n} t_i}{C + 1} \right\rfloor + 1 \right\}$$

The value of L_i is calculated by (Capacho Betancourt, 2008; Fathi et al., 2018):

$$L_i = m_{max} + 1 - \left\lceil \frac{t_i + \sum_{h \in F_i} t_h}{C} \right\rceil$$

In the matrix format, with F_i as defined earlier, L_i is calculated as follows:

$$L_i = m_{max} + 1 - \left\lceil \frac{t_i + \sum_{j=i+1}^{n}(M_{ij} \times t_j)}{C} \right\rceil$$

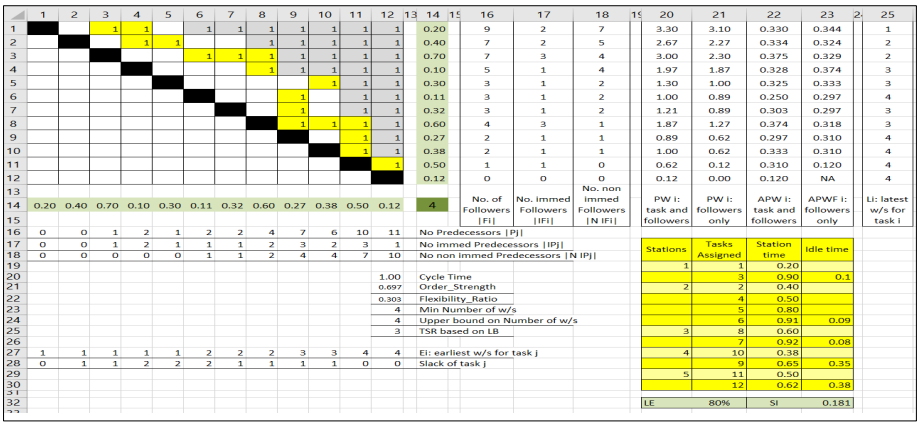


Fig. 4. Precedence matrix for 12-task ALBP (top left), values and indices of priority rules and problem parameters, and solution using MaxPW rule (bottom right)

In Figure 4, the precedence matrix of the 12-task example from Figure 1 is shown with relevant problem features and priority rules parameters. This was implemented in MS Excel. The 12-task example was solved using all ten rules, and all achieved 5-station solutions with line efficiency of 80%, but different smoothness index values. Best solution was given by MaxPW, which gave 0.181 smoothness index: about 25% lower than the others. This solution is shown in the lower right part of Figure 4.

4.2.3 Checking Task Assignment Feasibility with The Precedence Matrix

Solving the ALBP involves three main steps:

1. Developing a sequence of tasks, which does not violate any precedence constraints.
2. Assigning the tasks to workstations, in order of tasks in the sequence, without exceeding the cycle time on any workstation.
3. Improving the assignments by transferring and/or swapping tasks between workstations, without violating the cycle time limit or any of the precedence constraints.

The priority rules implement the first two steps. The third is usually the focus of the search-based heuristics (genetic algorithm, tabu search, etc.), which need to start with an initial solution, commonly developed with some priority rules. The sequence is a list of tasks, ordered in ascending or descending order based on some criteria. Sequences with no violations of the precedence constraints are feasible consequences, and the assignment can be as simple as dividing the sequence into sub-sequences, each with a total time less than or equal to the cycle time. The number of subsequences will be the number of workstations.

To illustrate, the values for the MaxPW and MinSLK rules for our 12-task example are given in Table 1, ordered from left to right in descending order for MaxPW and ascending order for MinSLK. Consider task 8, the precedence matrix shows tasks 1,2,3 and 4 as its predecessors, listed in column 8 (see Figure 4, top left). For MaxPW in Table 1, all four predecessors have higher positional weights and are ranked before task 8 in the sequence. Then the position of task 8 in the sequence is feasible. This can be shown true for all tasks in this sequence. Then the MaxPW sequence is feasible. For MinSLK, task 8 is positioned 7th, and since task 4 is 10th, not all predecessors of task 8 are ranked before it and hence the sequence is infeasible. Assigning tasks to workstations using this infeasible sequence may be complicated.

Table 1 - Task sequences based on MaxPW and MinSLK rules for the 12-task example

Position	1	2	3	4	5	6	7	8	9	10	11	12
MaxPW	3.30	3.00	2.67	1.97	1.87	1.30	1.21	1.00	1.00	0.89	0.62	0.12
Tasks	1	3	2	4	8	5	7	6	10	9	11	12
MinSLK	0	0	0	1	1	1	1	1	1	2	2	2
Tasks	1	11	12	2	3	7	8	9	10	4	5	6

Algorithm 2 can be used to check the feasibility of a sequence, based on the precedence matrix. As explained, the position of a task in a sequence is feasible if all its predecessors are ranked ahead of it. Algorithm 2 checks feasibility by considering each task in the sequence and confirming that all its predecessors have already been ranked before it in the sequence. In the matrix format, the predecessors of a task j are row tasks having entries of 1 in column j.

