

Sadeghi, Parisa; Diogo Rebelo, Rui; Soeiro Ferreira, José

Article

Using variable neighbourhood descent and genetic algorithms for sequencing mixed-model assembly systems in the footwear industry

Operations Research Perspectives

Provided in Cooperation with:

Elsevier

Suggested Citation: Sadeghi, Parisa; Diogo Rebelo, Rui; Soeiro Ferreira, José (2021) : Using variable neighbourhood descent and genetic algorithms for sequencing mixed-model assembly systems in the footwear industry, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 8, pp. 1-19,
<https://doi.org/10.1016/j.orp.2021.100193>

This Version is available at:

<https://hdl.handle.net/10419/246450>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

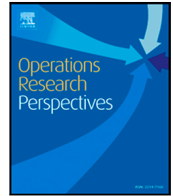
Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



<https://creativecommons.org/licenses/by/4.0/>



Using variable neighbourhood descent and genetic algorithms for sequencing mixed-model assembly systems in the footwear industry

Parisa Sadeghi^{a,*}, Rui Diogo Rebelo^a, José Soeiro Ferreira^{a,b}

^a INESC TEC - Technology and Science, Portugal

^b Faculty of Engineering, University of Porto, Campus da FEUP, Porto, Portugal

ARTICLE INFO

Keywords:

Mixed-model assembly line sequencing problem
Variable neighbourhood descent
Genetic algorithms
Dispatching rules

ABSTRACT

This paper addresses a new Mixed-model Assembly Line Sequencing Problem in the Footwear industry. This problem emerges in a large company, which benefits from advanced automated stitching systems. However, these systems need to be managed and optimised. Operators with varied abilities operate machines of various types, placed throughout the stitching lines. In different quantities, the components of the various shoe models, placed in boxes, move along the lines in either direction. The work assumes that the associated balancing problems have already been solved, thus solely concentrating on the sequencing procedures to minimise the makespan.

An optimisation model is presented, but it has just been useful to structure the problems and test small instances due to the practical problems' complexity and dimension. Consequently, two methods were developed, one based on Variable Neighbourhood Descent, named *VND-MSeq*, and the other based on Genetic Algorithms, referred to as *GA-MSeq*.

Computational results are included, referring to diverse instances and real large-size problems. These results allow for a comparison of the novel methods and to ascertain their effectiveness. We obtained better solutions than those available in the company.

1. Introduction

The production processes that rely on fashion have changed drastically. The footwear industry is one of those industries that has grown far beyond satisfying a basic human need and has had to modernise facilities, evolve planning methods, and respond to new and permanent market demands. In particular, there is a strong constraint to produce an increasing number of models of each category (women's, men's and children's shoes), according to people's desires, while reducing the required quantities for each category. Some companies, as in the case of this study, have invested in modern facilities that can handle such demands by running a large number of models simultaneously on the assembly lines. However, this requires excellent planning to ensure that the plants are well balanced and the operations are well coordinated.

The manufacture of shoes usually goes through essential processes such as cutting, stitching and assembling. A specific sequence of tasks must be followed for each shoe model, and the processing times for each task are different for each shoe model. In the cutting phase, materials are cut to prepare them for entering the production lines. In particular, the cutting line produces the material for the construction of the upper part of the shoe. The next step is the pre-stitching stage,

where the workpieces are assembled for the stitching line. In the stitching line, the upper part of the shoe is made and sewed by the stitching machines. Various edge treatments are performed on the material to complete the upper parts. At this stage, the expertise of the operator is of great importance. Finally, the upper and down parts are usually assembled together and the shoes are completely finished.

However, the focus of this research is on the stitching process that takes place at Kyaia, a large footwear company in Portugal whose production is almost entirely for worldwide export. The company has invested in completely new flexible assembly systems that must be adequately balanced and sequenced. These automated stitching systems comprise workstations consisting of specialised workers and machines, and the boxes containing the shoe components move in any direction between the workstations. Fig. 1 illustrates the situation. As mentioned earlier, there are different models in the lines; each model has a unique routing of various tasks, while each task in each routing has a different processing time.

Considering that the production plan changes rapidly, it is critical to know how many workstations are needed and how to assign operators

* Corresponding author.

E-mail addresses: parisa.sadeghi@inesctec.pt (P. Sadeghi), rui.d.rebelo@inesctec.pt (R.D. Rebelo), jsof@inesctec.pt (J.S. Ferreira).



Fig. 1. Company assembly systems.


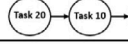




Different Shoe Models	Routing of each Model	Task Requirements	Machine Name	Operator Name
Model A 		Task 10	Ma2	Op2
Model B 		Task 20	Ma1, Ma4	Op1, Op6
Model C 		Task 30	Ma6	Op1, Op4
The processing time of a task may vary according to the model, e.g., the processing time of task 10 of model A is 7 minutes, but it takes 3 minutes in model B.		Task 40	Ma5, Ma3	Op3, Op4
Boxes move along the lines, with different quantities of the same shoe model and size (shoe size).		Task 50	Ma5	Op4

Fig. 2. Sequencing problem.

and tasks to each workstation. In short, the new practical mixed-model problem to solve is the sequencing of the tasks of operators and machines in the stitching lines to minimise the maximum completion time. The unique characteristics of the lines and their size justify the originality and complexity of the problem. Fig. 2 intends to help describe the sequencing problem.

Balancing and Sequencing problems are often linked when it comes to assembly systems. Balancing essentially involves planning the configuration of assembly systems, while sequencing involves determining the order of tasks at each workstation to optimise one or more objectives. According to Scholl and Becker [1], Becker and Scholl [2] and Sivasankaran and Shahabudeen [3], there are diverse types of assembly line problems. They can be classified as Simple Assembly Line (SAL), which refers to the assembly lines containing one model in the line, and Mixed Model Assembly line (MAL), which refers to the assembly lines containing different models of the same product at the same time in the line. Sivasankaran and Shahabudeen [3] present a literature review of assembly line problems, but most works about mixed-model lines are purely academic and relatively few deal with a real setting implemented results. Some references that concern real-world cases are Chen et al. [4], Oksuz et al. [5], and Sadeghi et al. [6]. Moreover, as Xiaobo and Ohno [7] and Lopes et al. [8] claim, mixed-model assembly lines are used in some production systems because they balance product diversity and production efficiency. Based on Xiaobo and Ohno [7], if a proper mixed-model balance is already achieved and the focus is on sequencing, we face a mixed-model sequencing problem. That is precisely the situation of this investigation (MALSP - Mixed-model Assembly Line Sequencing Problems). We assume existing solutions for the related balancing problems. The publications Sadeghi et al. [9] and Sadeghi et al. [6] describe how that is accomplished. General references on line sequencing are Dar-El and Cother [10], Graham et al. [11], Miltenburg [12], Tavakkoli-Moghaddam and Rahimi-Vahed [13] and Boysen et al. [14].

We have developed and tested an optimisation model for the sequencing problem to describe and understand the case under study. However, it could not provide optimal solutions for the company's real instances due to their complexity and size. Consequently, we resorted to approximation methods to obtain good quality solutions in a short computation time. The decision was made to choose Variable Neighbourhood Descent (VND) and Genetic Algorithms (GA). The proposed solution methods, VND-MSeq and GA-MSeq, are an appropriate adaptation of these metaheuristics to deal with the sequencing problem. We emphasise that a metaheuristic choice in a practical context is a straightforward and utterly sustained procedure, but it seemed that these have an excellent potential to be adapted to the case under study. Section 5 also discusses this choice in more detail. Moreover, VND has not been directly applied to the given (MALSP) problem to our knowledge.

The paper is organised as follows. Section 2 mainly contains a detailed description of the real case and the form of the decision problem. Section 3 presents the relevant literature. Section 4 describes the industrial sequencing problem and provides an optimisation model that contributes to understanding the problem and solving small instances. Section 5 presents the solution methods VND-MSeq, based on Variable Neighbourhood Descent, and the solution method GA-MSeq, based on Genetic Algorithms. The computational tests, results and comparisons of the two methods are given in Section 6. Finally, we draw the main conclusions in Section 7.

2. Industrial application

This section consists of a detailed description of the industrial application. There are two different automated stitching systems located in two separate places, as Fig. 1 shows. In both systems, a workstation is a location with a machine, which can be assigned to an operator. Thus, each workstation is a combination of a machine and an operator. Fig. 3



Fig. 3. A workstation in the stitching line.

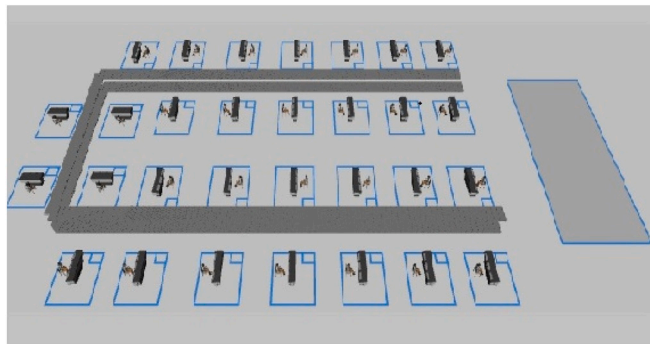


Fig. 4. Simplified design of the U-shaped smaller stitching line.

shows the workstations. Two or more workstations may be performing the same operation simultaneously, but it is possible that sometime later, the operation of one of them is changed. Solving the sequencing problems is determining the order and time of the operations at each workstation, optimising one or more objectives.

Each machine at a workstation faces a conveyor. The boxes containing the different shoe models of a certain size and colour move between the workstations in each direction. The number of maximum stopping points of the conveyor or workstations is different in the two systems; the larger has 190 stopping points, and the smaller has about 42. The layout of these two sewing systems is different, the larger consists of four parallel lines, the smaller has a *U* shape, but with serial manner. The larger system has, on one side, a warehouse (with boxes) connected to the conveyor. This one has a special passage, near the workstations and the warehouse, to change the boxes' position and allow them to move in any direction to the desired workstation. Hence, sometimes, the boxes go to the warehouse and get back to the right workstation, quickly and easily. The warehouse is at the end of the lines and there is a conveyor connecting them, making the movement easy and fast. In the smaller system there are two conveyors with some separation in height between them. Fig. 4 is a simplified drawing of this system with the two conveyors. Although the system has space for buffers, experience shows that it is not necessary to consider them in the modelling procedure.

There are diverse shoe models in the line simultaneously, depending on the customer's orders, and the only limit to the number of models is the line's capacity. The number of available shoe models in the lines is explained in Section 6 which is the computational results.

Always the models are placed in a box, the maximum number in the box is 10 pairs and this number is defined by the company, the

Table 1

Model routings of: SeqInstance.

Model	Task	Sequence	Processing time	Order quantity
A	10	2	7	20
A	20	1	8	20
A	30	3	4	20
B	10	1	3	10
B	20	2	5	10
B	30	4	5	10
B	40	3	3	10
B	50	5	6	10
C	20	2	7	29
C	40	1	3	29

balancing problem covers it, for more information see Sadeghi et al. [6]. As mentioned before, the created boxes and the quantities of them are not relevant for the sequencing problem, since they are defined. However, for the calculation of the processing time, that must be taken into account since it must be multiplied by the processing time of each pair. As shown in Fig. 2, the processing time of tasks is different according to the model.

To better understand the nature of the problem being solved, an instance named *SeqInstance* is provided. Table 1 contains the three models of *SeqInstance*, the tasks, and the respective model routings. Table 2 shows the mapping of the different boxes' tasks and models to the operators and machines. Note that the maximum quantity in each box is set to 10 pairs of shoes.

