INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH
https://doi.org/10.1080/00207543.2022.2081630

Genetic algorithm and decision support for assembly line balancing in the automotive industry

J. B. H. C. Didden[a], E. Lefeber[b], I. J. B. F. Adan [a] and I. W. F. Panhuijzen[c]

[a]Department of Industrial Engineering and Innovation Sciences, OPAC, Eindhoven University of Technology, Eindhoven, Netherlands; [b]Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, Netherlands; [c]VDL Nedcar, Born, Netherlands

ABSTRACT

An important and highly complex process in the automotive industry is the balancing of the assembly lines. Optimally distributing jobs among the lines in order to obtain the highest efficiency is mostly done manually, taking a lot of time. This paper aims to automate the process of line balancing for a real-world test case. Automotive assembly lines are highly complex, and multiple factors have to be considered while balancing the lines. All factors relevant in a case study at VDL Nedcar are considered, namely, mixed-model production, sequence-dependent setup times, variable workplaces with multiple operators and multiple assignment constraints. A Genetic Algorithm (GA) is proposed to solve the formulated balancing problem and to act as a decision support system. Results on newly proposed benchmark instances show that the solution is dependent on the relation between the takt time and processing time of jobs, as well as the setup times. In addition, results of a real-life case study show that the proposed GA is effective in balancing a real-world assembly line and that it can both increase the efficiency of the line and decrease the variance in operating time between all model variants when compared to current practice.

ARTICLE HISTORY
Received 3 June 2021
Accepted 25 April 2022

KEYWORDS
Assembly line balancing; mixed models; genetic algorithm; sequence-dependent setup time; variable workplaces

1. Introduction

Assembly lines are typically used in mass-production facilities as they allow products to be finished faster with a high level of efficiency. Assembly lines were first introduced in the early 1900s in the Ford factories, allowing for the fast production of the Model-T as each car was exactly the same (Ford and Crowther 1922). Nowadays, customers desire high customisation while still maintaining a short lead time. The use of manual labor allows for a variety of models to be produced on the same assembly line, as operators are highly flexible. Implementing these machines lead to considerably high investment costs, thus making the planning and configuration of these assembly lines of high importance (Boysen, Fliedner, and Scholl 2007).

An assembly line consists of workstations connected in series. A workpiece is launched down the line at fixed intervals, referred to as the takt time (or cycle time). The takt time is related to the desired production quantity over a given time period (i.e. production quantity divided by production time). The workpieces are transported through each workstation with the use of a conveyor belt. At every workstation, an operator performs jobs on the workpiece, in which the operator typically does not exceed the takt time. The jobs are also constrained to precedence relations due to physical or technological restrictions. The overall aim is to plan the required jobs to assemble the complete workpiece according to a given objective while taking into account the takt time and precedence constraints. This problem is referred to as the Assembly Line Balancing Problem (ALBP).

Despite the vast amount of academic research done over the past decades, there is still a gap between the real world and the academic world. Assembly line balancing is still done manually at companies such as VDL Nedcar and can take anywhere from a few weeks when the assembly lines need to be rescheduled due to quantity changes or new model introduction, up to a few months if new lines are built or new models are introduced. This is due to the high amount of models and tasks that are present within the assembly line, both of which are in the order of thousands. Furthermore, given a large amount of time needed to balance the line, it is difficult to quickly see what potential changes have on the assembly line, e.g. changes in the takt time, positioning of the equipment, or the introduction of a new model. Therefore, the use of an

CONTACT J.B.H.C. Didden j.b.h.c.didden@tue.nl Department of Industrial Engineering and Innovation Sciences, OPAC, Eindhoven University of Technology, PO Box 513 Eindhoven, 5600 MB, Netherlands

Supplemental data for this article can be accessed here. https://doi.org/10.1080/00207543.2022.2081630

automated solution to balance the assembly lines is especially of interest to industry, in order to integrate it as a decision support system.

The reason why, to the best of our knowledge, many companies do not apply optimisation techniques such as meta-heuristics to solve the ALBP can be due to a number of reasons, as also identified by Boysen, Fliedner, and Scholl (2007); (1) lack of 'real world' problems that allow for a correct representation of the actual problem (i.e. problems in literature are often simplified), (2) the difficulties in solving complex problems, as constraints are often too difficult to be included within an optimisation model and (3) academic results could not be used for practical applications as specific solutions could not be translated back to more generic problems. In addition, VDL Nedcar states that the balancing of assembly lines started out in a situation with a minimal amount of complexity and gradually developed in more detail (i.e. more constraints were added while data regarding these decisions were not being stored). This leads to missing information in the data systems that is needed to make use of e.g. heuristics, to aid in the process of balancing. It also leads to information not being readily available in a single system. Then, it is difficult to see whether automation of the process can lead to feasible solutions that can be applied to the current assembly lines.

Even though a lot of research has been done on solving the General Assembly Line Balancing Problem (GALBP), there is still a lack of research considering real-world case studies, especially in the automotive industry. In contrast to current literature, real-world assembly lines contain a variety of additional constraints that need to be taken into account to formulate feasible (and optimal) line balances. These constraints are often related to the positioning of certain tasks on the line due to logistical issues or equipment placement, differences in working times between operators and constraints imposed on the takt time.

Therefore, the objective of this paper is threefold: (i) to develop an optimisation model that incorporates *all* constraints identified within an assembly line, (ii) *implement* this model within the company and (iii) use the model to identify possible room for improvements with the current line (i.e. show how the model can be used for decision support). Moreover, we aim to keep the model as generic as possible so that it can be used in a variety of settings. We develop a similar problem as proposed by Naderi, Azab, and Borooshan (2019). Our problem, however, extends this model by adding sequence-dependent setup times and introducing a proportional cycle time constraint, as described by Boysen, Fliedner, and Scholl (2007). This proportional cycle time constraint states that the takt time should be met dependent on the individual demand of each model. To the best of our knowledge, no other papers have taken all these constraints into account simultaneously. In addition, a horizontal line balancing constraint is added, aiming to reduce the variation in station time between operators and models. This is especially desired by automotive producers, in order to keep the workload for all operators across all models the same. Note that this paper is deduced from a thesis, as can be seen in Didden (2020). For more background information, we would like to direct the reader to this thesis. This paper, however, contains more results that evaluate the concept of using the proposed solution method as a Decision Support System.

The ALBP as set for VDL Nedcar contains constraints such as Mixed Model Line Balancing, Sequence Dependent Setup Times, Multi-Manned workstations and assignment restrictions. Other constraints such as ergonomics are not taken into account, as this is typically considered as a separate problem within the company. To this extent, an MIP is formulated (see Didden 2020) and a Genetic Algorithm is proposed to balance a current assembly line. Furthermore, we explain how the size of the problem can potentially be reduced to accommodate for the high amount of tasks and models that are present in a single assembly line. Initial analysis of the results of the newly generated benchmark instances shows that the efficiency of line balancing depends heavily on the task times and setup times. Furthermore, results of a case study show that the proposed algorithm can reach a significantly better solution in terms of the number of operators and horizontal balancing, when compared to the current line balance. A more thorough analysis also reveals potential room for improvement when using the proposed algorithm as a decision support system, such as the need to decrease the size of mounting positions and the ability of the new line balance to respond to changes in demand.

The remainder of the paper is organised as follows. First, in Section 2, a review of the current literature is given. Then, the problem statement is formulated considering the constraints identified at VDL Nedcar in Section 3. A Genetic Algorithm is then formalised in Section 4 to solve the problem. Section 6 describes a case study and the results obtained by implementing the Genetic Algorithm. Finally, conclusions and future research directions are stated in Section 7.

2. Literature review

The ALBP has been a thoroughly researched topic in the past few decades. One of the first known descriptions of the problem was given by Salveson (1955) who proposed an Integer Programming (IP) model. Wee and Magazine (1982) used bin packing algorithms to solve

the SALBP, as this problem is closely related to the bin packing problem. İlker (1986) furthermore introduced the Simple Assembly Line Balancing Problem (SALBP) and its extension which relaxes some of the constraints, referred to as the General Assembly Line Balancing Problem (GALBP).

More recent papers formulate GALBP's to solve problems more closely related to actual manufacturing lines. Kim, Kim, and Kim (2000) solve the two-sided assembly line balancing problem (TALBP), where workers can perform tasks on the left and right side of a workstation. Each task may have a preferred side or can be performed on both sides. The author formulates a Genetic Algorithm (GA) to solve the problem. Uğur and Toklu (2009) extend the problem by adding mixed models to the TALBP. They formulated a mathematical model and simulated annealing algorithm to solve the problem. Another extension to the problem is by allowing multiple operators to perform tasks on the workpiece simultaneously. Becker and Scholl (2009) address this problem as the ALBP with variable parallel workplaces. Multiple operators are allowed to perform tasks on a workpiece and a workpiece is split into multiple mounting positions. A task is uniquely assigned to a mounting position and operators can simultaneously perform tasks as long as they do not hinder one another (i.e. share the same mounting position). Roshani and Giglio (2017) solve the problem of line balancing while incorporating variable parallel workplaces and mixed models. They limit the number of workplaces to a predefined number of operators depending on the problem at hand. A mathematical model and simulated annealing algorithm are produced to solve the problem. Lopes et al. (2020) extend the definition of Becker and Scholl (2009) by introducing extra flexibility, allowing workers to start and end in different workstations. A key difference is that the cycle time constraint is imposed on the operator's workload instead of the workstation's workload.

Task times are also handled differently throughout the literature. Boysen, Fliedner, and Scholl (2007) propose three different approaches to include task time within the problem: stochastically following a distribution function, dynamically through learning effects of workers, or deterministically when task times are fixed and known in advance. The first method is often proposed through the use of fuzzy processing times. Zacharia and Nearchou (2012) in turn extend the SALBP with the addition of fuzzy task times with the aim to minimise the fuzzy cycle time and the fuzzy smoothness index/fuzzy balance delay. The authors propose a Genetic Algorithm to solve this problem. They continue their work in Zacharia and Nearchou (2013), looking at an extension of the problem, with the objective of increasing the efficiency (i.e. minimising cycle time and number of workstations) of the assembly line. Here, a meta-heuristic based on a GA is proposed. Lastly, Alavidoost et al. (2017) also consider fuzzy task times and also propose a GA. However, the authors consider a U-shaped lined instead of a straight line.