4.2.4 Feasible Assignment of Tasks to WorkStation

Algorithm 3 is an efficient procedure to assign assembly tasks to the workstations while ensuring the feasibility of the assignments, by using the precedence matrix's rows and columns to check the task predecessors and followers. It has been used to execute the priority rules described earlier. It can be incorporated into the execution of more advanced balancing heuristics, whenever an assignment action is to be made.

Algorithm 2: Checking the feasibility of task sequences using the precedence matrix

1. For each position $k = 1$ *to* n in the task sequence, do:
   a. Set j equal the task in position k
   b. For each row $i = 1$ *to* $j - 1$, do:
      i. If $M_{ij} = 1$ then:
         1. If $k = 1$ then
            a. Sequence is infeasible
            b. End
         2. *Let* $R = 0$
         3. For each position $l = 1$ *to* $k - 1$ in the sequence, do:
            a. Set r equal to the task in position l
            b. If $i = r$ then $R = R + 1$
         4. Next l
         5. If $R \neq 1$ then
            a. Sequence is infeasible
            b. End
   c. Next i
2. Next k
3. End

While assigning tasks to a workstation, if a candidate task cannot be assigned because of workstation time limit or a predecessor that could not be assigned yet, then the task is skipped, and the next in sequence is considered. Skipping tasks can cause more infeasibility issues. In Algorithm 3, the assignment starts with the top-ranked task, and when a task is assigned to a station it is removed from the sequence. If tasks are skipped, the assignment process continues until any later task could be assigned, then the process returns to the current top of the sequence, to reattempt the skipped tasks. This may increase computational efforts, but it is an accurate execution of the ranked sequence, since skipped tasks still retain higher priority than.

Algorithm 3: Assignment of tasks to workstations and checking for feasibility at each attempt

1. Initialize the list of already assigned tasks
2. Let the current open station be station k; set $k = 1$
3. Set station time of k; $T_k = 0$
4. Set $l = 1$
5. Let j = candidate task; task ranked number l in sequence, its process time is t_j
6. If $T_k < t_j$ then:
   a. If j is not last in sequence, then set $l = l + 1$ and Go To Step 5
   b. Set $k = k + 1$ and Go To Step 3
7. For each row $i = 1$ *to* $j - 1$, do:

a. If $\boldsymbol{M_{ij} = 1}$ then:
   i. For each task $\boldsymbol{r}$ in the list of already assigned tasks, do:
      1. If $\boldsymbol{i = r}$ then Go To Step 8
   ii. Next $\boldsymbol{r}$
   iii. Mark task $\boldsymbol{i}$ as not yet assigned to a station
   iv. Mark task $\boldsymbol{j}$ as cannot be assigned to current open station
   v. Let $\boldsymbol{l = l + 1}$ and Go To step 5

8. Next $\boldsymbol{i}$
9. Assign $\boldsymbol{j}$ to current open station $\boldsymbol{k}$
10. Let $\boldsymbol{T_k = T_k + t_i}$
11. Append $\boldsymbol{j}$ the list of the already assigned tasks
12. Delete $\boldsymbol{j}$ from the ranked sequence
13. If the length of the ranked sequence ≥ 1 then Go To Step 4
14. End

5. Computational Experiments

The proposed precedence matrix model has been implemented in MS Excel and used to compare the priority rules listed earlier, using benchmarking assembly line balancing problems. The matrix for each problem was constructed using Algorithm 1. The assignment of tasks to workstations was carried out using Algorithm 3. All quantities defined in previous sections in terms of the rows and columns of the matrix were implemented and used. The benchmarking data was obtained from the Assembly Line Balancing webpage (Boysen et al., 2021), which hosts sets of benchmarking data for line balancing and scheduling applications. The data is available for free, for research and non-commercial purposes. The datasets based on Otto et al. (2013) were picked. For the current presentation, we used only small size problems of 20 tasks. The dataset includes 525 problems, which were generated with desired order strength (OS) values of 0.20, 0.60, and 0.90. The actual OS values reported on the data page are as follows, which matched our calculations with the matrix model:

- Desired OS of 0.20: 225 problems with actual OS ranging from 0.14 to 0.30.
- Desired OS of 0.60: 225 problems with actual OS ranging from 0.50 to 0.65.
- Desired OS of 0.90: 75 problems with actual OS ranging from 0.80 to 0.85.

All problems were indicated to have been solved optimally, except for only eight of them, and the number of workstations in the optimum solutions was reported. The dataset also includes the lower bound on the number of workstations for each problem. We used this lower bound to calculate TSR for each problem, which will be used in analyzing the results. The values of TSR varied from 1.25 to 10.0. Since the number of tasks is fixed, then the variability is attributed to the number of needed workstations, which depends on the task process times. For the analysis of the results, the TSR values were classified into three levels:

- Low TSR: less than 3.00: 182 problems with OS from 0.14 to 0.84
- Medium TSR: from 3.00 to 5.00: 177 problems with OS from 0.15 to 0.85
- High TSR: more than 5.00: 166 problems with OS from 0.15 to 0.84

The common line balance performance metrics, line efficiency and smoothness index, are used to evaluate the performance of the balancing rules. The line efficiency (LE) is the ratio of the total work content of the assembly job to the time available on all needed workstations. Higher values LE are better. It is calculated as:

$$LE = \frac{\sum_{i=1}^{n} t_i}{m \times C}$$

The smoothness index (SI) measures how leveled the task distribution among the workstations. A smaller SI value indicates more nearly equal station times, which implies fair work loading and minimized idle times. It is defined as follows where T_{max} is the longest

workstation time among all workstations, T_k the time of workstation k, and m the number of stations in the solution.