There are priority constraints between tasks and constraints on the scheduling of operators and machines. Table 3 shows the input data and the solutions to be obtained. What are the starting time (*S: Time*) and ending time (*E: Time*) of task 20 of model A in box 1 assigned to operator *Op1* and machine *Ma1*? This *Op1* and *Ma1* are the names of an operator and a machine and they are defined in the balancing so they are now the input parameter.

3. Relevant literature

This section presents a review of the relevant literature on MALSP, while also considering the practical problem of this work. Boysen et al. [14]) provided a classification for MALSP using a tuple notation that reflects features such as workstation boundaries, response to immediate work overload, setups, processing times, parallel workstations, concurrent work, number of workstations, start discipline, homogeneity of workstations, return speed, line layout, and goals. Bautista et al. [15] presented an extension of MALSP to deal with work overload and identified several mixed-product sequencing problems.

Table 2

Task assignment of SeqInstance (result of balancing) which is input data for sequencing.

Task	Model	Box no.	Box quantity	Operator	Machine
10	A	1	10	Op2	Ma2
20	A	1	10	Op1	Ma1
30	A	1	10	Op6	Ma4
10	A	2	10	Op2	Ma2
20	A	2	10	Op1	Ma1
30	A	2	10	Op6	Ma4
10	B	1	10	Op2	Ma2
20	B	1	10	Op1	Ma1
30	B	1	10	Op6	Ma4
40	B	1	10	Op4	Ma3
50	B	1	10	Op4	Ma5
20	C	1	10	Op1	Ma1
40	C	1	10	Op4	Ma3
20	C	2	10	Op6	Ma4
40	C	2	10	Op4	Ma3
20	C	3	9	Op6	Ma4
40	C	3	9	Op4	Ma3

Table 3

Input data and solutions that should be obtained for SeqInstance.

Task	Model	Box no.	Operator	Machine	S: Time	E: Time
20	A	1	Op1	Ma1	?	?

There are varied types of sequencing problems and various ways of solving them, such as accurate and approximate methods. For example, a mixed-model Just In Time (JIT) problem is solved by Miltenburg et al. [16] and Ng and Mak [17]. Miltenburg et al. [16] employed dynamic programming. Ng and Mak [17] mentioned it is difficult to solve such a problem mainly when there is a significant number of products with high production quantities, the solution method used is Branch and Bound (B&B). Xiaobo and Ohno [7] applied B&B to solve the problem of finding an optimum sequence that decreases the total line stoppage time for the mixed-model assembly line in a JIT production system. Bolat et al. [18] and Bard et al. [19] also adopted B&B methods to schedule jobs in an assembly line.

By prioritising all the tasks waiting for processing on the machines, dispatching rules can generate solutions. The prioritisation may consider task and machine attributes. When a machine is freed, a dispatching rule examines the waiting tasks and chooses the highest priority task. According to Pinedo and Chao [20], there exist numerous basic dispatching rules, but, in their work, they mentioned the most used ones. Xanthopoulos et al. [21] applied a series of simulation experiments involving seventeen dispatching rules and considered four. Kasemset and Pintaruean [22] examined six dispatching rules for the given sequencing problem of their work.

Below are some pertinent publications that deal with our research problems about the Variable Neighbourhood Search (VNS), VND and GA. VND and GA are the ones that are used in this work, and VNS is in the same category as VND.

Prandtstetter and Raidl [23] combined integer linear programming with VNS and considered six distinct neighbourhood structures; they mentioned that the results perform well on real-world instances. Yazdani et al. [24] presented a VNS algorithm to find the appropriate sequencing of operations on the machines to optimise the defined objective. In the study by Driessel and Mönch [25], job scheduling on parallel machines is addressed. Precedence constraints and sequence-dependent setup times are present in their paper and the solving approaches are several VNS methods. Chen et al. [4] scheduled a job production sequence with a hybrid approach, by VNS and Particle Swarm Optimisation (PSO), which achieves a near-optimal solution. Zhang et al. [26] mentioned that VNS algorithm has the excellent capability of local search with systematic neighbourhood search structures, while others such as Adamo et al. [27] used VND with a single neighbourhood structure.

GA metaheuristic is usually employed in sequencing and scheduling problems. Bierwirth and Mattfeld [28] used GA to reduce the mean flow-time. Cochran et al. [29] applied a multi-population GA to solve a multi-objective parallel machine scheduling problem. Multi-objective GA approaches were provided by Mansouri [30] for solving a JIT MALSP. Lei [31] carried out a study on multi-objective production scheduling and stated that the methods involved include Evolutionary Algorithm (EA), GA, Tabu Search, Simulated Annealing, Genetic Programming, Ant Colony Optimisation, Heuristics and Local search.

Regarding EA and GA, they are selected in almost half of the researched articles. In connection with the scheduling problem, Rebai et al. [32] solved the problem by three linear programs, a B&B algorithm, a local search and a GA. According to the computational results, the deviation of the metaheuristic solutions from the optimal solution is minimal. In another research, Huang et al. [33] employed GA, adaptive learning, and heuristics integrated into a sequential GA. Hence, highly satisfactory schedules were gained in a short period. Moradi and Zandieh [34] solved a MALSP with an imperialist competitive algorithm and compared their results with GA. As observed by Gen and Lin [35], among EA, distinct versions of GA since 1970 have attracted much attention; however, implementations vary widely. Rabbani et al. [36] mentioned that GA is better than PSO algorithm for the given multi-objective mixed-model two-sided assembly line sequencing problem. Zhang et al. [37] also used a GA for a multi-objective balancing and sequencing problem of a mixed-model assembly line, attaining better results than a non-dominated sorting genetic algorithm.

Concerning the footwear industry, the more generic references are mentioned here. Costa and Ferreira [38] did a simulation analysis for a flexible flow line scheduling problem in the footwear industry. Zangiacomi et al. [39] solved a production planning and scheduling problem for mass customisation systems; Süer et al. [40] used a three-phase hierarchical method for solving cell loading and scheduling; Chen et al. [41] proposed a GA on a hybrid flow shop scheduling problem. The research from Guimarães et al. [42] focused on the macro ergonomic evaluation of the workers, and Chen et al. [43] resort to simulation to configure the layouts of stitching lines. Chen et al. [44] aimed at solving a balancing problem by using a hybrid GA; Ulutas and Islier [45] considered a dynamic facility layout problem, and Dang and Pham [46] designed an assembly line by using simulation. Lopes et al. [47] contemplated a scheduling problem applied to an injection moulding machine in a footwear company. Quyen et al. [48] solved a resource-constrained assembly line balancing problem in a single model line and, as already mentioned, Sadeghi et al. [6] dealt with MALBP in this industry. Fani et al. [49] combined simulation and optimisation into a footwear company model to solve a planning and scheduling problem.

The literature review that the authors carried out did not reveal any sequencing problem similar to the complex one treated in this article. That is not surprising since the case under study originates from new automatic assembly systems designed specifically for footwear production.

Attending to Boysen et al. [14], we can identify many characteristics of the industrial problem. It is a mixed-model sequencing problem; considering the homogeneity of workstations, they are composed of operators and machines with the same characteristics (operator and machine), but their capabilities are distinct. Regarding the layout workstations are in a serial manner; in what concerns parallelism, some workstations can work on the same operation simultaneously. The processing times of tasks are fixed. On the other hand, some realistic features do not seem covered in that work. The systems have conveyors that allow the boxes with components to move in distinct directions. A warehouse is accessible automatically whenever needed. Besides, the unity is not a pair of shoes but pairs in boxes, moving in the systems. Consequently, the solution methods proposed also have merit for the originality of the adaptation and the implementation's success, as mentioned in the conclusions section.

4. The problem statement

The problem type studied in this paper has already been classified as MALSP. The footwear company has two distinct stitching systems, which are explained in detail in Section 2. Before proceeding to the sequencing problem, the balancing problem is resolved, which the authors addressed in other works. Notice that the objectives of the balancing problem are to minimise the number of workstations and smooth the operators' workload. In any case, and due to the strong interaction with sequencing, we provide some general statements about balancing. The equipment selection is considered in the balancing because the operators have different skills and the machines are of various types. When production orders arrive at the lines, the boxes are prepared with a maximum quantity of 10 pairs per box; then, daily line balancing is carried out, with the assignment of each task of a box model to an operator and a machine, but without programming the corresponding sequencing.

We propose now an optimisation model for the company mixed-model assembly sequencing problem. In particular, it will help better to understand the situation and the development of resolution methods.

Objective: The objective is to minimise the makespan, i.e., minimise the last completion times of all the tasks (associated with all boxes) of all operators and machines. This objective was selected after discussions with managers and the research team working with the company.

The following indices, parameters and decision variables are considered in the mixed-integer optimisation model.

Indices:

- j, j' : index for model, $j, j' \in [J]$, $[J] = \{1, 2, \dots, J\}$
- i, i' : index for task, $i, i' \in [I]$, $[I] = \{1, 2, \dots, I\}$
- k, k' : index for number of boxes, $k, k' \in [K]$, $[K] = \{1, 2, \dots, K\}$
- o : index for operator, $o \in [O]$, $[O] = \{1, 2, \dots, O\}$
- m : index for machine, $m \in [M]$, $[M] = \{1, 2, \dots, M\}$

Input parameters:

- P_{ij} = processing time of task i for model j
- S_{ijk} = sequence of task i for model j
- Q_{jk} = quantity of model j in box k
- EC_{ij} = earliest completion time of task i for model j
- $X_{ijko} = 1$: if task i of model j of box k is assigned to operator o ; 0: otherwise
- $Y_{ijkm} = 1$: if task i of model j of box k is assigned to machine m ; 0: otherwise

Av = available time

GM = great number

Decision variables:

- C_{ijk} = completion time of task i of model j of box k
- $U_{ijki'j'k'} = 1$: if completion time of task i' of model j' of box k' is greater than completion time of task i of model j of box k ; 0: otherwise
- $Cmax$: the largest completion time

Optimisation model:

Minimise $(Cmax)$

subject to:

$$C_{i'jk} \geq C_{ijk} + P_{i'j}Q_{jk}, \quad i, i' \in [I], \quad j \in [J], \quad k \in [K], \quad S_{i'j} > S_{ij} \quad (C1)$$

$$\begin{aligned} C_{ijk} &\geq C_{i'j'k'} + P_{ij}Q_{jk} - GM(U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j \neq j'), \quad k, k' \in [K], \\ o \in [O] \quad X_{ijko}X_{i'j'k'o} &= 1 \end{aligned} \quad (C2)$$

$$\begin{aligned} C_{i'j'k'} &\geq C_{ijk} + P_{i'j'}Q_{j'k'} - GM(1 - U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j \neq j'), \quad k, k' \in [K], \\ o \in [O] \quad X_{ijko}X_{i'j'k'o} &= 1 \end{aligned}$$

$$\begin{aligned} C_{ijk} &\geq C_{i'j'k'} + P_{ij}Q_{jk} - GM(U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j = j'), \quad k, k' \in [K], \quad (k \neq k'), \\ o \in [O] \quad X_{ijko}X_{i'j'k'o} &= 1 \end{aligned} \quad (C3)$$

$$\begin{aligned} C_{i'j'k'} &\geq C_{ijk} + P_{i'j'}Q_{j'k'} - GM(1 - U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j = j'), \quad k, k' \in [K], \quad (k \neq k'), \\ o \in [O] \quad X_{ijko}X_{i'j'k'o} &= 1 \end{aligned}$$

$$\begin{aligned} C_{ijk} &\geq C_{i'j'k'} + P_{ij}Q_{jk} - GM(U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j \neq j'), \quad k, k' \in [K], \\ m \in [M] \quad Y_{ijkm}Y_{i'j'k'm} &= 1 \end{aligned} \quad (C4)$$

$$\begin{aligned} C_{i'j'k'} &\geq C_{ijk} + P_{i'j'}Q_{j'k'} - GM(1 - U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j \neq j'), \quad k, k' \in [K], \\ m \in [M] \quad Y_{ijkm}Y_{i'j'k'm} &= 1 \end{aligned}$$

$$\begin{aligned} C_{ijk} &\geq C_{i'j'k'} + P_{ij}Q_{jk} - GM(U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j = j'), \quad k, k' \in [K], \quad (k \neq k'), \\ m \in [M] \quad Y_{ijkm}Y_{i'j'k'm} &= 1 \end{aligned} \quad (C5)$$

$$\begin{aligned} C_{i'j'k'} &\geq C_{ijk} + P_{i'j'}Q_{j'k'} - GM(1 - U_{ijki'j'k'}), \\ i, i' \in [I], \quad j, j' \in [J], \quad (j = j'), \quad k, k' \in [K], \quad (k \neq k'), \\ m \in [M] \quad Y_{ijkm}Y_{i'j'k'm} &= 1 \end{aligned}$$

$$C_{ijk} \leq Av, \quad i \in [I], \quad j \in [J], \quad k \in [K] \quad (C6)$$

$$C_{ijk} \geq EC_{ij}Q_{jk}, \quad i \in [I], \quad j \in [J], \quad k \in [K] \quad (C7)$$

$$C_{ijk} \leq Cmax, \quad i \in [I], \quad j \in [J], \quad k \in [K] \quad (C8)$$

Here are some explanations about the constraints:

(C1): The sequence of any task i for each model j inside box k must be respected.