Tang et al. (2017) solved the problem of two-sided assembly lines with stochastic task times and various other assignment constraints such as (in-)compatible tasks and synchronous tasks. Variability in task times may occur due to unqualified operators, machine breakdowns, complexity of tasks, or similar reasons. The authors consider a fixed stochastic task time and formulate a probabilistic cycle time constraint (i.e. the cycle time constraint must be met with a certain probability). A hybrid teaching-learning-based optimisation algorithm is employed to solve the problem. Similarly, Aydoğan et al. (2019) define stochastic task times and apply the same probabilistic cycle time constraint, however, only for a U-line. The authors develop a Particle Swarm Optimisation method to solve the proposed problem. Hamta et al. (2013) consider the problem where task times are restricted between an upper and lower bound and a learning effect is introduced stating the task time decreases the more often a task is performed.

Another common restriction within assembly lines, especially in factories where large workpieces are assembled (e.g. the automotive branch), is the setup time between tasks. A common way to address this issue is by setting tasks for which the setup time is too large as incompatible, see Boysen, Fliedner, and Scholl (2007). Andrés, Miralles, and Pastor (2008) formulate a problem where setup times, due to walking distances or other restrictions, are added to the workload of each workstation. These setup time are considered to be sequence-dependent, i.e, the setup time depends on the order in which the tasks are executed. Scholl, Boysen, and Fliedner (2013) extend the problem by defining forward and backward setup times. Forward setup time occur within the same cycle (e.g. an operator has to walk towards the next task on the same workpiece). A backward setup occurs between the last task of the current cycle and the first task on the next cycle (e.g. an operator has to walk to the next workpiece to be assembled). Delice (2019) considers the problem of two-sided assembly line balancing with forward and backward setup times. The author proposes a Genetic Algorithm (GA) to solve the proposed problem.

Over the past years, some research has been done using real-world applications in the automotive industry by considering a multitude of realistic constraints. Alghazi and Kurz (2018) consider the ALBP while taking into account a Mixed Model Assembly Line, parallel

workstations, zoning constrains and ergonomic factors. To this extent, they developed an IP and Constraint Programming (CP) model. Using actual OEM data, they prove that their CP model outperforms the IP model. Naderi, Azab, and Borooshan (2019) considered a five-sided multi-manned mixed-model assembly line balancing problem based on actual OEM data from two different companies. Additionally, workers are allowed to move along different sides of the car and a maximum of three workers are to be allocated per workstation. To solve the problem, the authors develop a Mixed Integer Linear Program (MILP) and a Logic-based Bender's decomposition algorithm. They prove that their algorithm can reach sub-optimal solutions for the case study provided. However, for both studies, sequence-dependent setup time are not included as tasks that are too far apart are set to be incompatible. This may lead to solutions not being explored. Ferrari et al. (2019) also consider a multi-manned assembly line balancing problem, with the addition of incompatibility of different mounting positions, equipment sharing and worker cooperation. To this extent, they develop a Mixed Integer Program (MIP) and Simulated Annealing (SA) algorithm. They test their algorithm with a case study at an Italian manufacturer considering around 600 tasks. They demonstrate that their problem can find solutions close to the estimated lower bound. However, a single model is considered which simplifies the problem significantly. Sternatz (2014) considers a real-world problem taken from the Volkswagen Group in which they consider multiple constraints that apply to actual assembly lines in the automotive industries. Constraints such as high product variety (Mixed Model), sequence-dependent setup times, assignment restrictions and multiple operators per workstation (Multi-Manned) are taken into account among others. The authors develop an Enhanced Multi Hoffman Heuristic which is shown to be effective in finding solutions for complex real-world problems. Our work extends the problem by Sternatz (2014) by adding horizontal balancing to the objective function and assessing new assignment restrictions such as minimum and maximum distances between certain tasks.

Table 1 presents a comparison of the relevant literature to the current work[1]. The classification scheme of Boysen, Fliedner, and Scholl (2007) is used for the comparison. This table shows that we are one of the first to incorporate all these different types of constraints.

3. Problem formulation

This paper considers the assembly line balancing problem as it occurs within an automotive manufacturing plant. A number of models N is to be produced on a serial assembly line (i.e. a single assembly line without feeder or parallel lines). We are given a set of tasks J (these are all tasks required to produce the N models) that need to be assigned to a set of (unique) workstations W, where workstations can differ according to equipment and sizing constraints. Each workstation is divided into Z zones. Each task $j \in J$ has a deterministic task time equal to $\tau_{j,n}$, dependent on the model $n \in N$. The order in which the models are launched down the line are decided separately in a sequencing problem. However, as the order of the model changes constantly due to changes in production volume, the line is balanced according to predictions of the model demand instead of the exact sequence of the models. Not every task is applicable for every model. In that case $\tau_{j,n} = 0$. In addition, common tasks between models must be processed on the same workstation and by the same operator. On each workstation, a number of operators o is assigned, each performing a single task at any given moment, where $o \in \{1, \ldots, O_{\max}\}$, with $O_{\max}$ being the maximum number of operators able to be assigned to any given workstation. In addition, only one operator can work in any zone $z \in Z$ simultaneously. Naturally, precedence relations are applicable, stating the order in which tasks need to be executed (e.g. the radiator needs to be installed before the front bumper can be put into place). A schematic overview of an assembly line is given in Figure 1 and a more detailed overview of a single workstation is given in Figure 2.

Typically, the sum of the task times at each workstation, and for each operator, cannot exceed the takt time $\mathcal{T}$. The takt time is an indicator of the production rate of the line and is defined as the ratio between the total available production time, and the production volume. The takt time is typically constant over the span of a couple of months. However, as some models require significantly more tasks compared to other models, the strict restriction of the takt time can cause unnecessary idle times for certain models, due to the mixed model nature of the line. Therefore, it is allowed that models exceed the takt time, with a maximum time equal to β_w, as long as the average working time of all operators across all models (dependent on the demand of the models) is less than or equal to the takt time, as seen in Equation (1). Tasks that exceed the takt time should be performed by the same operator and can be finished in consecutive workstations. This does, however, lead to workload unbalance, as for some models the sum of the task time for an operator can be significantly higher than for other models. This again can cause unnecessary idle times. The objective therefore becomes to reduce the absolute difference in the sum of the task time between all models, also known as horizontal balancing, see Thomopoulos (1970). The problem of horizontal balancing can be solved through different

Table 1. Comparison of current relevant literature on assembly line balancing.

		Kim, Kim, and Kim (2000)	Andrés, Miralles, and Pastor (2008)	Becker and Scholl (2009)	Uğur and Toklu (2010)	Hamta et al. (2013)	Sternatz (2014)	Roshani and Giglio (2017)	Tang et al. (2017)	Alghazi and Kurz (2018)	Aydoğan et al. (2019)	Zhang et al. (2019)	Delice (2019)	Naderi, Azab, and Borooshan (2019)	Ferrari et al. (2019)	Lopes et al. (2020)	Current Work
I.	*Product Specific Precendence Graphs*																
	Single	✓	✓	✓	✓	✓			✓	✓	✓	✓	✓		✓	✓	
	Multi																
	Mixed						✓	✓	✓					✓			✓
II.	*Setup Times*																
	None	✓		✓			✓	✓	✓	✓	✓	✓			✓	✓	
	Sequence Dependent		✓		✓	✓	✓						✓	✓			✓
III.	*Assignment Restriction*																
	Linked			✓		✓		✓	✓					✓			✓
	Incompatible			✓		✓		✓	✓					✓	✓	✓	✓
	Fixed							✓	✓			✓			✓		✓
	Type								✓			✓					✓
	Minimum																✓
	Maximum																✓
IV.	*Movement of Workpieces*																
	Paced																
	Average	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓	
	Each							✓						✓			
	Prob																✓
	Same																
	Divergent																
V.	*Line Layout*																
	Serial	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
	U-Line									✓		✓					
	Parallel																
VI.	**Parallelisation**																
	P_{line}																
	P_{stat}																
	P_{tasks}							✓									
	P_{work}	✓ (2)		✓ (K)	✓ (2)		✓ (K)	✓ (K)	✓ (2)	✓ (K)		✓ (K)	✓ (2)	✓ (K)	✓ (K)		✓ (K)
	None		✓									✓				✓	
VII.	*Solution Technique*																
	Mathematical Model	✓	✓	✓		✓		✓	✓	✓	✓		✓	✓	✓	✓	✓
	Constraint Programming								✓								
	Genetic Algorithm	✓											✓				✓
	PSO					✓				✓	✓						
	SA							✓							✓		
	GRASP		✓														
	COMSOAL				✓												
	Multi-Hoffmann						✓										
	Migrating Birds											✓					
	Benders Decomposition													✓			
	Model Based Heuristic															✓	

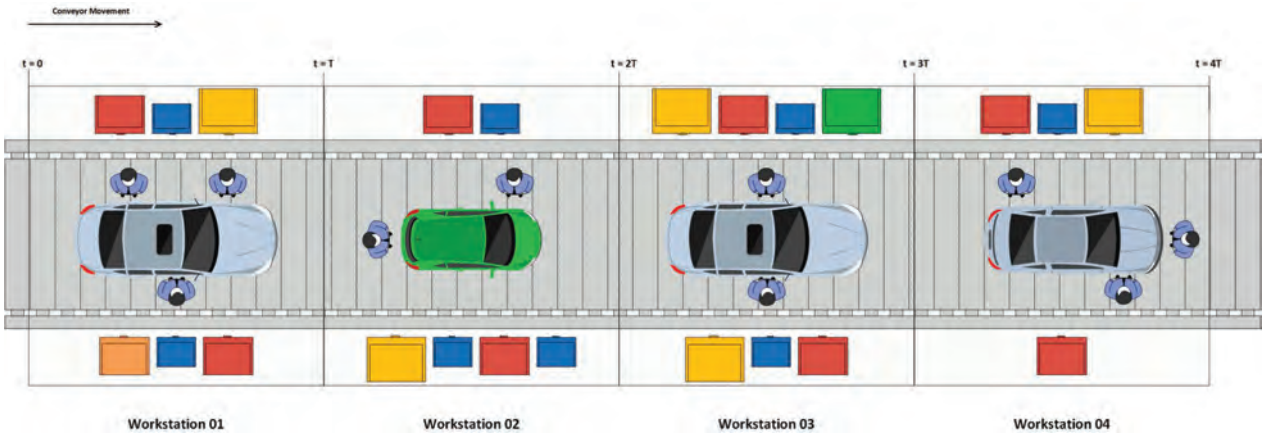

Figure 1. Schematic overview of a piece of the assembly line.