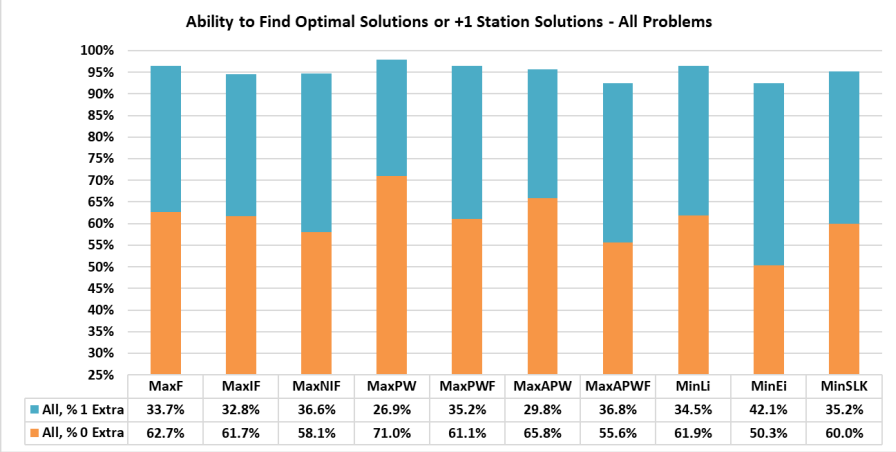
$$SI = \sqrt{\sum_{k=1}^{m} (T_{max} - T_k)^2}$$

6. Results and Discussions

All the work was implemented in MS Excel. The performance of the balancing rules was compared for the line efficiency and the smoothness index, considering the different levels of OS and TSR. In addition, since optimal solutions of the test problems have been reported, we calculate the percentage of problems that could be solved optimally using the priority rules. This measures the ability of the rules to find optimal solutions. This is meant to assess the attractiveness of the proposed matrix model to practitioners. Obtaining optimal solutions efficiently should be of great value to practitioners. The results are discussed in the following subsections.

6.1 Ability to Find Optimum Solutions

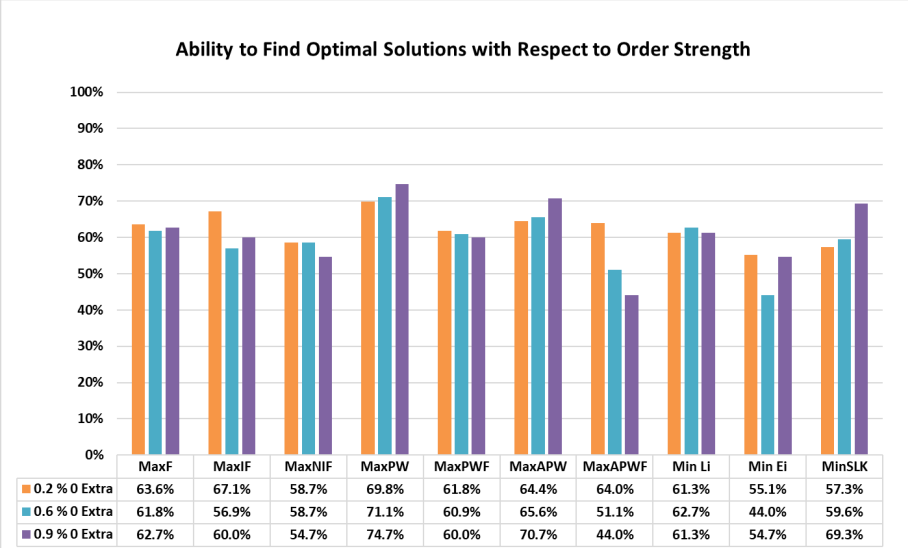
All tested balancing priority rules could find solutions with the optimum number of workstations, or the optimal number plus one extra station, in more than 90% of the problems and 95% for some of them. As in Figure 5, MaxPW performed relatively better than the others; finding solutions with optimum number of stations (0 extra) in 71% of the problems followed by MaxAPW at 65.8% of all problems. Achieving optimal solutions for an average of about 60% of the problems is a satisfying performance for these simple balancing methods. This is encouraging for analysts interested in collecting insights on potential alternative line designs efficiently with the proposed spreadsheet precedence matrix framework.