(C2) and (C3): Each operator o must work on only one task i of model j of box k , at any time (tasks of the same operator do not overlap in time).

(C4) and (C5): Each machine m must work on only one task i of model j of box k , at any time (tasks of the same machine do not overlap in time).

(C6): The completion time of task i of model j of box k should be less than the available time.

(C7): The completion time of task i of model j of box k cannot be less than the earliest competition time.

(C8): Cmax (Makespan) is the longest completion time.

The optimisation model was developed with CPLEX optimisation Studio. It helped gain insight into the problem's structure, solve small size instances, and evaluate the approximate methods VND-MSeq and GA-MSeq, to present subsequently. However, it was not possible to attain optimal solutions for the large real industrial instances. We will return to this subject in Section 6.

5. Solution methods

This section proposes and explains two solution methods, first the VND-MSeq and then the GA-MSeq. The link: <https://drive.inescetec.pt/s/rZEN9nLKfRcemLX> gives access to more details, examples and images that can help clarify the subjects of this section.

5.1. Solution method: VND-MSeq

VND is chosen as one of the solution approaches to the sequencing problem (MALSP). This metaheuristic selection took into account its adaptability to the problem, which is very important, and the existing experience. It should be noted that this method also has the advantage of facilitating diversification when testing diverse neighbourhood structures. Further, to our knowledge, VND was not directly applied to MALSP.

Variable Neighbourhood Descent Mixed-Model Sequencing (VND-MSeq) is composed of two parts. The first part is a constructive heuristic to generate the initial solution; in the second part, VND is used to improve it. The assignment of the tasks, related to the various models, to the operators and machines, is determined by the approach provided in Sadeghi et al. [6]. Now, the issue is the use of VND-MSeq to sequence the tasks/boxes on the production lines.

Table 4
LNS values for different tasks.

Task	Model	Box no.	LNS value
10	A	1	1
20	A	1	2
30	A	1	0
10	A	2	1
20	A	2	2
30	A	2	0
10	B	1	4
20	B	1	3
30	B	1	1
40	B	1	2
50	B	1	0
20	C	1	0
40	C	1	1
20	C	2	0
40	C	2	1
20	C	3	0
40	C	3	1

5.1.1. VND-MSeq: Constructive heuristic

Different dispatching rules are used to produce possible initial solutions for VND-MSeq. Thus, the one which is closer to the average objective function is chosen. According to Pinedo and Chao [20] and as mentioned earlier, there are many dispatching rules, but the ones applied in this section are as follows:

1. Largest Number of Successors (LNS);
2. Critical Path (CP);
3. Longest Processing Time (LPT);
4. Shortest Processing Time (SPT);

The first rule, which is the *Largest Number of Successors*, is used in three distinctive ways, and the *Critical Path* method is applied in two manners. The *Longest Processing Time* and *Shortest Processing Time* methods are each used in one way. Therefore, seven *Tables of Tasks (TT)* with different arrangements, and consequently, seven possible initial solutions, are constructed. As mentioned before, the one closer to the average objective function will be chosen as the initial solution. Moreover, a TT comprises all the available tasks; by changing the task's position in that table, new solutions will be achieved. Next, we describe the applied dispatching rules in more detail.

Largest Number of Successors (LNS): LNS rule gets a task and calculates the number of tasks next to it in that task's model routing. The LNS value for task 10 of model A is 1 because there is only one task after task 10 in that model's routing diagram. This value for task 20 of the same model (A) is 2. Table 4 shows LNS values for all tasks of *SeqInstance*. This table is a sample of a TT for the given example.

After calculating these values, they are sorted, considering the LNS values from the higher to the lower ones. This sorting process is done in three ways, and consequently, they could generate three different solutions. These three ways are explained in detail with the example in the given link at the beginning of this section (Section 5).

Critical Path (CP): CP rule, for each task, adds the processing times of the selected task and of the tasks that are next to it, in the routing diagram of the picked task. Table 5 displays the CP values for *SeqInstance*. Then, they are sorted in a descending mode but in two different ways. Therefore, they build two sequences of tasks and as a result, two different initial solutions are generated. Generally, the CP method gives priority to the tasks with higher CP values. The tables related to these two ways are explained in the given link at the beginning of this section (Section 5).

Longest Processing Time (LPT) and Shortest Processing Time (SPT): These two rules consider each task's processing time without any attention to its sequence in the related model routing diagram. Tasks are sorted in the LPT from the higher to the lower values, while on the SPT method, it happens oppositely: the sorting process is from the

Table 5
Table of tasks - CP values for different tasks.

Task	Model	Box no.	Box quantity	Box processing time	CP value
10	A	1	10	70	110
20	A	1	10	80	190
30	A	1	10	40	40
10	A	2	10	70	110
20	A	2	10	80	190
30	A	2	10	40	40
10	B	1	10	30	220
20	B	1	10	50	190
30	B	1	10	50	110
40	B	1	10	30	140
50	B	1	10	60	60
20	C	1	10	70	70
40	C	1	10	30	100
20	C	2	10	70	70
40	C	2	10	30	100
20	C	3	9	63	63
40	C	3	9	27	90

Table 6
Initial solution following by LNS, it is also a table of tasks.

Task	Model	Box no.	S: Time	E: Time
10	B	1	0	30
20	B	1	30	80
20	A	1	80	160
20	A	2	160	240
40	B	1	80	110
10	A	1	160	230
10	A	2	240	310
30	B	1	110	160
40	C	1	0	30
40	C	2	30	60
40	C	3	110	137
30	A	1	230	270
30	A	2	310	350
50	B	1	160	220
20	C	1	240	310
20	C	2	160	230
20	C	3	350	413

lower to the higher values. Since the sequence of the tasks in these two approaches is not considered, then the constructive heuristic changes the task position of a TT task to create a feasible sequence of tasks. Two tables of the example related to the LPT and SPT are presented in the given link at the beginning of this section (Section 5).

Feasible solutions are created by knowing which operator and machine are assigned to each TT task and considering the earliest starting time of the tasks. Operators and machines should not overlap in time to perform the tasks. As an example, the initial solution of Table 6 is obtained by the first way of the method LNS. Finally, after generating the seven potential initial solutions for improvement by the metaheuristic VND, the one that is closest to the average is selected as the initial solution.

In the next section, the improvement process will be explained.

5.1.2. VND-MSeq: Variable neighbourhood descent

The metaheuristic VND is now applied as an improvement procedure. According to Driessel and Mönch [25], VNS often outperform other methods for scheduling based approaches. Additionally, according to Roshanaei et al. [50], VNS metaheuristic technique, has quickly gained widespread success. Also, according to Amiri et al. [51], VNS is one of the famous metaheuristics, which has been successfully applied to solve optimisation problems. It can escape from local optima through systematic changes of the neighbourhood structures during the search process. The used VND follows the main structures of VNS, but the change of neighbourhoods is defined in a deterministic way. Five

Tasks' Position	Task Name-Model Name-Box Number	First Step (Swapping Tasks in Positions 5 and 10)	Second Step (Changing Positions of the Related Tasks)	Third Step (Adding the Remaining Tasks)
1	20-A-1			20-A-1
2	10-A-1		10-B-1	10-B-1
3	30-A-1		20-B-1	20-B-1
4	20-A-2		40-B-1	40-B-1
5	10-A-2	30-B-1	30-B-1	30-B-1
6	30-A-2			10-A-1
7	10-B-1			30-A-1
8	20-B-1			20-A-2
9	40-B-1			50-B-1
10	30-B-1	10-A-2	10-A-2	10-A-2
11	50-B-1		30-A-2	30-A-2
12	40-C-1			40-C-1
13	20-C-1			20-C-1
14	40-C-2			40-C-2
15	20-C-2			20-C-2
16	40-C-3			40-C-3
17	20-C-3			20-C-3

Fig. 5. Ns1, swap.

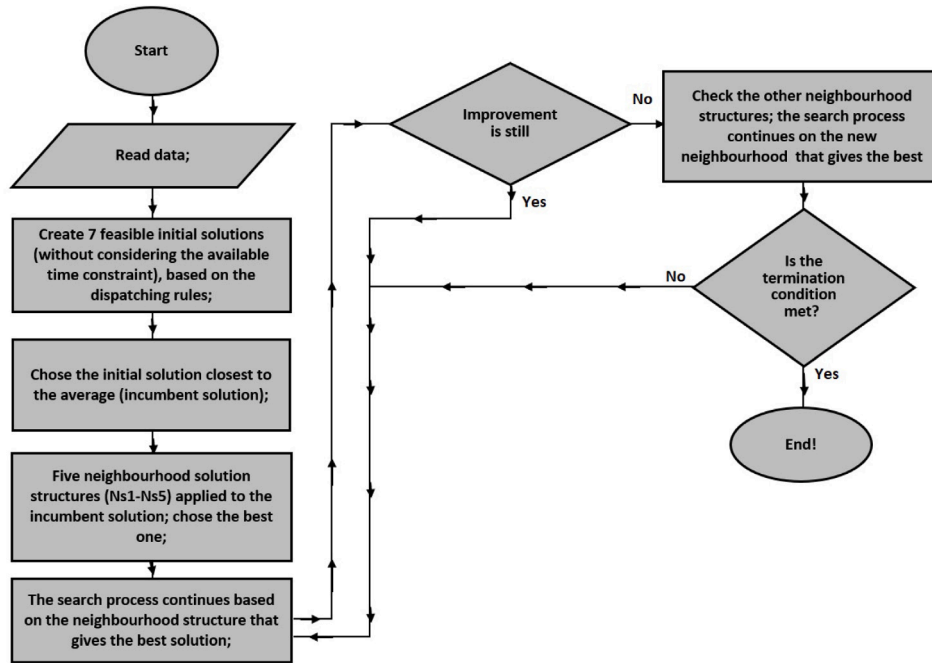


Fig. 6. General rules of VND-MSeq solution method.

neighbourhood structures (*Ns1* - *Ns5*) will be designed, in the following, to apply VND.