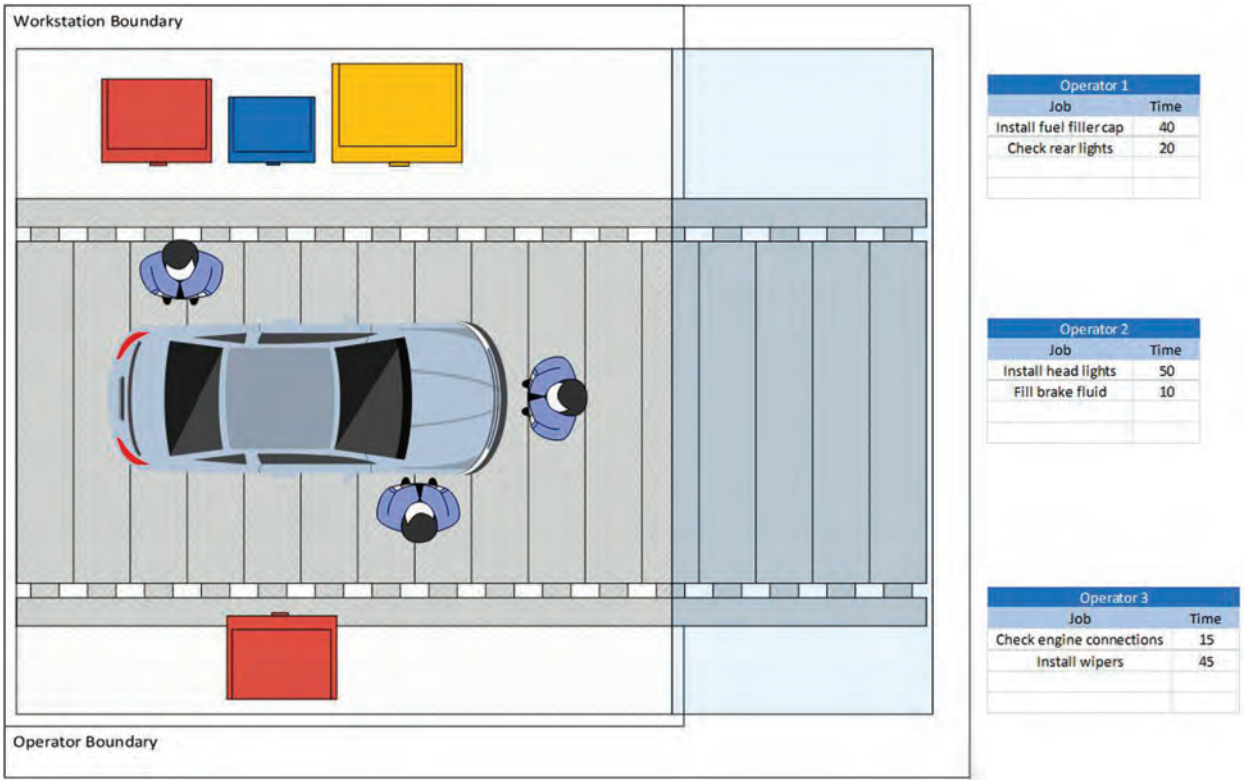

Figure 2. Schematic overview of a single workstation.

objective functions. For this, we refer to the work of Emde, Boysen, and Scholl (2010). In our case we aim to reduce the difference between the total task time for an operator and the takt time, as can be seen in Equation (2).

$$\sum_{n \in N} \alpha_n \cdot t_{w,o,n} \leq \mathcal{T} \tag{1}$$

$$\min \sum_{n \in N} \sum_{w \in W} \sum_{o \in O} |t_{w,o,n} - \mathcal{T}| \tag{2}$$

where $t_{w,o,n}$ is the station time of workstation w for operator o when model n is being produced. Due to the size of the cars produced on the line, a setup time needs to be added after a task is processed due to either walking distances or tool/equipment/material changes. These setup times also depend on the model that is produced. Not all tasks are performed for each model (e.g. tasks that are related to a convertible roof are only applicable to the models that have this type of roof), and not all models have the same size (e.g. an SUV is much larger than a compact car, which can cause an increase in walking distances between the front and back of the car). A similar method as proposed by Scholl, Boysen, and Fliedner (2013) is used. Between two consecutive tasks, $j_1, j_2 \in J$, in the same cycle, a forward setup time equal

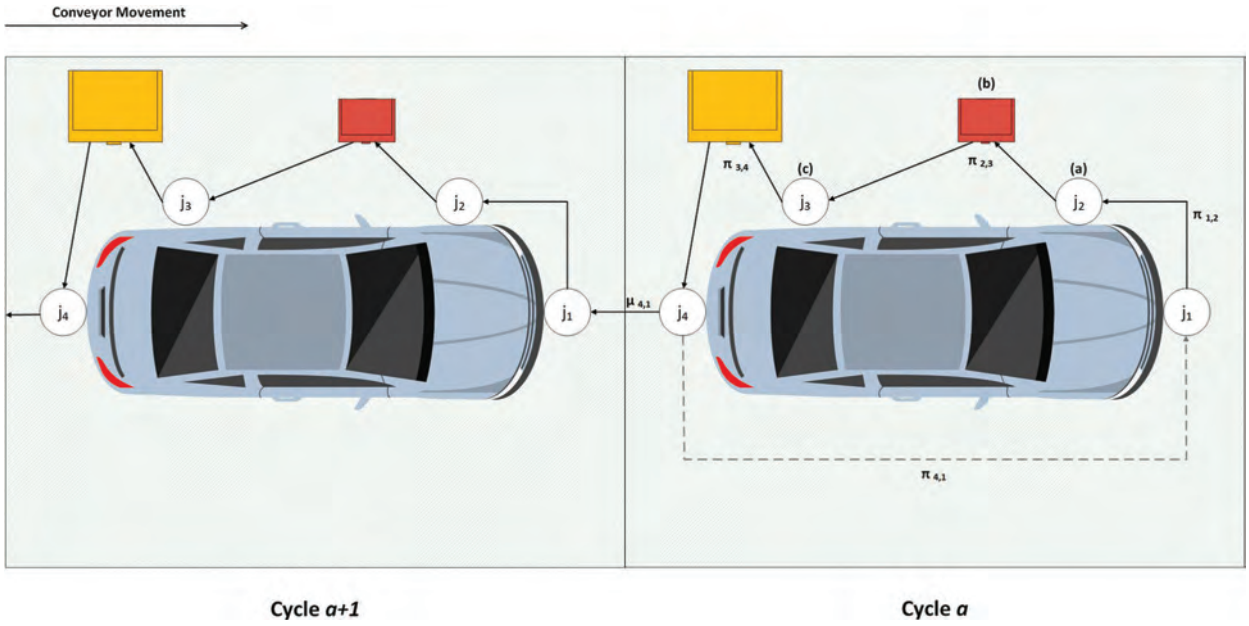

Figure 3. The connection between different tasks executed on a workpiece in consecutive cycles.

to $\pi_{j_1,j_2,n}$ is added. Furthermore, between the last task $j_2 \in J$ conducted on a model $n_2 \in N$ in one cycle, and the first task $j_1 \in J$ conducted on a model $n_1 \in N$ in the next adjacent cycle, a backwards setup time μ_{j_2,j_1,n_2,n_1} is added. See Figure 3 for a schematic overview of forward and backward setup times.

Example: Figure 3 shows how setups are defined and the differences between forward and backward setups. The simplest setup time is created solely by walking distances, as is the case between tasks j_1 and j_2. Here the operator does not have to change equipment or grab new materials. In contrast, the setup between tasks j_2 and j_3 is more elaborate. The operator must walk from position (a), to the material box at position (b), and then continue to task j_3 at position (c). This creates extra setup time in comparison with the situation that the operator walks directly from task j_2 at position (a) to task j_3 at (c). Next, the difference between a forward and backward setup becomes apparent. If the operator finishes cycle a with task j_4 and starts the next cycle $a+1$ with task j_1, then only a short setup time is needed (i.e. the backward setup μ_{j_4,j_1}). However, if task j_4 is succeeded by task j_1 in the same cycle a then a much longer setup time is relevant, namely the forward setup π_{j_4,j_1}.

Note that due to the addition of multiple operators per workstation, and the allowance of operators to process tasks over the station boundary, idle time may be incurred. These idle times are typically incurred when operators have to wait due to precedence relations between tasks or when operators in consecutive workstations are performing tasks in the same zones, as it is only allowed for one operator to work in a single zone simultaneously.

Lastly, we also consider assignment restrictions:

- Tasks that cannot be performed at the same workstation or by the same operator (Incompatible Tasks).

  Example: The installation and checking of the fuel tank must be done by a different operator.

- A pair of tasks that have a minimum or maximum distance between them (Minimum and Maximum tasks).

  Example: The filling of the break lines needs to be expanded over a minimum of 4 stations, thus the tasks related to the coupling and decoupling of the machines to perform these tasks must be at minimum 4 stations separated.

- Tasks that have to be performed at the same time by different operators on the same workstation (Linked Tasks).

  Example: A large workpiece that requires 2 operators to lift it.

- Tasks that have to be performed on a specific workstation and zone (Fixed Tasks).

  Example: Due to the dashboard being inserted from a feeder line at a specific side of the assembly line, any

tasks related to the installation of the dashboard need to be done at the workstation.

- Tasks that need to be assigned to a specific subset of workstations. (Type Tasks)

  Example: A subset can be workstations where the car is lifted. On this subset, tasks can be performed on the underside of the car.