	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	MinLi	MinEi	MinSLK
All, % 1 Extra	33.7%	32.8%	36.6%	26.9%	35.2%	29.8%	36.8%	34.5%	42.1%	35.2%
All, % 0 Extra	62.7%	61.7%	58.1%	71.0%	61.1%	65.8%	55.6%	61.9%	50.3%	60.0%

Fig. 5. Ability to find solutions with optimum number of workstations or optimum +1

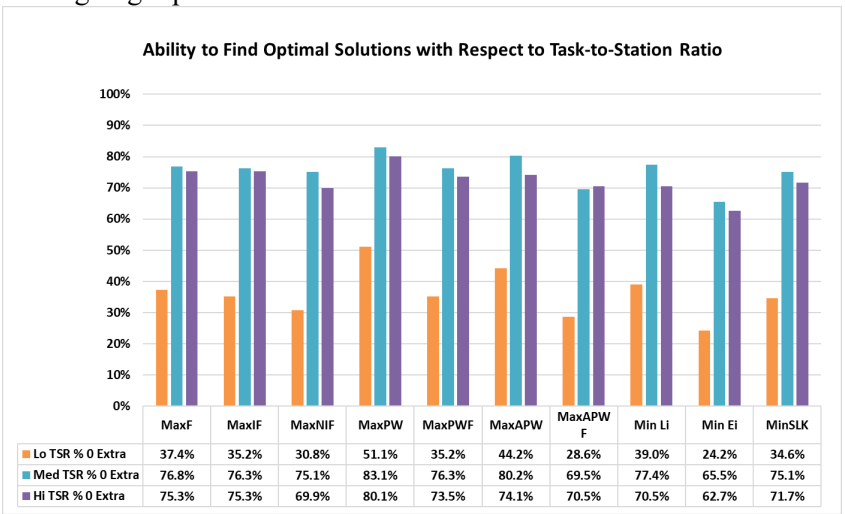
With respect to OS, Figure 6 shows the fractions of problems that could be solved with optimum number of stations. MaxPW can roughly be considered better than the others. There is no clear trend for the impact of OS on the performance. However, it can be noticed that more than half of the rules could achieve better results for the 0.9 OS than for the lower OS values. Higher OS implies fewer alternatives in assigning tasks to stations, and hence the rule-based methods can have better chances to perform well, since they only evaluate limited number of solutions.



	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	Min Li	Min Ei	MinSLK
0.2 % 0 Extra	63.6%	67.1%	58.7%	69.8%	61.8%	64.4%	64.0%	61.3%	55.1%	57.3%
0.6 % 0 Extra	61.8%	56.9%	58.7%	71.1%	60.9%	65.6%	51.1%	62.7%	44.0%	59.6%
0.9 % 0 Extra	62.7%	60.0%	54.7%	74.7%	60.0%	70.7%	44.0%	61.3%	54.7%	69.3%

Fig. 6. Ability to find solutions with optimum number of workstations vs. order strength

With respect to the impact of TSR, Figure 7 shows a significantly lower percentage (average of about 36%) of optimal solutions for the smaller TSR values, as compared to around 76% for medium TSR, and 72% for higher TSR values. Smaller TSR values imply a larger number of workstations, and fewer tasks at each station. Thus, the priority rules, which are already limited in their search capabilities, are having even less flexibility to combine the tasks on the workstations. Although the differences are not significant, the rules generally perform better for medium than for higher TSR. Higher TSR can be challenging for priority rules because of the increased flexibility, since the rules can evaluate a limited number of solutions. However, more investigations using larger problems are needed to confirm these observations.



	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	Min Li	Min Ei	MinSLK
Lo TSR % 0 Extra	37.4%	35.2%	30.8%	51.1%	35.2%	44.2%	28.6%	39.0%	24.2%	34.6%
Med TSR % 0 Extra	76.8%	76.3%	75.1%	83.1%	76.3%	80.2%	69.5%	77.4%	65.5%	75.1%
Hi TSR % 0 Extra	75.3%	75.3%	69.9%	80.1%	73.5%	74.1%	70.5%	70.5%	62.7%	71.7%

Fig. 7. Ability to find solutions with optimum number of workstations vs. task-to-station ratio

Although the current comparison is limited to small-sized problems, it can be argued, for this problem size, that TSR which is a function in number of tasks and number of workstations, may provide a better indicator of the expected solution quality than the popular OS, which uses the number of precedence relations among the tasks.

6.2 Line Efficiency

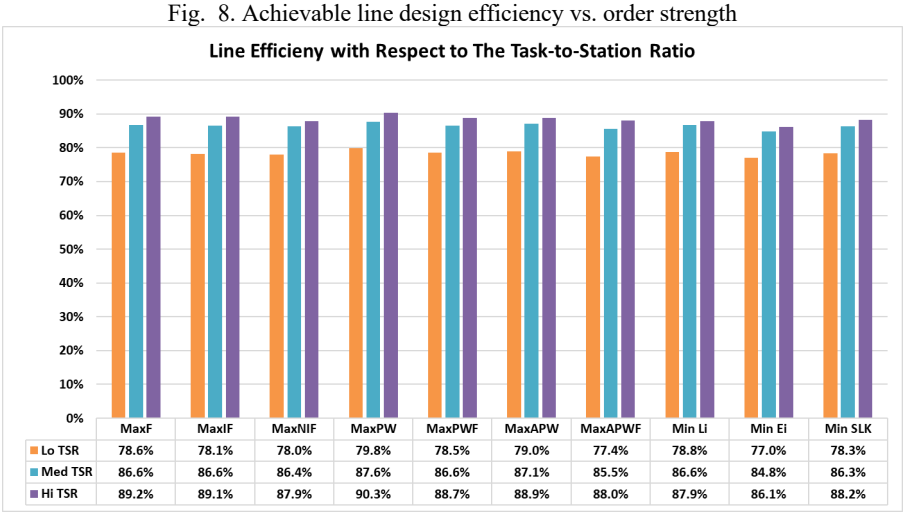
Figure 8 compares the achieved line efficiency (LE) with respect to OS. All the rules showed equivalent performance, achieving around 85% efficiency. It is also noticeable that the higher the OS the lower the efficiency. Lower efficiency means more workstations are needed and higher OS implies difficulty to assign tasks to workstation dure to stronger precedence constraints. This is consistent for all rules, although the differences are insignificant. LE decreases