As mentioned in the previous section, a TT represents a solution. The position of each task in the table shows its priority for assignment to each operator and machine. By changing the position of a task in a solution, it is possible to create a new solution. Therefore, the concept of neighbourhood structures can be used to change task positions in any solution. However, any change to the position of a task can result in an infeasible TT. Consequently, each neighbourhood structure should be applied in such a way as to avoid infeasibility. The dimension of each neighbourhood structure is firstly based on the given problem and, secondly, related to the defined numbers of search iterations and of neighbour solutions, which should be searched in each iteration. These aspects will be explained in more detail later.

A key issue linked with each neighbourhood structure is related to feasibility, which should occur in all *Ns1* to *Ns5*. By randomly (or by any other rule) changing the sequence of a task, an infeasible TT

may be obtained. One possibility to avoid this situation is to change the position of the tasks in the table so that tasks that are before to the task selected in the routing diagram are considered first. This rule is called *Precedence Preserving* and used in the whole paper to avoid infeasibility concerning model routings. Finally, the objective function will be calculated for the sequences of tasks (different solutions).

Ns1: This neighbourhood is based on the *Swap* structure and works as follows. Firstly, random numbers between zero and the number of available tasks that show each task's position are generated. For small instances, two random numbers are considered for one swap and for large instances, eight random values are considered for four swaps (eight tasks two by two). Next, the tasks are swapped by taking the random values two by two resulting in a new sequence of tasks. Besides, feasibility should be checked and the position of tasks should be changed if necessary. In the *SeqInstance*, which is a small one, by considering the incumbent solution of Table 6 and by generating two random values, a new TT is organised, as shown in Fig. 5. Generally,

in this neighbourhood space, three steps are taken as follows and as displayed in Fig. 5.

1. The first step involves the swapping of the selected tasks by changing their positions in the TT.
2. The second step includes a change in the positions of the tasks in the same model routing with the previous step's selected tasks. These tasks are the ones with a smaller number of sequence when compared to the picked tasks in the routing diagram. They are named as *Related* tasks. That means that they should be assigned first (the precedence preserving rule).
3. The third step requires a sequence by adding the remaining tasks (tasks that have not been considered and changed before) in the TT.

The number of solutions to be searched at each iteration does not involve all neighbour solutions because of time-consuming. The number of repetitions is based on parameter tuning and specially Taguchi Method (*TMet*), which is explained in Section 6.

Ns2: This neighbourhood structure is similar to the previous one (Ns1), but instead of being swapped, tasks are *transferred*; it is based on the structure *Transfer*. Ns2 (as in Ns1) uses two random numbers for small instances and eight random numbers for the large ones. Tasks from the first position (the first random value) are transferred to the second position (the second random value). After that, to meet the sequencing constraint, the tasks, which are before the transferred task, should also be transferred (second step, the precedence preserving rule). The last step, as previously mentioned, is adding the remaining tasks. The number of searches for new solutions is the same as in Ns1, which is tuned based on *TMet*. A figure related to the example and NS2 structure is in the given link at the beginning of Section 5.

Ns3: This neighbourhood is based on *Transfer* structure but uses *Intelligence*, which takes advantage of the critical path. Ns3 is named *Transfer, Intelligence*. Firstly, the critical path is obtained, and the critical path tasks are transferred one by one to the beginning of the TT. This gives them a priority for allocation. Afterwards, the other two steps (2 and 3) explained before are followed. The transference of all the critical tasks forms the first neighbour, then the next neighbour is found by transferring some of the critical tasks. This process is repeated until all neighbours of a current solution are considered. The number of searched solutions in Ns3 is dependent on the dimension of each problem. In general, 150 solutions have been used for large instances. The related figure is in the link at the beginning Section 5.

Ns4: It is based on *Transfer, Intelligence and Random Swap*. In details, Ns4 transfers all the critical path tasks and swap other tasks randomly. The structure of the transferred critical tasks in this neighbour structure is different from Ns3. This structure transfers in groups of two the critical path tasks to the beginning of the TT. Steps 2 and 3 are also applied in the same way as in all other neighbourhood structures. Moreover, by choosing four random numbers in large instances, swapping tasks are performed for four different tasks. For small instances, a swap is not applied. This structure searches all the neighbour solutions of a current solution. The related figure is in the link at the beginning Section 5.

Ns5: This neighbourhood is based on *Transfer, Intelligence and Random Transfer*. In details, Ns5 transfers critical path tasks and transfers other tasks randomly. This structure is the same as Ns2, with the only distinction associated with the tasks chosen, which are the critical path tasks. For this reason, we did not provide a Figure. However, only one transfer for the critical path tasks is applied for small instances, as mentioned earlier. In large instances, two transfers are done, one for the critical tasks and the other for the remaining tasks. The number of search processes in this structure is the same as in Ns1 and Ns2.

VND-MSeq method starts with all the neighbourhood structures and applies them to the current solution. Then, the best solution is chosen as the incumbent one. The search process stays on the structure that gives the best solution and searches that neighbourhood until obtaining an

Given Dispatching Rules: 1-LNS-First Way 2-LNS-Second Way 3-LNS-Third Way 4-CP-First Way 5-CP-Second Way 6-LPT 7-SPT
Change the sequence of tasks in the previous solution by swapping tasks and applying three given steps (section 5.1.2.)
Repeat the above process, nine times for large instances, and three times for the small ones.
Switch to the next dispatching rule, and do the above processes until obtaining 70 solutions for the medium and large instances and 28 for the small ones.

Fig. 7. Generating initial populations.

improvement. Otherwise, all other structures are applied to the current solution, and then the search moves towards the best one. This process repeats to the number of iterations established, which is obtained based on parameter tuning and especially *TMet* (see Section 6). Fig. 6 portrays the general rules of VND-MSeq.

5.2. Solution method: GA-MSeq

The second solution method is based on GA and is called Genetic Algorithm Mixed-Model Sequencing (*GA-MSeq*). According to Gen and Lin [35] GA, among evolutionary algorithms, has attracted much attention from the 1970s, and at the same time, it is well known that GA has an excellent exploration capability.

GA-MSeq method will also enable comparison and evaluation of the results achieved with VND-MSeq. Different dispatching rules, such as those in the previous section, are used for generating initial populations but in different ways. The procedures are explained in the following and consist of two main steps: First, the creation of the initial populations and second, the improvement procedure by a GA.

5.2.1. GA-MSeq: Initial Population

The first step is the generation of the initial population. A diverse set of initial solutions is used to facilitate the exploration of the search space. Moreover, a constructive heuristic is adopted to avoid getting infeasible solutions. The initial populations are provided by using four dispatching rules (LNS, CP, LPT, SPT), which were explained in Section 5.1.1. Based on these four rules, tasks are sorted in seven different ways, as in the previous solution method (VND-MSeq). The assignment of tasks to the operators and machines starts from the first task of TT and continues following their orders. With this procedure, seven solutions are generated; therefore, to reach other solutions, after creating one primary solution based on each dispatching rule, changes should be made in the TT. These changes modify the positions of different tasks to get new solutions. For GA-MSeq, initial population sizes are 70 and 28, large and small instances, respectively.

To achieve 70 solutions in large instances, nine results are obtained in every seven preliminary solutions and, in small instances, three results are obtained to have 28 solutions. Each solution should be feasible, taking into account all the constraints that are given in the optimisation model, but not considering the available time, as in the previous method VND-MSeq. To sum up, the general process for producing initial populations is shown in Fig. 7.

Operator	Machine	Task	Model	Box
First part		Second part		

Fig. 8. Configuration of a gene.

Gene N.er	Workstation N.er	(Operato, Machine)	(Task, Model, Box)
1	1	(Op2, Ma2)	(10, B, 1)
2	2	(Op1, Ma1)	(20, B, 1)
3	2	(Op1, Ma1)	(20, A, 1)
4	2	(Op1, Ma1)	(20, A, 2)
5	4	(Op4, Ma3)	(40, B, 1)
6	1	(Op2, Ma2)	(10, A, 1)
7	1	(Op2, Ma2)	(10, A, 2)
8	3	(Op6, Ma4)	(30, B, 1)
9	4	(Op4, Ma3)	(40, C, 1)
10	4	(Op4, Ma3)	(40, C, 2)
11	4	(Op4, Ma3)	(40, C, 3)
12	3	(Op6, Ma4)	(30, A, 1)
13	3	(Op6, Ma4)	(30, A, 2)
14	5	(Op4, Ma5)	(50, B, 1)
15	2	(Op1, Ma1)	(20, C, 1)
16	3	(Op6, Ma4)	(20, C, 2)
17	3	(Op6, Ma4)	(20, C, 3)

Fig. 9. A Chromosome for the *SeqInstance* instance.

5.2.2. GA-MSeq: Genetic algorithm

The second step is the improvement procedure which is a GA. In this step the encoding and decoding of the solutions (chromosomes) is a critical procedure. In general, it should take into account the problem under study. There are various types of encoding, such as permutation encoding, binary encoding and others. Also, different methods for the decoding are available (see Kucukkoc and Zhang [52] and Yu et al. [53]).

Solution encoding and decoding:

In this study, each chromosome represents a feasible solution. Solution encoding is necessary and it is based on permutation. Fig. 8 depicts the configuration of a gene, which is composed of two distinct parts. One considers the allocation (line balancing) of operators and machines. The other part is about the process information or sequencing that is the task, model, and box.

The allocation part assigns any task of a model (from any box) to an operator and machine with the ability to perform it. This part is not the focus of the work, as it is supposed to have prior knowledge of these allocations. However, they are necessary for the definition of gene provided, to lead to a viable solution, more quickly.

The permutation of the genes creates a chromosome. The position of the genes determines their order of assignment to the given operator and machine. Therefore, the permutation of the genes, which forms a solution, should be feasible, and it is achieved by having a feasible sequence of tasks. Returning to the example from Section 2, we have:

1. Task assignment in Table 2;
2. An initial solution in Table 6.

The chromosome related to this solution is depicted in Fig. 9.

The used decoding method is close to list scheduling, a method adopted in many types of research, according to Yu et al. [53]. This method firstly creates a task list, then picks tasks sequentially and schedules them as early as possible, on the chosen machine, by the available assignment rule. Fig. 10 describes the used decoding.

Selection process: This process includes both the selection of a gene and a chromosome to apply different operators such as crossover and mutation. Moreover, the selection is also used to choose a new generation in each iteration. GA-MSeq employs two types of selection methods:

1. Randomised Selection: As the name implies, this method is completely randomised and the GA-MSeq chooses random values, which can identify a chromosome or clarify the position of a gene.
2. Tournament Selection: In the reproduction of a new population, the chromosomes are sorted according to fitness. Then, the individual having the highest fitness (smallest makespan) is the winner.

The randomised selection is applied to choose parents for a crossover, picking chromosomes for mutation and selecting genes for crossover and mutation. Furthermore, the tournament selection is used to select the chromosomes as the new generation in each iteration.

Crossover and mutation: Crossovers and mutations are applied while taking into account the sequencing problem. Two-point crossover is used in two different ways: *Two-point crossover-1* and *Two-point crossover-2*. Mutation is applied in three ways based on Inversion and Insertion, but in a particular way, named as: *Mutation inversion*, *Mutation insertion* and *Mutation insertion-critical genes*. The details are presented in continue.