The goal is to determine the sequence of the tasks allocated to each operator and workstation. Multiple objectives can be chosen to formulate the line balancing problem, with the most typical ones being the minimisation of the line length (i.e. the number of workstations) or takt time. Other objectives may be the minimisation of the number of operators or workload balancing. In the case of multiple objectives, the objectives can each be weighed according to a weighting factor η_i with $\eta_i > \eta_{i+1} > \cdots > \eta_n$ to prioritise one objective function over another. In this case, the objective is four fold: (i) minimise the amount of operators, (ii) minimise the amount of workstations, (iii) minimise the total station time and (iv) achieve the best possible workload balancing. The objective function is given in Equation (3),

$$\min \eta_1 \cdot |O| + \eta_2 \cdot |W| + \eta_3 \cdot \sum_{n\in N}\sum_{w\in W}\sum_{o\in O_w} t_{w,o,n} + \eta_4 \cdot \sum_{n\in N}\sum_{w\in W}\sum_{o\in O_w} |t_{w,o,n} - \mathcal{T}|. \tag{3}$$

4. Algorithm

In this section a Genetic Algorithm (GA) is proposed to solve the problem that is stated in Section 3. A GA is able to to find favourable solutions due to its ability to go through a large search space. The GA in this situation is kept simple, it is only responsible to generate new solutions and improve on any previous solutions. In basic terms, it determines the order of which the jobs are executed on the line. Another heuristic, the decoding, eventually determines the exact assignment of jobs, both to workstations and operators. The basis of the GA can be summarised in the following steps:

(1) Generate an initial population.
(2) Assign a fitness score to each individual according to Equation (3).
(3) Use tournament method to select individuals to perform crossovers.
(4) Select individuals to perform mutation on.
(5) Collect newly generated individuals and original individuals into a new population and assign fitness scores. Select $\mathcal{N}$ individuals to be carried over to the next generation.
(6) Terminate if a stopping criterion is met (time limit, number of generations (*#gen*), fitness score), otherwise repeat from step 3.

The remainder of this section is organised as follows. First, Sections 4.1 and 4.2 describe the encoding of a solution and how the initial population of the GA is initialised, respectively. Next, Section 4.3 describes the crossover and mutation operators used within the GA. Section 4.4 presents the selection procedure of the GA. In Section 4.5 a decoding algorithm is proposed in order to transform the created individuals into a feasible line balance. Finally, Section 4.6 describes how additional assignment constraints are implemented within the algorithm.

4.1. Encoding

The first step to the GA is encoding the chromosomes (i.e. a possible solution for the problem). Chromosomes are built up out of genes. For the ALBP each gene in the chromosome represents a task, e.g. task 7 would be given a gene labeled with 7. The total length of a chromosome equals the number of available tasks. The sequence of the genes in the chromosome represents the sequencing (i.e. the order) of the tasks on the assembly line. See Figure 4 for a visual example of a chromosome. It must be noted that the order in which the tasks appear in the chromosome, do not exactly relate to the order in which the tasks are processed. Due to the fact that multiple operators can be present at every workstation, successive tasks that are present in the chromosome may start simultaneously, but at different operators. The de-coding heuristic used to define at which workstation task j is present is given in Section 4.5.

4.2. Initial population

Due to the fact that the solutions for the ALBP are highly sequence-dependent, mainly caused by the sequence-dependent setup times, a diverse population can be useful for the GA. The initial population should, therefore, also contain a variety of unique task sequences, thus allowing for the crossover and mutation to be effective. Therefore, it has been chosen to use the method developed by Scholl, Boysen, and Fliedner (2013) to generate the initial chromosomes. This method is referred to as rule-GRASP (Greedy Randomised Adaptive Search Procedure).

The rule-GRASP algorithm assigns available tasks from a generated candidate list (CL) to the current constructed sequence (labeled as σ). The tasks (labeled j) in

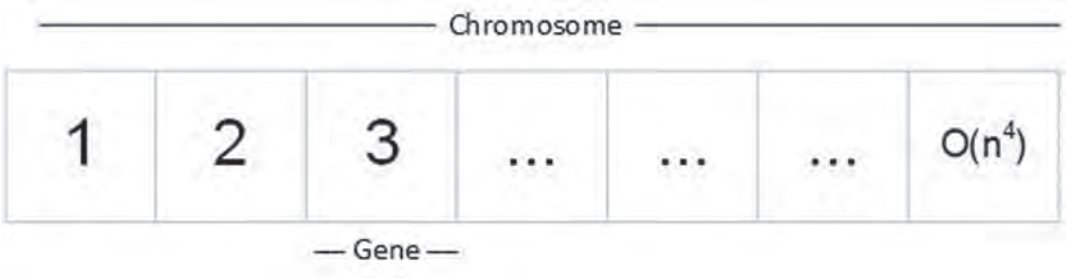

Figure 4. Example of a chromosome as used in the Genetic Algorithm.

the CL are ones that can be assigned to the next position in the sequence, due to their precedence relations (i.e. tasks that do not have any predecessors or whose predecessors have already been assigned). Then a priority rule (PR) is chosen at random from a uniform distribution and each task in the CL is assigned a value corresponding to the PR. The task with the lowest PR value is then assigned to the next position in the sequence. Lastly, the CL is updated by adding the direct successors of task j (i.e. from the set $\mathcal{S}_j^*$) to the CL if all other predecessors of these tasks have also been assigned. This process is repeated until all tasks have been assigned. See Algorithm 1 for a summary of the algorithm.

Algorithm 1: rule-GRASP algorithm

```
while CL ≠ ∅ do
    select priority rule from random uniform
      distribution;
    for j ∈ CL do
        calculate values for chosen PR;
        add j to σ;
        remove j from CL;
    end
    construct CL from S_j^*;
end
```

4.3. Crossover and mutation operator

Crossover children are created by selecting two parent chromosomes. The parents chromosomes can be chosen using a variety of methods such as: at random, roulette wheel selection or tournament selection. In this case, the tournament selection procedure is chosen. Here, a p_t part of the population of random chromosomes is chosen (with $n > 2$) from the current population and the two with the best fitness values are chosen to form a pair of parents. This process is repeated $\#gen/2$ times.

In order to retain feasible sequences of tasks (i.e. according to the precedence relations), a two point crossover method is chosen. To start, two random integers ($C1$ and $C2$) between $(2, n-1)$ are chosen (with $C1 < C2$). $C1$ and $C2$ denote positions in the chromosome. Then, the head and tail portions of the first parent chromosome ($P1$) are copied to the first offspring ($O1$). The head is the genes from positions 1 up to $C1$ and the tail from positions $C2$ up to n. Then the middle section (genes between $C1$ and $C2$) of P1 is ordered in the sequence it occurs in the second parent chromosome ($P2$). The elements are the copies to $O1$. The same procedure is used to create $O2$, where the roles of $P1$ and $P2$ are reversed. See Figure 5 for an example of the two point crossover method. In addition, not all parents produce an offspring with the crossover method. There is a probability, p_c with $0.5 \leq p_c \leq 1$, that two parent chromosomes produce an offspring.

The second method used to diversify the population is through mutation. Mutation children are created by first randomly selecting a gene in a parent chromosome. For this gene, the latest assigned predecessor ($m1$) and earliest assigned successor ($m2$) are recorded. A random integer from $(m1, m2)$ is then chosen and the gene is inserted into this point. There is a probability that a mutation will happen equal to p_m. See Figure 6 for a visual representation of the mutation operator.

4.4. Selection

Elitism (and selection) is a third important step in a GA. Elitism (or survival of the fittest), is used to carry over the solutions with the best fitness value for the next generation. This is done to preserve the quality of the solutions to the next generations. As has been explained earlier, the newly formed population and the original population are pooled together. From this total population a certain number, $\mathcal{N}$, of individuals with the best fitness score are selected to be carried over to the next generation.

4.5. Decoding

The decoding of the chromosome is based on the information needed for the fitness function and the desired output, in this case which tasks are allocated to which

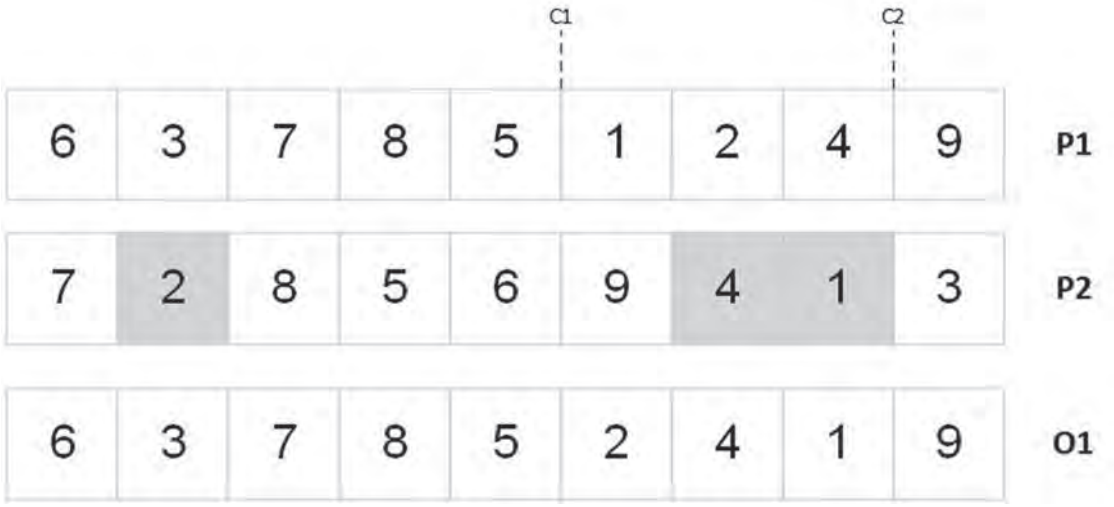

Figure 5. Example of the crossover operation.

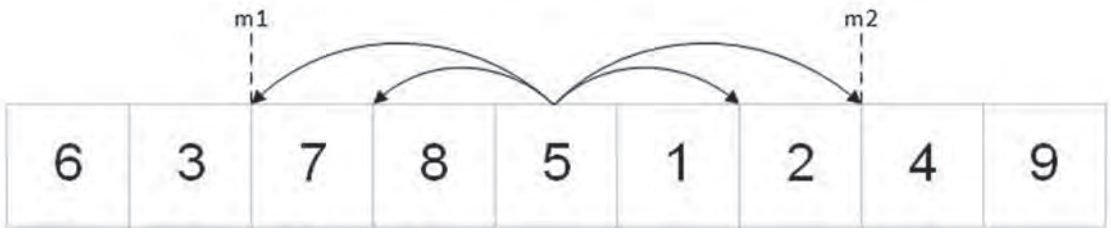

Figure 6. Example of the mutation operator.

workstation. The allocation of tasks should be deterministic, a sequence of tasks should always yield the same output, therefore, no randomness can be involved. When allocating tasks to a workstation, the assumptions as posed in Section 3 should be adhered to. For this purpose, a heuristic has been written to decode the chromosomes. The heuristics is based on three choices, the first being which operator a task is assigned to, secondly if a task can be allocated to the current workstation and lastly, how many operators are assigned to each workstation.