within 5% range with OS increasing from 0.2 to 0.9. MaxPW, also showed marginally higher efficiency than other rules for all OS values.

For TSR (Figure 9), smaller TSR values allowed LE of less than 80%; between 77% (MinEi) to 79.8% (MaxPW). Higher TSR allowed higher LE results, roughly from 86.1% to 90.3% (MaxPW). There is a consistent trend that higher TSR leads to higher efficiency for all rules. Higher TSR indicates fewer workstations, which directly impacts the efficiency level. This, again, suggests that TSR can be more indicative of the expected solution quality and length of the line, than OS. It is worth noting that, TSR is calculated based on estimated lower bond of the number of workstations, while final solutions often require more stations than the lower bound. In fact, an optimum solution can be defined as the one using the minimum number of workstations, which is estimated by the lower bound value. Despite this, TSR appears to be a better predictor of solution quality than OS.

	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	Min Li	Min Ei	Min SLK
OS = 0.2	85.5%	86.0%	84.7%	86.3%	85.2%	85.2%	85.2%	85.0%	84.0%	84.6%
OS = 0.6	84.8%	83.8%	84.2%	86.0%	84.7%	85.0%	83.1%	84.5%	81.7%	84.0%
OS = 0.9	81.8%	81.7%	80.8%	83.6%	81.5%	83.2%	79.3%	81.9%	80.5%	83.1%
All	84.6%	84.4%	83.9%	85.8%	84.4%	84.9%	83.5%	84.3%	82.5%	84.1%

Fig. 8. Achievable line design efficiency vs. order strength



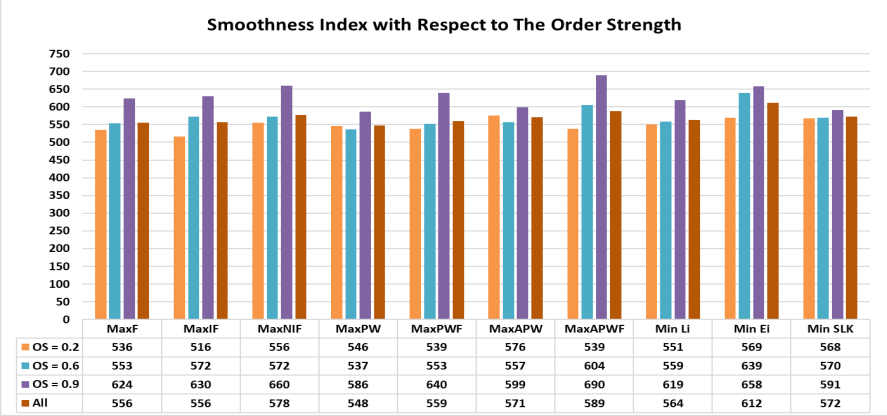
	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	Min Li	Min Ei	Min SLK
Lo TSR	78.6%	78.1%	78.0%	79.8%	78.5%	79.0%	77.4%	78.8%	77.0%	78.3%
Med TSR	86.6%	86.6%	86.4%	87.6%	86.6%	87.1%	85.5%	86.6%	84.8%	86.3%
Hi TSR	89.2%	89.1%	87.9%	90.3%	88.7%	88.9%	88.0%	87.9%	86.1%	88.2%

Fig. 9. Achievable line design efficiency vs. task-to-station-ratio

6.3 Balance Smoothness Index

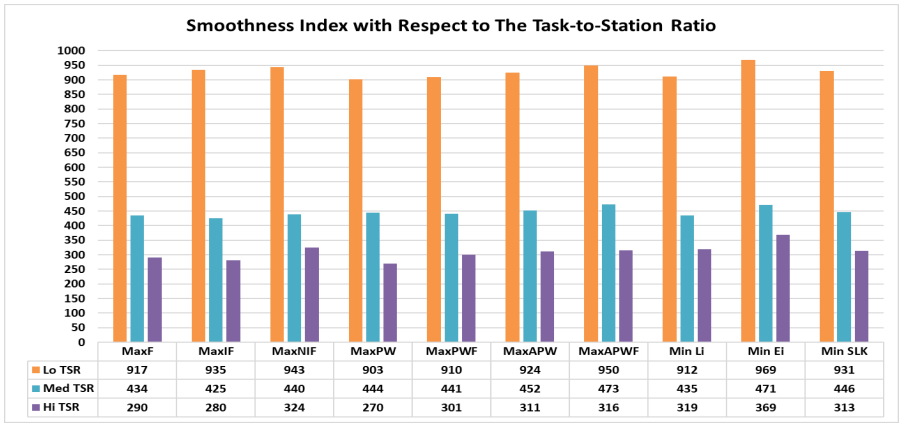
Figure 10 compares the achieved smoothness index (SI) with respect to OS. All rules are again performing similarly. Higher OS leads to relatively higher SI values. Higher OS implies more precedence relations and hence difficulty assigning tasks to stations. This can lead to using more workstations, making it harder to level the station times. Higher OS was also associated with lower LE. Lower SI is desirable as it indicates more balanced task assignments.