Two-point crossover-1: Firstly, the parents based on the selection process as explained above are chosen. Secondly, since two points in this type of crossover are needed, two random numbers between zero and a maximum number of available genes are generated. These two numbers determine the positions of two genes. All the genes of each parent between two selected points are transferred to each child, in the same positions as the parents. The precedence preserving rule is used to avoid infeasibility, by considering the routing diagram of the task in the selected chromosome and changing the position of genes. The other genes of each child are the same as those of other parents, which were not used before in this child. This type of crossover and the applied rules are explained in detail on the example (*SeqInstance*) given in Figs. 11 and 12. Fig. 11 shows two chosen chromosomes and the genes between two points, which are between positions 6 and 12 in colour. Fig. 12 depicts the offsprings, and the cells with dots are the *Related* genes or the ones whose positions are changed according to the precedence preserving rule.

Two-point crossover-2: In this crossover, after choosing parents two points are selected. Only the genes of the first and second points are copied to each other's child, considering the precedence preserving rule. According to this rule, if a gene is copied to a child, then its *Related* genes, which are of the same model and box, are also copied to that child by considering the sequencing constraint of that model routing. More explanations with helpful figures on the instance *SeqInstance* are presented in the link that is given at the beginning of Section 5.

Mutation inversion: In general, a mutation is any type of change in a chromosome. However, after choosing a chromosome, this mutation randomly selects two mutation points and reverses the order of the genes between these two points. The rule of precedence preserving is applied in a way to avoid infeasible solutions. Fig. 13 illustrates this type of mutation.

Mutation insertion: The second mutation is based on the insertion of genes from the first position into the second one, considering the sequence of tasks in the selected model routing (precedence preserving rule). In large instances, four random numbers are chosen to apply two insertions, and in small instances, only one insertion is applied. More explanations about this are provided at the given link at the beginning of Section 5.

Mutation insertion-critical genes: The primary issues in this mutation are the same as the previous one, but with one difference associated with the selection of the genes for insertion. In this mutation, the first and second genes should be chosen from the critical ones, which are related to the critical tasks, and they should be transferred to the beginning of the picked chromosome. Therefore, through this method, we give priority to the critical tasks for the assignment. Moreover, genes are transferred considering their routing diagram.

Data: A chromosome

Result: A complete solution and, as a result, Cmax

```

1 for Counter1 = 1 to Counter1 ≤ MaxNumberOfGenes do
2   Select the gene, with the index equal to Counter1, geneCounter1;
3   Get geneCounter1 and pick the machine from it;
4   Get geneCounter1 and pick the task from it, and get the earliest starting time of that task
   (EarliestStartingTime);
5   if machineFreeTime = EarliestStartingTime and machine is free until the ending time of
   that task: machineFreeTime + DurationofTask then
6     Pick the operator from geneCounter1;
7     if operatorFreeTime = EarliestStartingTime and is free until the ending time of that
     task: EarliestStartingTime + DurationofTask then
8       Set the starting time and ending time of that task equal to:
       StartingTime = EarliestStartingTime &
       EndingTime = EarliestStartingTime + DurationofTask;
9       Update the free time and occupation time of that operator and machine;
10    end
11  else
12    Find another time early as possible (> EarliestStartingTime) which that machine is
    free until the ending time of that task, define it as NewEarliestStartingTime;
13    if Operator of geneCounter1 is free at NewEarliestStartingTime and be free until the
    ending time of that task NewEarliestStartingTime + DurationofTask then
14      Set the starting time and ending time of that task equal to:
      StartingTime = NewEarliestStartingTime &
      EndingTime = NewEarliestStartingTime + DurationofTask;
15      Update the free time and occupation time of that operator and machine;
16      Update EarliestStartingTime of the remaining tasks in the geneCounter1 which are
      in the same box;
17    else
18      Put EarliestStartingTime = NewEarliestStartingTime;
19      Find another time early as possible (> EarliestStartingTime) which that operator
      is free until the ending time of that task, define it as
      NewNewEarliestStartingTime.;
20      if Machine of geneCounter1 is free at NewNewEarliestStartingTime and be
      free until the ending time of that task
      NewNewEarliestStartingTime + DurationofTask then
21        Set the starting time and ending time of that task equal to:
        StartingTime = NewNewEarliestStartingTime &
        EndingTime = NewNewEarliestStartingTime + DurationofTask;
22        Update the free time and occupation time of that operator and machine;
23        Update EarliestStartingTime of the remaining tasks in the geneCounter1 which
        are in the same box;
24      else
25        Put EarliestStartingTime = NewNewEarliestStartingTime;
26        go to line 12.
27      end
28    end
29  end
30 end
31 Calculate the maximum finishing time from all the genes of the given chromosome;

```

Fig. 10. The decoding method.

The mutation process is similar to the previous one, but with the fundamental difference noted earlier. For this reason, we do not include more examples. Below, there are more details.

1. Random selection of a chromosome.
2. Determination of the critical genes of that chromosome.
3. Random selection of two critical genes.

First Chromosome			Second Chromosome		
Workstation No.	(Operator, Machine)	(Task, Model, Box)	Workstation No.	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)	1	(Op2, Ma2)	(10, B, 1)
2	(Op1, Ma1)	(20, A, 2)	2	(Op1, Ma1)	(20, B, 1)
1	(Op2, Ma2)	(10, A, 1)	4	(Op4, Ma3)	(40, B, 1)
1	(Op2, Ma2)	(10, A, 2)	3	(Op6, Ma4)	(30, B, 1)
4	(Op4, Ma3)	(40, C, 1)	5	(Op4, Ma5)	(50, B, 1)
2	(Op1, Ma1)	(20, C, 1)	2	(Op1, Ma1)	(20, A, 1)
4	(Op4, Ma3)	(40, C, 2)	1	(Op2, Ma2)	(10, A, 1)
3	(Op6, Ma4)	(20, C, 2)	3	(Op6, Ma4)	(30, A, 1)
4	(Op4, Ma3)	(40, C, 3)	2	(Op1, Ma1)	(20, A, 2)
3	(Op6, Ma4)	(20, C, 3)	1	(Op2, Ma2)	(10, A, 2)
1	(Op2, Ma2)	(10, B, 1)	3	(Op6, Ma4)	(30, A, 2)
2	(Op1, Ma1)	(20, B, 1)	4	(Op4, Ma3)	(40, C, 1)
4	(Op4, Ma3)	(40, B, 1)	2	(Op1, Ma1)	(20, C, 1)
3	(Op6, Ma4)	(30, B, 1)	4	(Op4, Ma3)	(40, C, 2)
5	(Op4, Ma5)	(50, B, 1)	3	(Op6, Ma4)	(20, C, 2)
3	(Op6, Ma4)	(30, A, 1)	4	(Op4, Ma3)	(40, C, 3)
3	(Op6, Ma4)	(30, A, 2)	3	(Op6, Ma4)	(20, C, 3)

Fig. 11. Chromosomes of Two-point crossover-1.

First Child			Second Child		
Workstation No.	(Operator, Machine)	(Task, Model, Box)	Workstation No.	(Operator, Machine)	(Task, Model, Box)
4	(Op4, Ma3)	(40, C, 2)	2	(Op1, Ma1)	(20, A, 1)
3	(Op6, Ma4)	(20, C, 2)	1	(Op2, Ma2)	(10, A, 1)
4	(Op4, Ma3)	(40, C, 3)	3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(20, C, 3)	2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, B, 1)	4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, A, 1)	2	(Op1, Ma1)	(20, C, 1)
1	(Op2, Ma2)	(10, A, 1)	4	(Op4, Ma3)	(40, C, 2)
3	(Op6, Ma4)	(30, A, 1)	3	(Op6, Ma4)	(20, C, 2)
2	(Op1, Ma1)	(20, A, 2)	4	(Op4, Ma3)	(40, C, 3)
1	(Op2, Ma2)	(10, A, 2)	3	(Op6, Ma4)	(20, C, 3)
3	(Op6, Ma4)	(30, A, 2)	1	(Op2, Ma2)	(10, B, 1)
4	(Op4, Ma3)	(40, C, 1)	2	(Op1, Ma1)	(20, B, 1)
2	(Op1, Ma1)	(20, C, 1)	4	(Op4, Ma3)	(40, B, 1)
2	(Op1, Ma1)	(20, B, 1)	3	(Op6, Ma4)	(30, B, 1)
4	(Op4, Ma3)	(40, B, 1)	5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, B, 1)	1	(Op2, Ma2)	(10, A, 2)
5	(Op4, Ma5)	(50, B, 1)	3	(Op6, Ma4)	(30, A, 2)

Cells with dots denote the genes transferred due to the precedence preserving rule. They are from the parent that gives that child the genes which are between two coloured points.

Fig. 12. Offsprings of Two-point crossover-1.

- Transference of the selected genes to the beginning of the picked chromosome.
- Observation of the sequencing diagram of the selected genes. Consider any genes that should be assigned before them.

Evaluation function and GA conditions: Evaluation function and GA conditions: The evaluation function determines the quality of a candidate solution. Therefore, the fitness evaluation is based on the objective function, which is the minimisation of the maximum completion time (makespan or Cmax). The population size for large instances is 70, and for small sizes, it is 28 chromosomes. Percentages of crossover, mutation, and iteration runs of GA-MSeq are tuned based on TMet, which is explained in Section 6. Fig. 14 depicts the general rules of GA-MSeq.

6. Computational results

6.1. Tests design

Computational tests are run on some small-size instances and real instances. The real instances from the company are of large dimension. But since they are associated with the smaller and larger stitching systems, mentioned in Section 4, they are divided into medium-size and large-size data, for comparison purposes. The real instances used are in <https://drive.inescotec.pt/s/rZEN9nLKfRcemLX> to provide the reader with a more favourable view of the data. The work in process is also considered in the data. They are transformed into new models and, therefore, increasing the number of available models.

Chromosome			Chromosome - After Mutation		
Workstation No.	(Operator, Machine)	(Task, Model, Box)	Workstation No.	(Operator, Machine)	(Task, Model, Box)
2	(Op1, Ma1)	(20, A, 1)	2	(Op1, Ma1)	(20, A, 1)
2	(Op1, Ma1)	(20, A, 2)	2	(Op1, Ma1)	(20, A, 2)
1	(Op2, Ma2)	(10, A, 1)	1	(Op2, Ma2)	(10, A, 1)
1	(Op2, Ma2)	(10, A, 2)	1	(Op2, Ma2)	(10, A, 2)
4	(Op4, Ma3)	(40, C, 1)	4	(Op4, Ma3)	(40, C, 1)
2	(Op1, Ma1)	(20, C, 1)	1	(Op2, Ma2)	(10, B, 1)
4	(Op4, Ma3)	(40, C, 2)	2	(Op1, Ma1)	(20, B, 1)
3	(Op6, Ma4)	(20, C, 2)	4	(Op4, Ma3)	(40, C, 3)
4	(Op4, Ma3)	(40, C, 3)	3	(Op6, Ma4)	(20, C, 3)
3	(Op6, Ma4)	(20, C, 3)	4	(Op4, Ma3)	(40, C, 2)
1	(Op2, Ma2)	(10, B, 1)	3	(Op6, Ma4)	(20, C, 2)
2	(Op1, Ma1)	(20, B, 1)	2	(Op1, Ma1)	(20, C, 1)
4	(Op4, Ma3)	(40, B, 1)	4	(Op4, Ma3)	(40, B, 1)
3	(Op6, Ma4)	(30, B, 1)	3	(Op6, Ma4)	(30, B, 1)
5	(Op4, Ma5)	(50, B, 1)	5	(Op4, Ma5)	(50, B, 1)
3	(Op6, Ma4)	(30, A, 1)	3	(Op6, Ma4)	(30, A, 1)
3	(Op6, Ma4)	(30, A, 2)	3	(Op6, Ma4)	(30, A, 2)

Cells with dots are the genes that are inverted based on the given rule.

Fig. 13. Mutation inversion.