First, the assignment of tasks to a workstation is done through the calculation of its earliest starting time on every available operator on the current workstation. The start time of a task depends on four factors: (1) the end time of the last allocated task to an operator, (2) a forward setup time, (3) idle time created by precedence relations caused by tasks allocated to other operators on the same workstation (I_1) and (4) idle time created by precedence relations and mounting positions by tasks allocated to operators on previous workstations (I_2). The start time for each model and operator is calculated and the minimum start time over all models is the start time used to decide to which operator the task is allocated to. Finally, the output is a list of operators in ascending order from earliest to latest start time. It is noted that in case of a tie (i.e. the start times for two or more operators is equal), then the operator with the lowest index is chosen as the best operator, as opposed to choosing one of the operators at random.

Secondly, a check has to be done to decide if a task can be allocated to the current workstation. The station time $t_{w,o,n}$ for each model is calculated, which includes all possible idle and setup times (τ and μ). In order to calculate setup times, the first and last (i.e. f and l respectively) allocated task to a workstation for each model must be tracked. As the backward setup time can vary between any combination of models (e.g. the setup from model n_1 to n_2 might be different than from n_1 to n_3 depending if a task is executed for a specific model), this time is averaged according to the first task that is relevant for a model, according to (4).

$$\mu_{j_2,j_1} = \frac{1}{N^2} \sum_{n_1 \in N} \sum_{n_2 \in N} \mu_{j_2,j_1,n_1,n_2} \quad \forall\, (j_1, j_2) \in J. \qquad (4)$$

When all station times are calculated, they are averaged according to the model demand. If the average station time is lower that the takt time ($\mathcal{T}$) and the maximum station time is lower than the operators boundary ($\mathcal{T}_{\max}$), then the task is allocated to the selected operator. Otherwise, the check is repeated for the next operator in the list. If the current task cannot be allocated to any operator, a new workstation is opened.

The choice of deciding the number of operators on each workstation is done through the efficiency (η) of each operator. First, a maximum amount of operators per workstation is set, $O_{\max}$. This number typically depends on the available space on the assembly line. Next, a threshold value for the efficiency is set. The efficiency is the ratio between the useful time of an operator (i.e. time spent executing a task) and the takt time. This is in turn calculated for every model and is averaged according to the demand for each model. In other words:

$$\eta_{o,w} = \alpha_n \frac{\sum_{j\in(w,o)} \tau_j}{\mathcal{T}}. \tag{5}$$

After no more tasks can be allocated to the current workstation, the efficiency of each operator is calculated. If the efficiency of a single operator is lower than the threshold value, the number of operators for that workstation is reduced by one and the tasks are re-allocated to the remaining operators. This process is repeated until all operators reach the threshold value or until there is only a single operator left on the current workstation. The final output of the algorithm is the number of operators per workstation (O_w), number of workstations (W_{total}), a list of allocated tasks to each workstation and operator ($\mathcal{J}_{w,o}$) and the station times (t). The general heuristic can be seen in Algorithm 2.

Algorithm 2: Decoding Algorithm

Initialisation: set $W_{total} = 1, O_w = O_{\max}$, add σ_1 to $\mathcal{J}_{1,1}$, calculate $t_{1,1,n}$, add σ_1 to $f_{1,1,n}, l_{1,1,n}$ dependent on model;
$ii = 2$;
while $ii \leq noTasks$ **do**
  Set j to σ_{ii};
  Calculate I_1;
  Calculate I_2;
  Decide best operator to assign task to based on earliest start time;
  for $o \in O$ **do**
    for $n \in N$ **do**
      Determine $\tau_{l,j,n}$ and $\mu_{j,f}$;
      Calculate $t_{w,o,n}$ based off idle and setup times;
    end
    if *mean station time* $\leq \mathcal{T}$ *and max station time* $\leq \mathcal{T}_{\max}$ **then**
      add σ_{ii} to $\mathcal{J}_{w,o}$;
      Determine $t_{w,o,n}, f_{w,o,n}, l_{w,o,n}$;
      break;
    else
      continue;
    end
  end
  if *task fits in workstation* **then**
    $ii = ii + 1$;
  else
    Calculate $\eta_{w,o}$ for all operators on current workstation;
    if *mean efficiency* $\geq \eta_t$ *or* $O_w = 1$ **then**
      $W_{total} = W_{total} + 1, O_w = O_{max}$;
      add σ_{ii} to $\mathcal{J}_{w,1}$, set $f_{w,1,n}, l_{w,1,n}$ dependent on model;
      Calculate $t_{w,o,n}$;
      $ii = ii + 1$
    else
      empty $\mathcal{J}_{w,o}$ and reset $f_{w,o,n}, l_{w,o,n}$ for all operators on current workstation;
      $ii = prev$;
      $O_w = O_w - 1$;
    end
  end
end

4.6. Extra constraints

Most requirements for the assembly line as described in Section 3 are addressed in the decoding algorithm as described earlier. However, most constraints related to assignment restrictions are addressed through a penalty function in the fitness evaluation of the GA. Then in case one of these assignment restrictions is violated (e.g. a task is assigned to a non-compatible workstation) the solution is discarded in the next iteration of the GA.

Linked Tasks tasks are handled differently within the GA. If the current task being assigned is one of the two tasks of a linked task, then a new workstation is opened, and both task packages are assigned to different operators on the newly opened workstation. It is also noted that the minimum number of operators on this workstation is equal to 2 instead of 1. Furthermore, there is one special case of Minimum/Maximum tasks, where there needs to be a certain amount of empty workstations between two tasks, typically due to safety regulations. If the first task of one of these tasks pairs is assigned to a workstation,

then, after the workstation is closed, the number of current workstations (W_{total} in Algorithm 2) is increased by x instead of 1, with x being the number of empty workstations needed between the tasks.

5. Experimental results

This section provides an initial analysis on multiple test cases using the GA as a decision support system. First, the experimental settings, as well as newly generated benchmark instances, are given in Section 5.1. Next, a Design of Experiments is presented in Section 5.2 to analyse the influence of certain parameters on the quality of the solution. Lastly, Section 5.3 presents results of a Mixed Integer Linear Program and the proposed GA, as well as ananalysis on the generated benchmark instances.

5.1. Experimental settings

As stated before, no other research considers all the constraints mentioned in this paper, therefore, no standard benchmarks are available for testing. However, in order to show the effectiveness of this algorithm, new benchmarks instances are proposed[2]. The benchmarks are combinations of multiple different benchmark problems obtained from https://assembly-line-balancing.de/. The newly generated test instances only omit any assignment restrictions but contain all remaining constraints as previously described in Section 7.

The proposed GA is implemented in Matlab and tested on 259 different instances, each with a varying number of tasks, takt time, number of models and precedence relations. In order to speed up the GA, some early termination criteria are implemented. First, if after mutation and crossover less than 6 unique individuals remain in the population, or if after 20 generations the fitness value of the best individual has not changed, the simulation is terminated. In addition, the GA is also terminated after 900 seconds. No limit is set on the number of generations, as initial testing showed that in some cases, due to the horizontal line balancing, the fitness value can still decrease. For all problems, a theoretical lower bound of the number of operators as proposed in Didden (2020) is given for comparison.

Lastly, in order to analyse the results, multiple performance measure are given. The most important performance measures to consider are the efficiency of the line balance, the smoothness index and the lower bound. The overall line efficiency is the average of the individual efficiencies of all operators, in other words:

$$\eta = \frac{1}{O_{total}} \sum_{w \in W} \sum_{o \in O_w} \eta_{o,w},$$

where $\eta_{o,w}$ is as stated in Equation (5). The smoothness index results directly from the horizontal balancing fitness functions as given in Section 4.5, however, scaled according to the takt time:

$$SI = \frac{\sum_{n \in N} \sum_{w \in W} \sum_{o \in O_w} |t_{w,o,n} - \mathcal{T}|}{\mathcal{T}}.$$

5.2. Design of experiments

The algorithm performance depends on the chosen factors for the available parameters. In order to come to a correct choice for these parameters, a Design of Experiments (DoE) is done. During DoE, multiple values of each parameter are checked in order to evaluate their influence on the results. In this case, two values for each parameter are chosen, a lower and upper bound value. The values of each parameter are chosen through an educated guess and can be seen in Table 2.

A two-factor factorial design is performed, with an addition of 17 centre points to check for possible curvature. This results in a total of $2^6 + 17 = 81$ runs. *StatGraphics18* is used to create the experimental design and to analyse the results. For each run, 10 samples are taken. A test case is created consisting of 90 tasks and 1000 models, taken from a real-world example at VDL Nedcar. This ensures similar results to an actual real-world case, however decreases the computation time.

Figure 7 shows the main effects plot for each of the six parameters. A line is drawn from the lower bound to the upper bound (left to right), corresponding to the effect of the bound. The goal is to minimise the cost function (fitness), which corresponds to the lowest point on the line. The longer the line, the more effect a certain parameter has. In addition, the steeper the slope of the line, the greater the magnitude of the main effect. A negative slope relates to a better result with a higher value of the parameter. It becomes clear that η has the largest influence on the results.

The main effects significance must also be checked, this is shown by the Pareto chart in Figure 8. Here the standardised effects are shown of the main parameters as well as the two-factor interaction between parameters. The blue vertical line represents the significance level. It can be seen that the threshold efficiency (η)

Table 2. Table showing the lower and upper bound values for the parameters of the GA for the DoE.