With respect to TSR, Figure 11 shows significant differences in performance for the different TSR levels. Lower TSR led to more than double the SI values compared to the higher TSR. Low TSR means larger numbers of stations and fewer tasks at each station. In such cases, the solutions are expected to have significant idle times. This supports that TSR can be a better predictor of the assembly line balancing solution quality than OS.



	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	Min Li	Min Ei	Min SLK
OS = 0.2	536	516	556	546	539	576	539	551	569	568
OS = 0.6	553	572	572	537	553	557	604	559	639	570
OS = 0.9	624	630	660	586	640	599	690	619	658	591
All	556	556	578	548	559	571	589	564	612	572

Fig. 10. Achievable smoothness index vs. order strength



	MaxF	MaxIF	MaxNIF	MaxPW	MaxPWF	MaxAPW	MaxAPWF	Min Li	Min Ei	Min SLK
Lo TSR	917	935	943	903	910	924	950	912	969	931
Med TSR	434	425	440	444	441	452	473	435	471	446
Hi TSR	290	280	324	270	301	311	316	319	369	313

Fig. 11. Achievable smoothness index vs. task-to-station ratio

In summary, this limited comparison of the balancing priority rules was meant to demonstrate the use of the precedence matrix-based framework. The ability of simple priority rules to achieve optimal solutions is encouraging to practitioners who prefer to have quick feasible solutions, which appear to be a common attitude in practice as discussed in Section 1. The proposed framework can provide them with efficient and inexpensive tools to perform the analysis. With simple priority rules, there are still chances to find optimal solutions efficiently.

The results of the comparison show that all tested priority rules performed similarly in general, with MaxPW slightly outperforming the others. The better performance of the positional weight rule is consistent with previous studies, which have usually found the positional weight to be an effective rule than other rules.

Other insights can also be drawn based on the results of the comparison. The TSR appears to be a more effective indicator of the performance of the balancing rules than OS. The number of workstations and the number of tasks (e.g., as in TSR) can be more effective than the popular order strength; OS, which is a function of the number of precedence relations and number of tasks, in assessing problem complexity. Both OS and TSR are functions of the number of tasks. TSR uses the number of workstations, which is related to the solution, while OS uses the number of precedence relations which are related to the problem inputs. It can be argued that the number of workstations, or estimates of the same, would be important in defining effective measures of performance and objective functions for simple ALBP. However, more comprehensive comparisons are needed to evaluate this argument.

7. Conclusion

This article proposes the use of the precedence matrix as a foundation for the development of efficient analysis frameworks for the simple assembly line balancing problem (ALBP). We have described the implementation of the precedence matrix-based framework in a spreadsheet model, leveraging the cost-effectiveness and robust data manipulation capabilities of the spreadsheet applications. The matrix model offers a straightforward organization of the line

balancing data for analysis purposes and provides a simple and effective tool for guiding the development of feasible assembly line design solutions. It supports two key components in developing balancing procedures: managing the inputs for solution procedures and ensuring feasible solutions. The proposed matrix framework is flexible and can be integrated with various balancing methods. In this work, we utilized simple priority rules for line balancing within the spreadsheet model. The outcomes demonstrated an efficient and cost-effective tool for production managers to evaluate potential line designs. The goal of this work was to help bridge the gap between research and practice in the area of assembly line balancing.

We introduced algorithms for constructing the precedence matrix from core balancing problem data and for using its contents to assign assembly tasks to workstations, resulting in feasible line designs. The proposed framework is easily extensible, allowing for the inclusion of additional performance measures and problem indicators, defined based on the rows and columns of the matrix.

A comparison of the performance of balancing priority rules was conducted using the matrix model. The findings indicated that the balancing rules included could achieve optimal solution in more than 60% of the test problems. No significant differences in performance were observed among them, yet the positional weight-based rules showed relatively better performance consistently. As discussed earlier, production managers often depend on their experiences and intuition to quickly develop feasible solutions. The proposed matrix model with the simple balancing rules has the potential to fit their needs, offering managers the flexibility to select the rules that best align with their specific operational goals. Other advanced balancing methods could be used as well.

In addition, the results of the comparison suggest that the task-to-station ratio (TSR), which is a function in number of workstations and number of tasks, can be a more reliable indicator of ALBP complexity and predictor of expected solution quality than the popular order strength (OS), which is a function in number of tasks and precedence relations among them. This implies that more emphasis should be placed on the number of workstations in developing performance metrics and objective functions for the line balancing problem. Nonetheless, more investigations are required to substantiate these arguments.