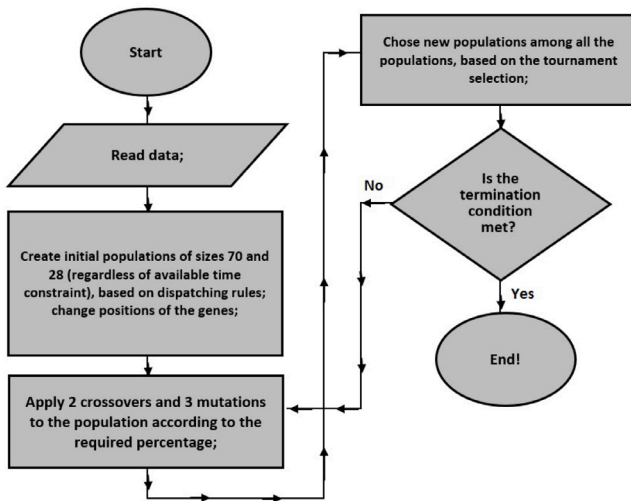


Fig. 14. General rules of GA-MSeq solution method.

The given optimisation model of this paper on the small-size instances reaches the optimal solution by using the Solver IBM ILOG CPLEX optimisation Studio, version 12.8. The computational tests were run on an Intel (R) Core (TM) i5-5200U CPU with 8 Gbs of Random Access Memory. All the defined methods are implemented in C++ and compiled with Microsoft Visual Studio 2008, running until the termination conditions are achieved, and according to the mentioned rules. The solution methods developed to solve these problems are also compared with each other and with reality.

The way to calculate some indicators, which are used in the next section, is explained as follows.

1. The *Gap_Seq* is the gap of the optimal result to the given methods which is calculated according to:

$$Gap_Seq = \left(\frac{Improvement_{obj} - Optimal_{obj}}{Optimal_{obj}} \right) \times 100$$

Here, *Improvement_obj* is the objective value of the final solution, achieved by using each developed method (VND-MSeq and GA-MSeq). *Optimal_obj* is the optimal solution determined by the optimisation model.

2. The *Improvement_Seq*, is related to the initial and final solutions of each developed method (VND-MSeq and GA-MSeq). It is calculated as follows:

$$Improvement_Seq = \left(\frac{Improvement_{obj} - Initial_{obj}}{Initial_{obj}} \right) \times 100$$

Here, *Improvement_obj* is the same as mentioned above, and the *Initial_obj* is the initial solution determined by each method.

3. The *Gain_Seq*, is associated with the difference between the available time in the company (*Av*), which is a one-day production plan, and the objective of each proposed method. This calculation is according to the next expression:

$$Gain_Seq = \left(\frac{Av - Improvement_{obj}}{Av} \right) \times 100$$

6.2. Parameter tuning

As stated by Talbi [54], parameter tuning may allow great flexibility and robustness, but needs careful initialisation. In this work, there are some parameters for both given methods, which should be tuned; therefore, a series of experiments are accomplished, and suitable data are analysed with statistical techniques Fallahi et al. [55]. The parameters can take discrete values, named as levels. Each level has a known value, otherwise if it does not have any known, then some benefit from the literature is considered Boyabatli and Sabuncuoglu [56]. TMet is employed to work with the combination of the levels of the investigated parameters.

Two types of parameters are considered for VND-MSeq method: the number of searched neighbours in 3 neighbourhood structures (Ns1, Ns2, and Ns5) and the number of iteration runs of VND-MSeq. The parameters related to the searched size are defined based on the other neighbourhood sizes (Ns3 and Ns4), which were all neighbours of any incumbent solution. In the other parameter, which is the iteration runs of VND-MSeq, the running time limits and the literature values are considered. Therefore, for VND-MSeq, there are 4 parameters with 2

Table 7
Different parameters and levels for VND-MSeq.

		No.of Ns1 searched solutions (P1)	No.of Ns2 searched solutions (P2)	No.of Ns5 searched solutions (P3)	No.of run iterations(P4)
Small-size	Level 1	5	5	5	20
	Level 2	10	10	10	30
Medium-size	Level 1	45	45	45	40
	Level 2	90	90	90	50
Large-size	Level 1	75	75	75	40
	Level 2	150	150	150	50

Table 8
Different parameters and levels for GA-MSeq.

		Percentage of two-point crossover-1 (P1)	Percentage of two-point crossover-2 (P2)	Percentage of mutation inversion (P3)	Percentage of mutation insertion (P4)	Percentage of mutation insertion-critical genes (P5)	No.of run iterations (P6)
Small-size	Level 1	0.2	0.3	0.1	0.05	0.1	20
	Level 2	0.4	0.4	0.15	0.1	0.15	30
	Level 3	0.5	–	–	–	–	–
Medium-size/ Large-size	Level 1	0.2	0.3	0.1	0.05	0.1	40
	Level 2	0.4	0.4	0.15	0.1	0.15	50
	Level 3	0.5	–	–	–	–	–

Table 9
Selected values of parameters affecting VND-MSeq and GA-MSeq.

	P1	P2	P3	P4	P5	P6
Small-size-VND-MSeq	Level 2 (Val = 10)	Level 2 (Val = 10)	Level 2 (Val = 10)	Level 1 (Val = 30)	–	–
Medium-size-VND-MSeq	Level 2 (Val = 90)	Level 1 (Val = 45)	Level 1 (Val = 45)	Level 2 (Val = 50)	–	–
Large-size-VND-MSeq	Level 1 (Val = 75)	Level 2 (Val = 150)	Level 2 (Val = 150)	Level 2 (Val = 50)	–	–
Small-size-GA-MSeq	Level 2 (Val = 0.4)	Level 1 (Val = 0.3)	Level 2 (Val = 0.15)	Level 2 (Val = 0.1)	Level 1 (Val = 0.1)	Level 2 (Val = 30)
Medium-size-GA-MSeq	Level 3 (Val = 0.5)	Level 2 (Val = 0.4)	Level 1 (Val = 0.1)	Level 1 (Val = 0.05)	Level 2 (Val = 0.15)	Level 2 (Val = 30)
Large-size-GA-MSeq	Level 3 (Val = 0.5)	Level 2 (Val = 0.4)	Level 1 (Val = 0.1)	Level 1 (Val = 0.05)	Level 2 (Val = 0.15)	Level 2 (Val = 30)

Table 10
Results of the sequencing problems for the small-size instances, using the optimisation model and VND-MSeq.

Instances	No.of models	No.of boxes	No.of tasks	No.of used operators	No.of used machines	Initial solution		Improved solution		Optimal solution		
						Objective (Cmax)	Time (s)	Objective (Cmax)	Time (s)	Objective (Cmax)	Time (s)	Gap_Seq
Small-size-1	2	3	3	2	2	200	0.06	200	0.28	200	0.05	0%
Small-size-2	3	5	4	3	3	290	0.06	270	0.66	270	0.06	0%
Small-size-3	3	8	4	4	4	458	0.10	422	1.30	412	0.12	2%
Small-size-4	4	6	5	3	3	396	0.12	360	1.38	340	0.26	6%
Small-size-5	4	13	5	4	4	746	0.36	650	27.03	614	16,390	6%
Small-size-6	6	34	6	4	4	2,570	1.95	2,400	100.04	2,400	73,426	0%

Table 11
Results of the sequencing problems for the small-size instances, using the optimisation model and GA-MSeq.

Instances	No.of models	No.of boxes	No. of tasks	No.of used operators	No.of used machines	Best initial population		Improved solution		Optimal solution		
						Objective (Cmax)	Time (s)	Objective (Cmax)	Time (s)	Objective (Cmax)	Time (s)	Gap_Seq
Small-size-1	2	3	3	2	2	200	0.17	200	0.51	200	0.05	0%
Small-size-2	3	5	4	3	3	270	0.32	270	1.44	270	0.06	0%
Small-size-3	3	8	4	4	4	428	0.50	418	3.17	412	0.12	1%
Small-size-4	4	6	5	3	3	352	0.40	342	2.87	340	0.26	1%
Small-size-5	4	13	5	4	4	658	2.37	614	63.59	614	16,390	0%
Small-size-6	6	34	6	4	4	2,400	10.32	2,400	102.41	2,400	73,426	0%

levels, shown in Table 7. Then by applying TMet and Minitab software 19, different tests are designed based on the Orthogonal arrays. An orthogonal array means that the parameter levels are weighted equally. According to this array, and considering a two-level design, 8 experiments are needed. These 8 experiments run on 2 instances from each category size (small, medium, and large), and hence, 48 runs are required.

For GA-MSeq we regard 6 parameters; 5 of them are the percentages of crossovers and mutations, and 1 is the number of GA-MSeq iteration runs. These percentages are defined so that the totals of minimum and maximum cases are equal to 60% and 90% for crossovers, 25% and 40% for mutations. These values are selected base on the literature and the running time limits. The number of iteration runs is the same as for VND-MSeq and Table 8 depicts these values. As the parameters have

Table 12

Real instance data and number of operators and machines used.

Instances	No.of models	No.of boxes	No.of tasks	No.of available operators	No.of available machines	No.of machine types	Max operator abilities	No. of used operators	No.of used machines	Used % of Av
Medium-size-1	15	127	15	28	41	15	11	10	10	80%
Medium-size-2	11	171	10	26	40	10	8	9	9	80%
Medium-size-3	16	159	15	28	41	14	12	11	11	80%
Medium-size-4	41	209	30	27	42	20	19	13	14	80%
Medium-size-5	35	94	47	17	31	5	39	11	12	80%
Large-size-1	303	334	70	86	146	15	36	69	71	60%
Large-size-2	320	356	62	86	152	16	31	70	73	60%
Large-size-3	298	335	68	86	148	14	36	67	70	60%
Large-size-4	326	377	73	86	143	15	36	73	77	60%
Large-size-5	344	399	69	86	148	14	36	70	71	60%
Large-size-6	185	354	85	86	141	14	41	82	88	60%
Large-size-7	196	401	70	86	141	14	33	74	80	60%
Large-size-8	157	283	77	86	144	14	42	78	84	60%
Large-size-9	148	291	93	86	143	14	45	78	81	60%

Table 13

Results of the sequencing problems for the small-size instances, using the optimisation model and GA-MSeq.

Instances	No.of models	No.of boxes	No. of tasks	No.of used operators	No.of used machines	Best initial population		Improved solution		Optimal solution		
						Objective (Cmax)	Time (s)	Objective (Cmax)	Time (s)	Objective (Cmax)	Time (s)	Gap_Seq
Small-size-1	2	3	3	2	2	200	0.17	200	0.51	200	0.05	0%
Small-size-2	3	5	4	3	3	270	0.32	270	1.44	270	0.06	0%
Small-size-3	3	8	4	4	4	428	0.50	418	3.17	412	0.12	1%
Small-size-4	4	6	5	3	3	352	0.40	342	2.87	340	0.26	1%
Small-size-5	4	13	5	4	4	658	2.37	614	63.59	614	16,390	0%
Small-size-6	6	34	6	4	4	2,400	10.32	2,400	102.41	2,400	73,426	0%

Table 14

Real instance data and number of operators and machines used.