Parameter	Lower Bound	Upper Bound	Optimal Value
p_m	0.1	0.9	0.9
p_c	0.1	0.9	0.9
#*gen*	10	100	100
$\mathcal{N}$	10	100	100
η_t	0.1	0.75	0.8
p_t	0.05	0.2	0.2

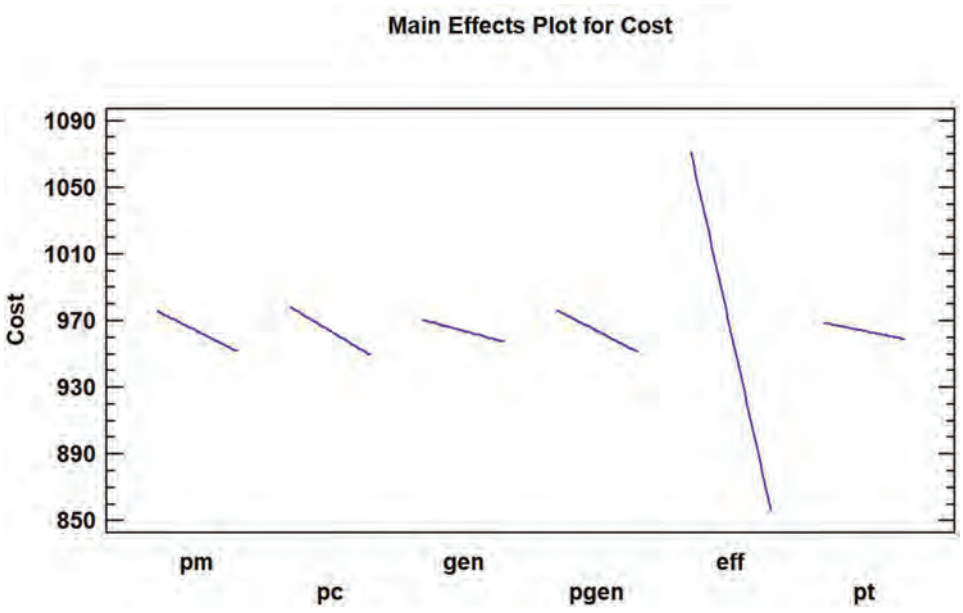

Figure 7. The main effects plot for the parameters of the GA.

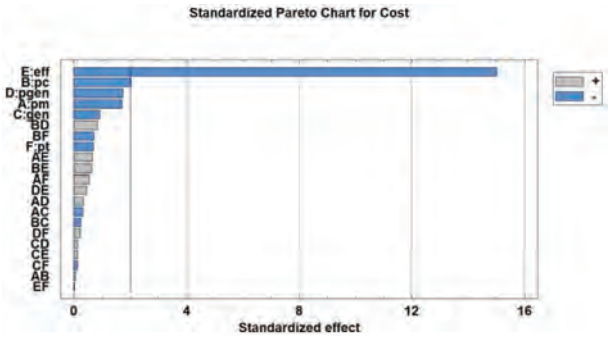

Figure 8. The standardised effect charts for the parameters of the GA.

and probability of crossover (p_c) have significant effects. Lastly, the optimal value for each parameter can be decided in *StatGraphics18*, this result is shown in Table 2.

5.3. Experimental results

In the online appendix, a Mixed Integer Linear Program (MILP) is given for the model described in this section. Some of the smaller benchmark instances were chosen to test the MILP. To this extent, the MILP was implemented in Python, using IBM ILOG CPLEX as a solver, with a given time limit of 7200 seconds. Results for four of the test cases can be seen in Table 3. Only a single test case could be solved optimally in the given time limit, while two others that contained more workstations and operators to append tasks to, could not be solved within the given time limit. The fourth case that contained 31 tasks received an *out-of-memory* error. It can therefore be concluded an MILP cannot be used for real-life test cases.

The results of the GA for the all benchmark instances can be found in the supplementary materials. It can be seen that for most instances, the GA can achieve results relatively close to the theoretical lower bound. Even though that the lower bound does not provide the exact optimum (in some instances it is closer than for others), some conclusions can still be drawn. In case that the average task time is relatively small compared to the takt time (e.g. see parameters of testcase $kilbrid_c = 110$ in the results file[3]), the solution quickly converges to the lower bound. The algorithm can find multiple solutions that reach the lower bound in terms of the number of operators. The only difference between the found solutions is the smoothness index. However, given the tight precedence constraints, the algorithm still quickly converges to a minimum.

Table 3. Comparison of the results of the MILP model and the proposed GA for small test instances.

	Instance Settings			MILP Results					Algorithm Results			
Instance	No. Of Tasks	No. Of Models	Takt Time	No. Of Operators	No. Of Workstations	Objective	CPU Time	Gap	No. Of Operators	No. Of Workstations	Objective	CPU Time
Mertens	11	2	18	2	1	211.775	1181	0	2	1	211.775	16.25
Mertens	11	2	18	4	3	433.325	7200	35	4	3	433.19	16.2
Jackson	13	2	18	3	3	335.147	7200	33	3	2	322.525	1.2
Roszieg	31	3	25	–	–	–	< 7200	–	6	5	655.088	14.2

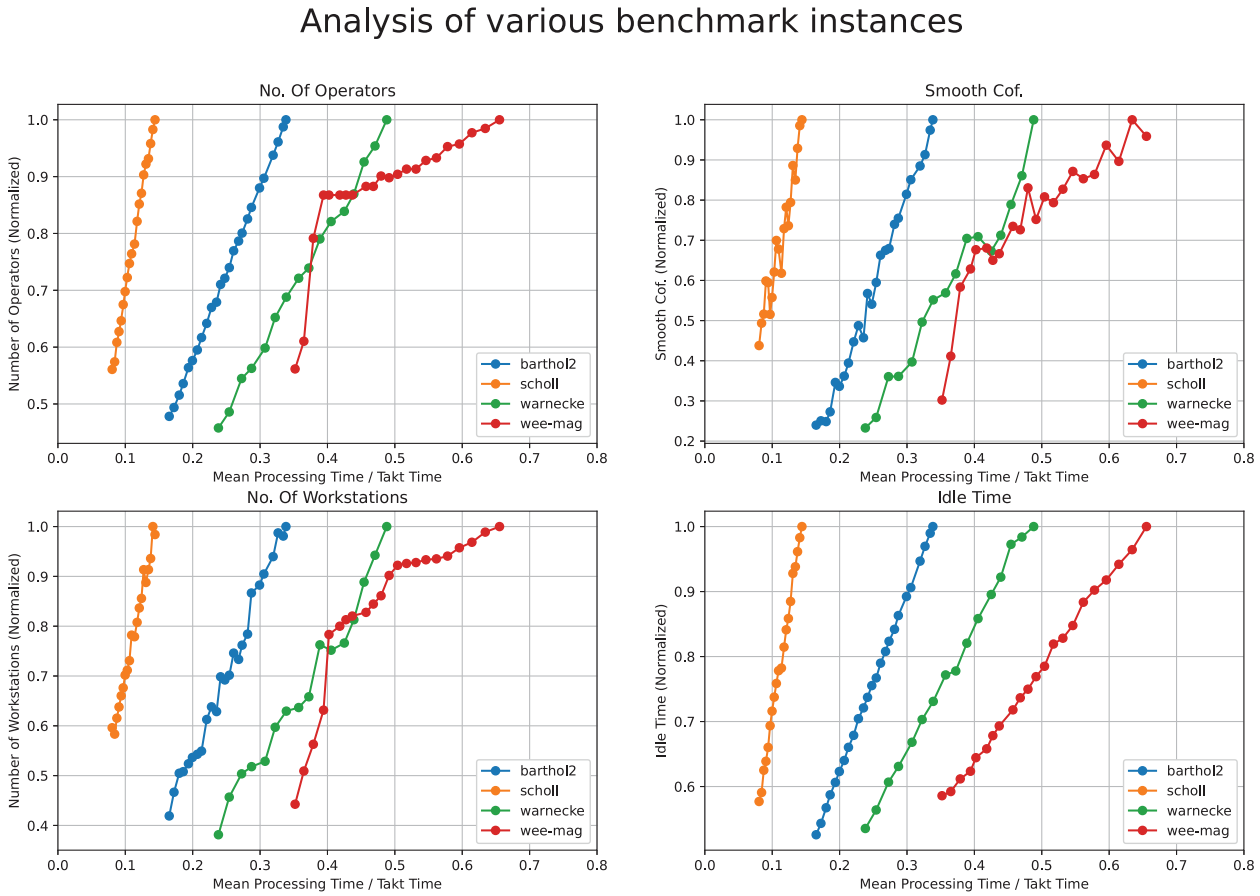

Figure 9. Analysis of various benchmark instances.

Regarding the results of the benchmark instances, some interesting observations can be made. First, the relative task time, which denotes the ratio between the mean task time of all models and the takt time, has a pronounced effect on the quality of the solution. As can be seen in Figure 9, typically if the takt time is increased, and all other parameters are kept the same, then a linear relation between the objective functions and the relative task time is present. However, in case of the instances of *wee-mag*, a different trend is seen. The relative task time here is high in most case (> 0.5), meaning that often only a single task can fit on a workstation. Increasing the takt time therefore does not give the same linear relation. A substantial drop in the objective values can be seen between $\mathcal{T} = 52$ and $\mathcal{T} = 54$. At this point, the relative number of tasks that fit within one operators takt time increases from 2 to 3, allowing more variability to be found within the solution space. Therefore, it is better to keep the relative ratio between task time and the takt time as low as possible.

A second observation that can be made is that the CPU time is not necessarily dependent on the number of jobs or the number of models that are on the assembly line. Comparing the benchmark instances *Arc83* and *Tonge70*, a lot of similarities in number of jobs, models and order strength of the precedence graph (i.e. the order strength (OS)). The relative task time is different, however, for a lower relative task time it would be expected that the CPU time increases, as the solution space becomes bigger (i.e. more tasks would fit within an operators takt time). However, a potential cause of the increase in CPU time is the range of forward and backward setup times. The range in both forward and backward setup times for the instances of *Arc83* are much smaller, compared to those of *Tonge70*. This can cause the solution space to decrease; even though the OS of the precedence graphs is relatively the same, if the range of setup times is large, then picking a different line balancing may result in a steep increase of total setup time in the line. As a high amount of total setup time is undesired, the solution space becomes restricted.

6. Case study

In this section, the results of a case study is described which took place at VDL Nedcar. The entire assembly line at VDL Nedcar is split up into multiple smaller sections, each responsible for different assembly procedures.