Future developments of the matrix framework may involve developing the matrix structure further to include more than two dimensions, allowing more information to be included in addition to the precedence relations. Other balancing approaches may also be included. This will enable the model to support other versions of the ALBP and provide a more nuanced and comprehensive approach to assembly line balancing.

References

Ahmadi, T., & van der Rhee, B. (2023). Multiobjective Line Balancing Game: Collaboration and Peer Evaluation. *INFORMS Transactions on Education*, *23*(3), 179–195. https://doi.org/https://doi.org/10.1287/ited.2022.0277

Bao, Z., Chen, L., & Qiu, K. (2023). An aircraft final assembly line balancing problem considering resource constraints and parallel task scheduling. *Computers & Industrial Engineering*, *182*, 109436. https://doi.org/https://doi.org/10.1016/j.cie.2023.109436

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, *142*(2), 259–277. https://doi.org/https://doi.org/10.1016/j.ijpe.2012.10.020

Boysen, N., Fleidner, M., Klein, R., & Scholl, A. (2021). Assembly Line Balancing, Datasets and Research Topics. *NA,[Online], Https://Assembly-Line-Balancing. de/, Accessed June*, *18*. https://assembly-line-balancing.de/

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

Capacho Betancourt, L. (2008). *ASALBP: the alternative subgraphs assembly line balancing problem. Formalization and resolution procedures*. Universitat Politècnica de Catalunya. https://doi.org/10.5821/dissertation-2117-93265

Çelik, M. T., & Arslankaya, S. (2023). Solution of the assembly line balancing problem using the rank positional weight method and Kilbridge and Wester heuristics method: An application in the cable industry. *Journal of Engineering Research*, *11*(3), 182–191. https://doi.org/https://doi.org/10.1016/j.jer.2023.100082

Chiang, W.-C. (1998). The application of a tabu search metaheuristic to the assembly line balancing problem. *Annals of Operations Research*, *77*(0), 209–227. https://doi.org/https://doi.org/10.1023/A:1018925411397

Chutima, P. (2020). Research trends and outlooks in assembly line balancing problems. *Engineering Journal*, *24*(5), 93–134. https://doi.org/https://doi.org/10.4186/ej.2020.24.5.93

Chutima, P. (2022). A comprehensive review of robotic assembly line balancing problem. *Journal of Intelligent Manufacturing*, *33*(1), 1–34. https://doi.org/https://doi.org/10.1007/s10845-020-01641-7

Driscoll, J., & Thilakawardana, D. (2001). The definition of assembly line balancing difficulty and evaluation of balance solution quality. *Robotics and Computer-Integrated Manufacturing*, *17*(1–2), 81–86. https://doi.org/https://doi.org/10.1016/S0736-5845(00)00040-5

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 & Industrial Engineering*, *139*, 106182. https://doi.org/https://doi.org/10.1016/j.cie.2019.106182

Erel, E., & Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, *9*(5), 414–434. https://doi.org/https://doi.org/10.1080/095372898233902

Fathi, M., Fontes, D. B. M. M., Urenda Moris, M., & Ghobakhloo, M. (2018). Assembly line balancing problem: A comparative evaluation of heuristics and a computational assessment of objectives. *Journal of Modelling in Management*, *13*(2), 455–474. https://doi.org/https://doi.org/10.1108/JM2-03-2017-0027

Gonzales-Rodriguez, D. C. (2022). System improvement through the application of assembly line balancing. *Proceedings of the International Conference on Industrial Engineering and Operations Management*, 4545–4558. https://ieomsociety.org/istanbul2022/proceedings/

Hamta, N., Ghomi, S. M. T. F., Jolai, F., & Shirazi, M. A. (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/https://doi.org/10.1016/j.ijpe.2012.03.013

Helgeson, W. B., & Birnie, D. P. (1961). Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering*, *12*(6), 394–398.

Hoffmann, T. R. (1963). Assembly line balancing with a precedence matrix. *Management Science*, *9*(4), 551–562. https://doi.org/https://doi.org/10.1287/mnsc.9.4.551

Hu, X., Xu, Z., Yang, L., & Zhou, R. (2015). A Novel Assembly Line Scheduling Algorithm Based on CE-PSO. *Mathematical Problems in Engineering*, *2015*(1), 685824. https://doi.org/https://doi.org/10.1155/2015/685824

Jackson, J. R. (1956). A computing procedure for a line balancing problem. *Management Science*, *2*(3), 261–271. https://doi.org/https://doi.org/10.1287/mnsc.2.3.261

Jiao, Y., Jin, H., Xing, X., Li, M., & Liu, X. (2021). Assembly line balance research methods, literature and development review. *Concurrent Engineering*, *29*(2), 183–194. https://doi.org/https://doi.org/10.1177/1063293X20987910

Katiraee, N., Calzavara, M., Finco, S., Battaïa, O., & Battini, D. (2023). Assembly line balancing and worker assignment considering workers' expertise and perceived physical effort. *International Journal of Production Research*, *61*(20), 6939–6959. https://doi.org/https://doi.org/10.1080/00207543.2022.2140219