Instances	No.of models	No.of boxes	No.of tasks	No.of available operators	No.of available machines	No.of machine types	Max operator abilities	No. of used operators	No.of used machines	Used % of Av
Medium-size-1	15	127	15	28	41	15	11	10	10	80%
Medium-size-2	11	171	10	26	40	10	8	9	9	80%
Medium-size-3	16	159	15	28	41	14	12	11	11	80%
Medium-size-4	41	209	30	27	42	20	19	13	14	80%
Medium-size-5	35	94	47	17	31	5	39	11	12	80%
Large-size-1	303	334	70	86	146	15	36	69	71	60%
Large-size-2	320	356	62	86	152	16	31	70	73	60%
Large-size-3	298	335	68	86	148	14	36	67	70	60%
Large-size-4	326	377	73	86	143	15	36	73	77	60%
Large-size-5	344	399	69	86	148	14	36	70	71	60%
Large-size-6	185	354	85	86	141	14	41	82	88	60%
Large-size-7	196	401	70	86	141	14	33	74	80	60%
Large-size-8	157	283	77	86	144	14	42	78	84	60%
Large-size-9	148	291	93	86	143	14	45	78	81	60%

distinct levels, and each level may have different values, TMet analysis is applied again. According to the orthogonal array and considering mixed-level design, 33 experiments are expected. With 33 runs for 1 instance of each category size (small, medium and large), 99 runs are required.

The appendix includes the results of the experiments based on TMet analysis. Table 9 represents the selected values of the parameters involved in VND-MSeq and GA-MSeq, based on TMet analysis.

6.3. Computational results

This section addresses the sequencing results. Firstly, and for the small-instances, the results of the optimisation model are reported and then compared with those achieved by the two developed methods (VND-MSeq and GA-MSeq). Secondly, the two approximate sequencing methods are applied to the real instances considering the values for the defined parameters. Then, these results are compared with existing results. The existing results in this work, are called *Company Reality*, which is also equal to Av. The input data for the sequencing, as

mentioned earlier, are the output data of the balancing. Meanwhile, a Lower Bound (LB) for the problem is calculated.

Table 10 features the results of the computational tests on the generated instances achieved by the optimisation model and VND-MSeq method.

VND-MSeq finds an optimal solution in three instances, and the three instances have a *Gap_Seq* equal to 2%, and 6%. As can be seen, instances number 5 and 6 are solved optimally but in hours, while VND-MSeq solved them quickly; particularly, instance 6 has a *Gap_Seq* equal to zero. Table 11 shows the results of the other method, GA-MSeq, for the small-size instances. GA-MSeq returns the optimal solution in four instances, and the other two instances have *Gap_Seq* equal to 1%. Therefore, the complete results in small-sized instances by GA-MSeq are better than by VND-MSeq. In summary, the average *Gap_Seq* for the small-size instances solved by VND-MSeq is 2%, and GA-MSeq is close to 0%. Moreover, both methods' initial solutions and populations are expected to be of good quality since they are not randomly generated but based on different dispatch rules. A test of the optimisation model's running times is performed, and the results show that they increase significantly as the number of boxes increases. It is also clear that the

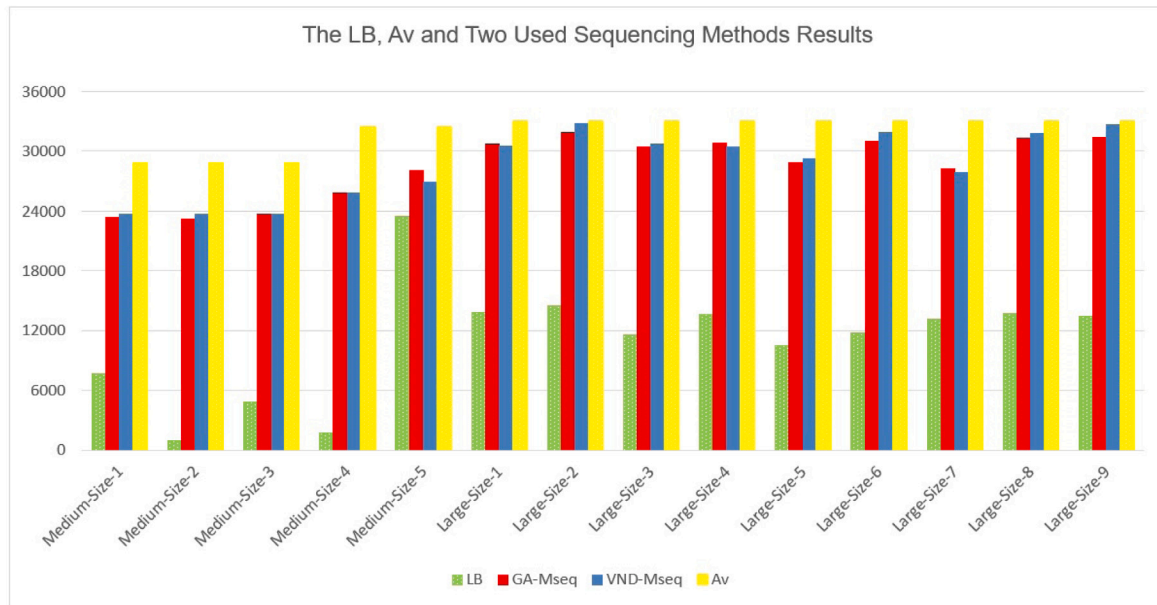


Fig. 15. Comparison of VND-MSeq and GA-MSeq results for real data problems.

Table 15

A comparison between VND-MSeq and GA-MSeq methods for the sequencing problems.

Method	Improvement_Seq			Gain_Seq		
	Min	Max	Average	Min	Max	Average
VND-MSeq	0%	20%	13%	1%	20%	11%
GA-MSeq	0%	12%	8%	3%	20%	11%

execution time of instance 6, without a significant dimension, is about 2 hours; therefore, using the optimisation model is not a good option for daily sequencing.

Two sequencing methods solve the medium and large size real instances. The results of both methods were then compared with *Company Reality*. The input data for the sequencing is taken from Sadeghi et al. [6]. In this work, the production plans could be met, for medium and large size problems, in 60% and 80% of the Av, respectively. The final information of the used instances is presented in Table 12.

Next, VND-MSeq and GA-MSeq are used for real instances. Their results are shown in Tables 13 and 14. Every indicator tabComResSeqOptVNDMSeq in this table are explained before. A comparison of both used methods is provided in Table 15.

Finally, according to the sequencing results, the following points can be listed:

1. The best objective value for the initial population of GA-MSeq is better than the objective value of the initial solution of VND-MSeq. In some instances, if it is not better, then they are equal to each other.
2. The time of gaining the initial solution of VND-MSeq is better than in GA-MSeq. It is because, in GA-MSeq, a population is created, but in VND-MSeq, only a solution is gained.
3. *Improvement_Seq* in VND-MSeq is better than in GA-MSeq.
4. In general, *Gain_Seq* in GA-MSeq is better than in VND-MSeq, which means that GA-MSeq produces, in general, better results. However, the difference between the final solutions is not significant.

5. The running times of both methods, for the small and medium-size instances, are less than 7 minutes.
6. VND-MSeq has less running times than GA-MSeq, for small and medium-size instances.
7. The running times of VND-MSeq, for large-size instances, are about 1 h and 30 min maximum, which is acceptable for practical purposes. However, in maximum, these values are 1 hour in GA-MSeq. Therefore, GA-MSeq has less running times than VND-MSeq, for the majority of large-size instances.
8. There is an instance that cannot be improved by both methods (instance of *Medium-Size-4*). The reason is that the Cmax on that instance is equal to the working time of one of the operators in the line.

Fig. 15 summarises the comparisons illustrated for the medium and large-size instances.

7. Conclusions

The industrial context of this work is a large footwear exporting company with new automated flexible stitching systems. The research work focused on modelling and solving the specific Mixed-model Assembly Line Sequencing Problems (MALSP).

The paper mostly contributed at two levels, in the scientific area of assembly line sequencing and the practical solution of complex industrial problems arising in two distinct automatic stitching systems. A new mixed-model sequencing problem was described, and an optimisation model was presented and tested. The real problems' size prevented the direct use of exact optimisation methods, so the authors devised two approximate methods, named VND-MSeq and GA-MSeq. The company invested in entirely new flexible assembly systems, whose operation needed to be optimised. Let us remark that the difficulties of sequencing the mixed-model tasks were treated somewhat manual, although they already depended on balancing solutions obtained previously. According to the industrial case description, the units moving in the lines are not pairs of shoes but boxes containing copious quantities of product components. Such boxes are automatically transported, in any direction, between workstations or from the warehouse to any workstation. In the end, the methods designed and implemented helped

the project team contribute to improving the company's procedures and overcoming those daily difficulties.

The optimisation model presented allowed understanding the problem better and solving small-size instances. Besides, the model can be adapted to other similar industrial cases. As mentioned, the first contribution to the resolution of the MALSP were two approximate methods, VND-MSeq, based on Variable Neighbourhood Descent, and GA-MSeq, based on Genetic Algorithms. Although following known metaheuristics, they were designed and adapted accordingly. Moreover, they include several dispatching rules to generate initial solutions and populations.

The methods were compared in various ways, in many instances, from generated to those based on real data. The average optimal gaps for both methods (*Gap_Seq*), relative to the small-size instances, are smaller than 3%. GA-MSeq could not find the optimal solutions in two examples, with the gaps 1%. The objective function values of GA-MSeq slightly exceed those obtained by VND-MSeq. VND-MSeq performs better on small and medium-size instances in terms of running times, but the opposite is true for large-size instances, where GA-MSeq performed considerably better. As for the practical cases, the instances are based precisely on the data available in the company, only modified to use the developed methods (we called them *Company Reality*).

In conclusion, the developed methods' results show a significant improvement compared to the company's known results. Additionally, the solution approaches achieve results faster than before and with less planning effort. Another relevant insight of the implementation is allowing the generation of the list of boxes for each workstation, avoiding the current situation of a list of recommended boxes to select later, whenever the workstation needs to work.

According to the team leader at Kyaia, the benefits are reducing stocks between operations and each order's production time due to better balance and sequencing. The work carried out achieves better results through the more comprehensive analysis of all models and resources simultaneously, one of the current difficulties. It was also observed that the operators' occupancy rate globally increased because of better attention to their qualifications. These promising indicators make the optimisation approach bring effective advantages to the company's overall efficiency.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020. Moreover, the first author is grateful for FCT for awarding her the Ph.D. Grant SFRH/BD/91550/2012.

Appendix

The Appendix covers the Taguchi Method (TMet) analysis results to select the best values for the parameters of VND-MSeq and GA-MSeq. Tables A.16 and A.17. present the experiments that should be carried out for both methods, using the *Minitab software* 19 software. According to TMet, both should run on the selected instances for the defined numbers 48 and 99. The results of these runs are in Tables A.18 and A.19. As mentioned in Section 6, the experiments of VND-MSeq were run on two instances; the values in Tables A.18 represent their average.

The signal-to-noise ratios (S/N ratios, which are a measure of robustness) for all instance sizes of VND-MSeq and GA-MSeq are shown in Figs. A.16 and A.17. In terms of the minimisation function of the problems, each parameter's selected value or level is the one with the smaller S/N ratio. Although the mean values of the S/N ratios are negative (see Figs. A.16 and A.17), the selection process only considers their absolute values. These selected values are used in both solution methods to run in different instances. Table 9 in Section 6 has the results.