For this case study, a part of the assembly line was taken to analyse and apply the developed GA to. This part of assembly line is referred to as ALX throughout the remainder of this section. Section 6.1 describes the method used to collect and sort data as an input for the DSS is described. Section 6.2 presents analysis of the GA on one of the assembly lines at VDL Nedcar.

6.1. Data collection

In order to implement automated line balancing at Nedcar, the correct data needs to be gathered, analysed and adapted to fit in the previously described model. The main data that is needed is as follows:

- List of tasks with corresponding task time
- List of available mounting positions
- Precedence relations between tasks
- List of variants and their corresponding demands
- Forward and backward setup times
- Takt Time

In addition, some extra constraints may need to be added to the model depending on the assembly line section that is being balanced. Each assembly line is unique, e.g. some lines have tasks that have to be done on a specific workstation due to equipment or material constraints and other tasks might require two operators to work simultaneously on the same task. An important step to take is to analyse the assembly line thoroughly and take note of special constraints. It must however be noted that not each additional constraint should be adhered to strictly, as this may reduce the degrees of freedom in the eventual line balance. E.g. a task might require a specific piece of equipment that is only available at a single workstation, therefore a constraint could be added that fixes this specific task to that workstation. However, the cost reduction of the new line balance might outweigh the costs of moving the equipment. This only becomes apparent if the initial balance is done with the least amount of additional constraints.

This case study considers a single assembly line at VDL Nedcar, referred to as ALX. An initial analysis of the data shows that ALX contains around 1000 tasks that have to be processed. Considering the amount of options that are installed on the assembly line, the total amount of variants is bounded by $\mathcal{O}(10^4)$ (an exact number is not known as not all option combinations can exist). It becomes apparent that the amount of variants quickly exceeds the amount of tasks, as the number of variants increases exponentially with an increasing amount of options. Therefore, both the amount of tasks and variants have to be reduced within reason in order to balance the line within a reasonable time frame. Given the observations made in Section 5, the number of tasks should be reduced sufficiently, while still keeping the ratio between the mean processing time of tasks and the takt time reasonable (e.g. according to the results presented in Section 5 this should be lower than 3).

Precedence relations are not directly available. The reason for this is that assembly line balancing is done by hand. The responsible engineer typically knows the order in which the tasks have to be done, however, the exact relations are never written down. A similar principle applies to setup times. These are added later in the balance if it becomes apparent that an operator has to cover extra distance between tasks. Furthermore, in order to reduce the number of tasks, certain task packages are created, which consists of tasks with direct precedence relations between them (i.e. tasks that have to be done in a specific order). For the models a similar procedure is followed. Model variants are created based on the tasks that have to be performed on them, instead of the options they contain (i.e. variants are created that are relevant for a certain section of the line). This reduces the number of models in the line by 90%. The methods of how both data types were collected is explained in more detail in Didden (2020).

Lastly, the forward and backward setup times have to be calculated. This is done by defining setups between mounting positions, instead of defining them for each task pair individually, as this requires a high amount of information needed that is not readily available. Each task package is assigned a unique mounting position, according to the ones given in Figure 10. Some individual tasks already contain setup times (i.e. time needed to walk to grab equipment/material or to walk to the vehicle). As this is the case, and because it is difficult to find the exact time that has been added, it was decided to add a low amount of forward and backward setup times in order not to add more setup times than occur. Similarly, this is in line with the analysis in Section 5. Keeping forward setup times in a smaller range, while still being feasible, increases the solution space. Standard times between

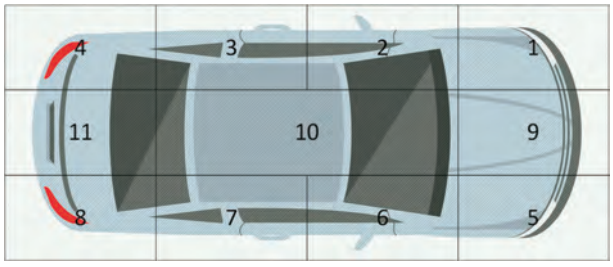

Figure 10. The mounting positions available to each task package.

mounting position are added for both the forward and backward setup times. This also includes an additional mounting position 12 that is not included in Figure 10. This mounting position is an off-line pre-assembly, typically done on the side of the assembly line.

6.2. Test case

The only parameter that was altered from the DoE results is the number of generations. It became apparent during testing that the fitness function still decreases after 100 generations, due to the horizontal balancing parameter. It was therefore chosen to swap out the limit on the number of generations by a time limit. This limit was set to *7200* seconds, as finding a better solution is of more importance than finding a solution quickly. The other settings of the GA are the upper bounds found in Table 2, including the maximum takt time $\mathcal{T}_{\max} = 1.3\mathcal{T}$. Lastly, the calculation of the lower bound, as can be found in Didden (2020), results in a maximum possible line efficiency of 95%.

Next, the algorithm is tested on the real-life test case at VDL Nedcar. The current line balance can be seen in Figure 11, while the new line balance generated by the GA can be seen in Figure 12. Overall, a substantial increase was seen in the line efficiency[4], with both less operators and workstations used. The smoothness index has been decreased from 10733 to 3498, thus resulting in a more even station load throughout the assembly line, which can be seen in Figure 12. The maximum station load has been decreased from 205% to 128%, reaching the desired value of 130%.

Given the new line balance, some managerial insights can be formulated. A first observation that can be made

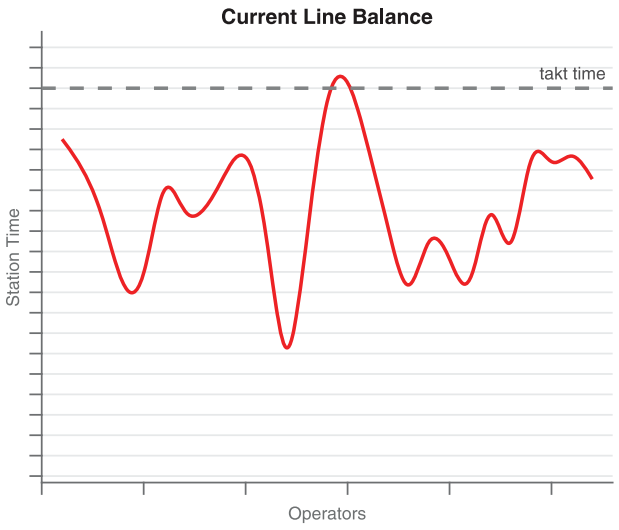

Figure 11. Distribution of the station times over all operators for the current line balance.

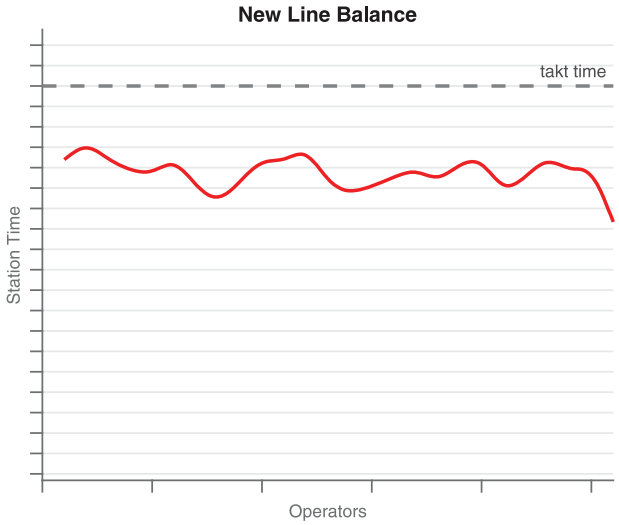

Figure 12. Distribution of the station times over all operators for the new line balance.

with the new line balance, is that for all operators, the mean station load has dropped significantly. Figure 12 shows that, with the new line balance, there is room to decrease the takt time, without needing to re-balance the line. This, however, comes at the cost of needing to increase the maximum allowable station load, in order to satisfy all the constraints. The takt time can be dropped by approximately 9%, given that the maximum allowable takt time is increased to $1.5\mathcal{T}$. This can directly help to counteract any small deviations in demand that may be present, without the need to re-balance of make adjustments within the line.

Furthermore, it is noticed within the data that the mounting positions as provided by the company are too big. Given the constraint that only a single operator can be presented in a mounting position at any given time, drastically increases the number of operators and number of workstations necessary. This causes an infeasible solution to be obtained. In order to come to a feasible solution given the input data, it was decided to relax the constraint and allow two operators to be present within a mounting position.

Lastly, Figure 12 still often shows some steep decreases in station time for some operators. This has multiple causes. First, some models have significantly more jobs compared to other models. Even though the demand for these models is low, it still causes inbalances within the system that are difficult to address. Secondly, assignment restrictions also can cause certain operators to not be fully utilised. As some tasks are fixed to a workstation, or have to be done with multiple operators at once, an operator has a reduced number of jobs that can be assigned. Jobs that will be placed after the assignment of

restricted jobs may be to large to be fit in, causing the operators not to be utilised fully. Lastly, a big drop can be seen in station time for the last operator. This shows potential improvements that can be made within the line, as more tasks can be allocated to it in order to fully utilise the last operator, further increasing the total efficiency of the assembly line.

7. Conclusion and further research directions

This paper studies the problems of an assembly line balancing with the addition of mixed models, sequence-dependent setup times, zoning and multiple assignment constraints. This problem is mainly inspired from the current situation at VDL Nedcar. Compared to previous studies, the main novelty within this problem lies in the usage of a proportional cycle time constraint, were operators are allowed to exceed the takt time to a certain extent, as long as on average (according to the demand of each model) the takt time is met. Furthermore, an additional objective function of horizontal balancing is added within the proposed model. To this extent, a GA was developed in order to find better quality solutions in a shorter amount of time, compared to the current manual line balancing procedures.

The experimental results reveal numerous things, that can be used in practice in order to improve the quality and efficiency of existing line balances. First, the size of jobs (i.e. their processing times) should be kept relatively small compared to the takt time, e.g. below three, in order to be able to explore more solutions regarding the line balance. Secondly, setup times also have a negative effect on the solution space. If the range of setup times is large, then indirect precedence constraints are created, as large setup times are undesired. It is beneficial for companies to look at ways to reduce setup time between different jobs. These observations can also help with the process of data collection, as reducing the number of tasks and models considered during balancing decrease computational time, allowing managers to explore more options.