Lapierre, S. D., Ruiz, A., & Soriano, P. (2006). Balancing assembly lines with tabu search. *European Journal of Operational Research*, *168*(3), 826–837. https://doi.org/https://doi.org/10.1016/j.ejor.2004.07.031

Manoria, A., Mishra, S. K., & Maheshwar, S. (2012). Expert System based on RPW Technique to Evaluating Multi Product Assembly Line Balancing Solution. *International Journal of Computer Applications*, *40*(4), 27–32. https://doi.org/10.5120/5034-7185

Mastor, A. A. (1970). An experimental investigation and comparative evaluation of production line balancing techniques. *Management Science*, *16*(11), 728–746. https://doi.org/https://doi.org/10.1287/mnsc.16.11.728

Moodie, C. L. (1964). *A heuristic method of assembly line balancing for assumptions of constant or variable work element times*. Purdue University ProQuest Dissertations & Theses, 1964. 6408691.

Moreira, M. C. O., Ritt, M., Costa, A. M., & Chaves, A. A. (2012). Simple heuristics for the assembly line worker assignment and balancing problem. *Journal of Heuristics*, *18*, 505–524. https://doi.org/https://doi.org/10.1007/s10732-012-9195-5

Nearchou, A. C. (2007). Balancing large assembly lines by a new heuristic based on differential evolution method. *The International Journal of Advanced Manufacturing Technology*, *34*(9), 1016–1029. https://doi.org/10.1007/s00170-006-0655-7

Nicholas, J. (2018). *Lean production for competitive advantage: a comprehensive guide to lean methodologies and management practices* (2nd ed.). Productivity Press. https://doi.org/https://doi.org/10.4324/9781351139083

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBP Gen for assembly line balancing. *European Journal of Operational Research*, *228*(1), 33–45. https://doi.org/https://doi.org/10.1016/j.ejor.2012.12.029

Ozdemir, R., Sarigol, I., AlMutairi, S., AlMeea, S., Murad, A., Naqi, A., & AlNasser, N. (2021). Fuzzy multi-objective model for assembly line balancing with ergonomic risks consideration. *International Journal of Production Economics*, *239*, 108188. https://doi.org/https://doi.org/10.1016/j.ijpe.2021.108188

Pearce, B. W., Antani, K., Mears, L., Funk, K., Mayorga, M. E., & Kurz, M. E. (2019). An effective integer program for a general assembly line balancing problem with parallel workers and additional assignment restrictions. *Journal of Manufacturing Systems*, *50*, 180–192. https://doi.org/https://doi.org/10.1016/j.jmsy.2018.12.011

Rahman, S. M. A., Rahman, M. F., Tseng, T.-L. B., & Kamal, T. (2023). A simulation-based approach for line balancing under demand uncertainty in production environment. *2023 Winter Simulation Conference (WSC)*, 2020–2030. https://doi.org/10.1109/WSC60868.2023.10408105

Rashid, M. F. F., Hutabarat, W., & Tiwari, A. (2012). A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology*, *59*, 335–349. https://doi.org/https://doi.org/10.1007/s00170-011-3499-8

Şahin, M., & Kellegöz, T. (2024). Novel mathematical modelling approaches and a new lower bounding scheme for multi-manned assembly line balancing problems with walking workers. *Computers & Industrial Engineering*, *190*, 110043. https://doi.org/https://doi.org/10.1016/j.cie.2024.110043

Salveson, M. E. (1955). The assembly line balancing problem. *Transactions of the ASME*, *7*, 939–947. https://doi.org/https://doi.org/10.1115/1.4014559

Schlueter, M. J., & Ostermeier, F. F. (2022). Dynamic line balancing in unpaced mixed-model assembly lines: A problem classification. *CIRP Journal of Manufacturing Science and Technology*, *37*, 134–142. https://doi.org/https://doi.org/10.1016/j.cirpj.2022.01.012

Scholl, A., & Voß, S. (1997). Simple assembly line balancing—Heuristic approaches. *Journal of Heuristics*, *2*, 217–244. https://doi.org/https://link.springer.com/article/10.1007/BF00127358

Sotskov, Y. N. (2023). Assembly and production line designing, balancing and scheduling with inaccurate data: A survey and perspectives. *Algorithms*, *16*(2), 100. https://doi.org/https://doi.org/10.3390/a16020100

Talbot, F. B., & Patterson, J. H. (1984). An integer programming algorithm with network cuts for solving the assembly line balancing problem. *Management Science*, *30*(1), 85–99. https://doi.org/https://doi.org/10.1287/mnsc.30.1.85

Talbot, F. B., Patterson, J. H., & Gehrlein, W. V. (1986). A comparative evaluation of heuristic line balancing techniques. *Management Science*, *32*(4), 430–454. https://doi.org/https://doi.org/10.1287/mnsc.32.4.430

Warshall, S. (1962). A theorem on boolean matrices. *Journal of the ACM (JACM)*, *9*(1), 11–12. https://doi.org/https://doi.org/10.1145/321105.321107