The computational study on a real assembly line at VDL Nedcar showed that the GA clearly outperforms the current balancing method, in terms of efficiency, smoothness (i.e. the deviation in station times) and computational time. Most notable was the reduction in the smoothness index by a factor of three, which directly relates to a reduction in idle times among operators and therefore also the amount of line stoppages. In addition, the new line balance also allows slight deviations in the takt time to be quickly addressed.

Further, the case study also showed that the size of mounting positions should be decreased to allow for feasible solutions. Also, the way assignment restrictions are set should be looked at closely. To strict restrictions on the assignment of tasks can impact the line balance, causing the smoothness to decreases and more operators to be necessary. The model has also shown potential to be integrated within a real-life assembly plant. Currently, VDL Nedcar is working on implementing this model as a Decision Support System for partially automating the line balancing procedures. In addition, insights gained from data collection also proved to be useful for setting up the environment to integrate these kinds of models in an industrial setting.

Interesting future research directions include enhancements to the proposed algorithm, with more complex crossover and mutation operators to improve solution quality further, combining the balancing with the assembly line sequencing problem to further evaluate the effects of line balancing and, lastly, implementation of the results in a simulation model to act as a digital twin for the assembly line.

8. Data availability statement

The data that support the findings of this study are available in GitHub at https://github.com/JDidden789/Line-Balancing-Instances.git. This data was derived from the following resources available in the public domain: https://assembly-line-balancing.de/. Data regarding the case study is not available, as participants of this research did not agree to share their data due to confidentiality.

Notes

1. Some papers mentioned in the literature review have been omitted
2. These instances can be found at https://github.com/JDidden789/Line-Balancing-Instances.git
3. The results file can be found at https://github.com/JDidden789/Line-Balancing-Instances.git
4. The actual increase cannot be stated due to confidentiality

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

J. B. H. C. Didden is a PhD candidate in the section Operation, Planning, Accounting and Control, Department of Industrial Engineering and Innovation Sciences at the Eindhoven University of Technology (TU/e). His current research focuses on the application of Multi-Agent System to solve complex scheduling problems in High-Mix-Low-Volume production environments.

E. Lefeber received the M.Sc. degree in applied mathematics and the Ph.D. degree in the subject of tracking control of nonlinear mechanical systems from the University of Twente, Enschede, The Netherlands, in 1996 and 2000, respectively. Since 2000, he has been an Assistant Professor with the Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands, where he was involved in the modelling and control of manufacturing systems from 2000 to 2015, and then he joined the Dynamics and Control Group in 2015. His current research interests include nonlinear control theory, in particular the control of drones and the control of platooning vehicles.

I. J. B. F. Adan is a Professor at the Department of Industrial Engineering of the Eindhoven University of Technology. His current research interests are in the modelling and design of manufacturing systems, warehousing systems and transportation systems, and more specifically, in the analysis of multi-dimensional Markov processes and queueing models.

I. W. F. Panhuijzen hold the function of Program Manager at VDL Nedcar. He has a BSc and MSc in Mechanical Engineering, with a specialisation in Manufacturing Networks, from Eindhoven University of Technology.

ORCID

I. J. B. F. Adan http://orcid.org/0000-0002-4493-6367

References

Alavidoost, M. H., M. H. Fazel Zarandi, Mosahar Tarimoradi, and Yaser Nemati. 2017. "Modified Genetic Algorithm for Simple Straight and U-shaped Assembly Line Balancing with Fuzzy Processing Times." *Journal of Intelligent Manufacturing* 28 (2): 313–336. http://dx.doi.org/10.1007/s10845-014-0978-4

Alghazi, Anas, and Mary E. Kurz. 2018. "Mixed Model Line Balancing with Parallel Stations, Zoning Constraints, and Ergonomics." *Constraints* 23 (1): 123–153.

Andrés, Carlos, Crisóbal Miralles, and Rafael Pastor. 2008. "Balancing and Scheduling Tasks in Parallel Assembly Lines with Sequence-dependent Setup Times." *European Journal of Operational Research* 213: 81–96.

Aydoğan, Emel Kızılkaya, Yılmaz Delice, Uğur Özcan, Cevriye Gencer, and Özkan Bali. 2019. "Balancing Stochastic U-lines Using Particle Swarm Optimization." *Journal of Intelligent Manufacturing* 30 (1): 97–111.

Becker, Christian, and Armin Scholl. 2009. "Balancing Assembly Lines with Variable Parallel Workplaces: Problem Definition and Effective Solution Procedure." *European Journal of Operational Research* 199 (2): 359–374. http://dx.doi.org/10.1016/j.ejor.2008.11.051

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2007. "A Classification of Assembly Line Balancing Problems." *European Journal of Operational Research* 183 (2): 674–693.

Delice, Yilmaz. 2019. "A Genetic Algorithm Approach for Balancing Two-sided Assembly Lines with Setups." *Assembly Automation* 39: 827–839.

Didden, Jeroen B. H. C. 2020. "Automating Balancing and Sequencing of Assembly Lines in An Automotive Manufacturing Plant." MSc, Technical University of Eindhoven. https://research.tue.nl/en/studentTheses/automating-balancing-and-sequencing-of-assembly-lines-in-an-autom.

Emde, Simon, Nils Boysen, and Armin Scholl. 2010. "Balancing Mixed-model Assembly Lines: A Computational Evaluation of Objectives to Smoothen Workload." *International Journal of Production Research* 48 (11): 3173–3191.

Ferrari, Emilio, Maurizio Faccio, Mauro Gamberi, Silvia Margelli, and Francesco Pilati. 2019. "Multi-manned Assembly Line Synchronization with Compatible Mounting Positions, Equipment Sharing and Workers Cooperation." *IFAC-PapersOnLine* 52 (13): 1502–1507. https://doi.org/10.1016/j.ifacol.2019.11.412

Ford, Henry, and Samuel Crowther. 1922. *My Life and Work*. New York: Doubleday.

Hamta, Nima, S. M. T. Fatemi Ghomi, F. Jolai, and M. Akbarpour Shirazi. 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. http://dx.doi.org/10.1016/j.ijpe.2012.03.013

İlker, Baybars. 1986. "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem." *Management Science* 32 (8): 909–932. http://pubsonline.informs.org/doi/abs/10.1287/mnsc.32.8.909.

Kim, Yeo Keun, Yeongho Kim, and Yong Ju Kim. 2000. "Two-sided Assembly Line Balancing: A Genetic Algorithm Approach." *Production Planning and Control* 11 (1): 44–53.

Lopes, Thiago Cantos, Giuliano Vidal Pastre, Adalberto Sato Michels, and Leandro Magatão. 2020. "Flexible Multi-manned Assembly Line Balancing Problem: Model, Heuristic Procedure, and Lower Bounds for Line Length Minimization." *Omega (United Kingdom)* 95: 102063. https://doi.org/10.1016/j.omega.2019.04.006

Naderi, Bahman, Ahmed Azab, and Katayoun Borooshan. 2019. "A Realistic Multi-manned Five-sided Mixed-model Assembly Line Balancing and Scheduling Problem with Moving Workers and Limited Workspace." *International Journal of Production Research* 57 (3): 643–661.

Roshani, Abdolreza, and Davide Giglio. 2017. "Simulated Annealing Algorithms for the Multi-manned Assembly Line Balancing Problem: Minimising Cycle Time." *International Journal of Production Research* 55 (10): 2731–2751. http://dx.doi.org/10.1080/00207543.2016.1181286

Salveson, M. E. 1955. "The Assembly Line Balancing Problem." *The Journal of Industrial Engineering*,6 (3): 18–25.

Scholl, Armin, Nils Boysen, and Malte Fliedner. 2013. "The Assembly Line Balancing and Scheduling Problem with Sequence-dependent Setup Times: Problem Extension, Model Formulation and Efficient Heuristics." *OR Spectrum* 35 (1): 291–320.

Sternatz, Johannes. 2014. "Enhanced Multi-Hoffmann Heuristic for Efficiently Solving Real-world Assembly Line Balancing Problems in Automotive Industry." *European Journal of Operational Research* 235 (3): 740–754. http://dx.doi.org/10.1016/j.ejor.2013.11.005

Tang, Qiuhua, Zixiang Li, Li Ping Zhang, and Chaoyong Zhang. 2017. "Balancing Stochastic Two-sided Assembly Line with Multiple Constraints Using Hybrid Teaching-learning-based Optimization Algorithm." *Computers and Operations Research*82: 102–113. http://dx.doi.org/10.1016/j.cor.2017.01.015

Thomopoulos, Nick T. 1970. "Mixed Model Line Balancing with Smoothed Station Assignments." *Management Science* 16 (9): 593–603. https://about.jstor.org/terms.

Uğur, Özcan, and Bilal Toklu. 2009. "Balancing of Mixed-model Two-sided Assembly Lines." *Computers and Industrial Engineering* 57 (1): 217–227.

Uğur, Özcan, and Bilal Toklu. 2010. "Balancing Two-sided Assembly Lines with Sequence-dependent Setup Times." *International Journal of Production Research* 48 (18): 5363–5383. https://www.tandfonline.com/action/journalInformation?journalCode = tprs20.

Wee, T. S., and M. J. Magazine. 1982. "Assembly Line Balancing As Generalized Bin Packing." *Operations Research Letters* 1 (2): 56–58.

Zacharia, P. Th., and Andreas C. Nearchou. 2012. "Multi-objective Fuzzy Assembly Line Balancing Using Genetic Algorithms." *Journal of Intelligent Manufacturing* 23 (3): 615–627.

Zacharia, P. Th., and Andreas C. Nearchou. 2013. "A Meta-heuristic Algorithm for the Fuzzy Assembly Line Balancing Type-E Problem." *Computers and Operations Research* 40 (12): 3033–3044. http://dx.doi.org/10.1016/j.cor.2013.07.012

Zhang, Zikai, Qiuhua Tang, Dayong Han, and Zixiang Li. 2019. "Enhanced Migrating Birds Optimization Algorithm for U-shaped Assembly Line Balancing Problems with Workers Assignment." *Neural Computing and Applications* 31 (11): 7501–7515. https://doi.org/10.1007/s00521-018-3596-9