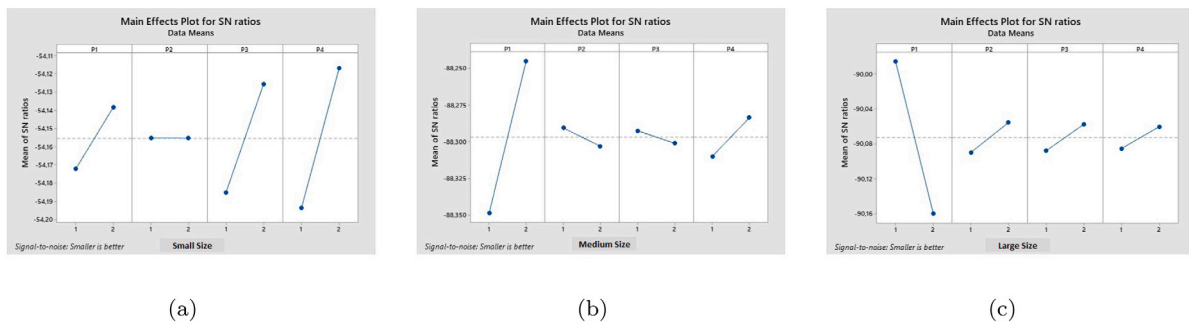


Fig. A.16. S/N ratios charts of TMet analysis related to VND-MSeq.

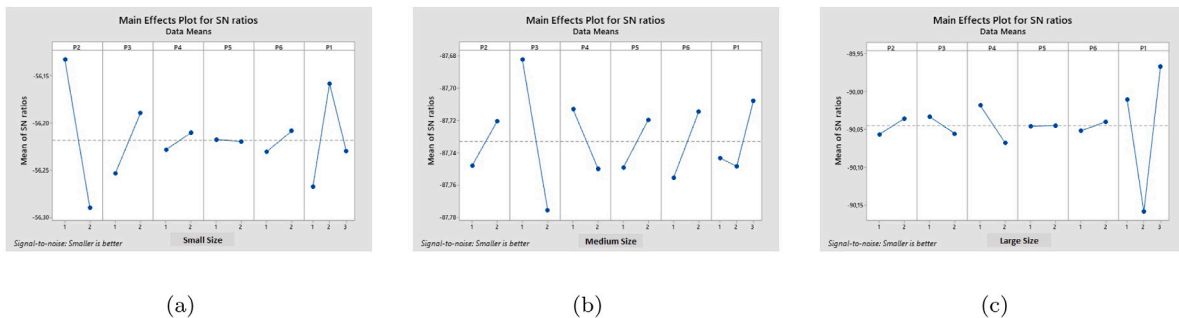


Fig. A.17. S/N ratios charts of TMet analysis related to GA-MSeq.

Table A.16

Recommended experiments for VND-MSeq by TMet in Minitab software.

	Test	P1	P2	P3	P4
Small-size	1	5	5	5	20
	2	5	5	10	30
	3	5	10	5	30
	4	5	10	10	20
	5	10	5	5	30
	6	10	5	10	20
	7	10	10	5	20
	8	10	10	10	30
Medium-size	1	45	45	45	40
	2	45	45	90	50
	3	45	90	45	50
	4	45	90	90	40
	5	90	45	45	50
	6	90	45	90	40
	7	90	90	45	40
	8	90	90	90	50
Large-size	1	75	75	75	40
	2	75	75	150	50
	3	75	150	75	50
	4	75	150	150	40
	5	150	75	75	50
	6	150	75	150	40
	7	150	150	75	40
	8	150	150	150	50

Table A.17

Recommended experiments for GA-MSeq by TMet in Minitab software.

	Test	P1	P2	P3	P4	P5	P6
Small-size	1	0.2	0.3	0.1	0.05	0.1	20
	2	0.4	0.3	0.1	0.05	0.1	20
	3	0.5	0.3	0.1	0.05	0.1	20
	4	0.2	0.3	0.1	0.1	0.15	30
	5	0.4	0.3	0.1	0.1	0.15	30
	6	0.5	0.3	0.1	0.1	0.15	30
	7	0.2	0.3	0.15	0.05	0.15	30
	8	0.4	0.3	0.15	0.05	0.15	30
	9	0.5	0.3	0.15	0.05	0.15	30
	10	0.2	0.3	0.15	0.1	0.1	30
	11	0.4	0.3	0.15	0.1	0.1	30
	12	0.5	0.3	0.15	0.1	0.1	30
	13	0.2	0.3	0.15	0.1	0.15	20
	14	0.4	0.3	0.15	0.1	0.15	20
	15	0.5	0.3	0.15	0.1	0.15	20
	16	0.2	0.4	0.1	0.1	0.15	20
	17	0.4	0.4	0.1	0.1	0.15	20
	18	0.5	0.4	0.1	0.1	0.15	20
	19	0.2	0.4	0.1	0.1	0.1	30
	20	0.4	0.4	0.1	0.1	0.1	30
	21	0.5	0.4	0.1	0.1	0.1	30
	22	0.2	0.4	0.1	0.05	0.15	30
	23	0.4	0.4	0.1	0.05	0.15	30
	24	0.5	0.4	0.1	0.05	0.15	30
	25	0.2	0.4	0.15	0.1	0.1	20
	26	0.4	0.4	0.15	0.1	0.1	20
	27	0.5	0.4	0.15	0.1	0.1	20
	28	0.2	0.4	0.15	0.05	0.15	20
	29	0.4	0.4	0.15	0.05	0.15	20
	30	0.5	0.4	0.15	0.05	0.15	20
	31	0.2	0.4	0.15	0.05	0.1	30
	32	0.4	0.4	0.15	0.05	0.1	30
	33	0.5	0.4	0.15	0.05	0.1	30

(continued on next page)

Table A.17 (continued).

	Test	P1	P2	P3	P4	P5	P6
Medium-size/ Large-size	1	0.2	0.3	0.1	0.05	0.1	40
	2	0.4	0.3	0.1	0.05	0.1	40
	3	0.5	0.3	0.1	0.05	0.1	40
	4	0.2	0.3	0.1	0.1	0.15	50
	5	0.4	0.3	0.1	0.1	0.15	50
	6	0.5	0.3	0.1	0.1	0.15	50
	7	0.2	0.3	0.15	0.05	0.15	50
	8	0.4	0.3	0.15	0.05	0.15	50
	9	0.5	0.3	0.15	0.05	0.15	50
	10	0.2	0.3	0.15	0.1	0.1	50
	11	0.4	0.3	0.15	0.1	0.1	50
	12	0.5	0.3	0.15	0.1	0.1	50
	13	0.2	0.3	0.15	0.1	0.15	40
	14	0.4	0.3	0.15	0.1	0.15	40
	15	0.5	0.3	0.15	0.1	0.15	40
	16	0.2	0.4	0.1	0.1	0.15	40
	17	0.4	0.4	0.1	0.1	0.15	40
	18	0.5	0.4	0.1	0.1	0.15	40
	19	0.2	0.4	0.1	0.1	0.1	50
	20	0.4	0.4	0.1	0.1	0.1	50
	21	0.5	0.4	0.1	0.1	0.1	50
	22	0.2	0.4	0.1	0.05	0.15	50
	23	0.4	0.4	0.1	0.05	0.15	50
	24	0.5	0.4	0.1	0.05	0.15	50
	25	0.2	0.4	0.15	0.1	0.1	40
	26	0.4	0.4	0.15	0.1	0.1	40
	27	0.5	0.4	0.15	0.1	0.1	40
	28	0.2	0.4	0.15	0.05	0.15	40
	29	0.4	0.4	0.15	0.05	0.15	40
	30	0.5	0.4	0.15	0.05	0.15	40
	31	0.2	0.4	0.15	0.05	0.1	50
	32	0.4	0.4	0.15	0.05	0.1	50
	33	0.5	0.4	0.15	0.05	0.1	50

Table A.18

Average results of experiments related to VND-MSeq.

	Test	Average results
Small-size	1	511
	2	518
	3	525
	4	525
	5	517
	6	501
	7	512
	8	518
Medium-size	1	26,037
	2	26,679
	3	26,128
	4	25,756
	5	25,312
	6	25,885
	7	26,454
	8	25,712
Large-size	1	32,139
	2	31,345
	3	31,568
	4	31,237
	5	32,016
	6	32,315
	7	32,060
	8	32,455

Table A.19
Results of experiments related to GA-MSeq.

	Test	Results	Test	Results	Test	Results	Test	Results
Small-size	1	638	10	650	19	658	28	638
	2	650	11	614	20	642	29	658
	3	642	12	656	21	658	30	658
	4	658	13	650	22	660	31	650
	5	642	14	632	23	658	32	624
	6	624	15	630	24	658	33	658
	7	660	16	642	25	654	–	–
	8	638	17	658	26	654	–	–
	9	628	18	658	27	658	–	–
Medium-size	1	24,051	10	24,651	19	24,617	28	24,651
	2	24,651	11	24,651	20	24,120	29	24,651
	3	24137	12	24,651	21	24,137	30	24,651
	4	24,651	13	24391	22	24,051	31	24,651
	5	24,156	14	24,651	23	24,111	32	24,051
	6	24,137	15	24,111	24	23651	33	24,114
	7	24,471	16	24,214	25	23,871	–	–
	8	24,410	17	24,111	26	24,871	–	–
	9	24,257	18	24,471	27	24,871	–	–
Large-size	1	31,560	10	31,742	19	32,406	28	31,988
	2	31,832	11	31,256	20	31,978	29	32,090
	3	31,588	12	32,539	21	31,036	30	31,186
	4	32,284	13	32,326	22	31,160	31	31,470
	5	31,670	14	33,338	23	32,646	32	33,068
	6	31,844	15	30,750	24	30,746	33	31,276
	7	31,422	16	31,059	25	30,874	–	–
	8	31,610	17	32,050	26	32,788	–	–
	9	31,746	18	32,340	27	31,510	–	–

References

- [1] Scholl A, Becker C. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European J Oper Res* 2006;168(3):666–93. <http://dx.doi.org/10.1016/j.ejor.2004.07.022>.
- [2] Becker C, Scholl A. A survey on problems and methods in generalized assembly line balancing. *European J Oper Res* 2006;168(3):694–715.
- [3] Sivasankaran P, Shahabudeen P. Literature review of assembly line balancing problems. *Int J Adv Manuf Technol* 2014;73(9–12):1665–94.
- [4] Chen Y-Y, Cheng C-Y, Wang L-C, Chen T-L. A hybrid approach based on the variable neighborhood search and particle swarm optimization for parallel machine scheduling problems—a case study for solar cell industry. *Int J Prod Econ* 2013;141(1):66–78.
- [5] Oksuz MK, Buyukozkan K, Satoglu SI. U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics. *Comput Ind Eng* 2017;112:246–63.
- [6] Sadeghi P, Rebelo RD, Ferreira JS. Balancing mixed-model assembly systems in the footwear industry with a variable neighbourhood descent method. *Comput Ind Eng* 2018;121:161–76.
- [7] Xiaobo Z, Ohno K. A sequencing problem for a mixed-model assembly line in a JIT production system. *Comput Ind Eng* 1994;27(1–4):71–4.
- [8] Lopes TC, Michels AS, Sikora CGS, Magatão L. Balancing and cyclical scheduling of asynchronous mixed-model assembly lines with parallel stations. *J Manuf Syst* 2019;50:193–200.
- [9] Sadeghi P, Rebelo RD, Ferreira JS. Balancing a mixed-model assembly system in the footwear industry. In: IFIP international conference on advances in production management systems. Springer; 2017, p. 527–35.
- [10] Dar-El EM, Cothier R. Assembly line sequencing for model mix. *Int J Prod Res* 1975;13(5):463–77.
- [11] Graham RL, Lawler EL, Lenstra JK, Kan AR. Optimization and approximation in deterministic sequencing and scheduling: a survey. 5, Elsevier; 1979, p. 287–326.
- [12] Miltenburg J. Level schedules for mixed-model assembly lines in just-in-time production systems. *Manage Sci* 1989;35(2):192–207.
- [13] Tavakkoli-Moghaddam R, Rahimi-Vahed A. Multi-criteria sequencing problem for a mixed-model assembly line in a JIT production system. *Appl Math Comput* 2006;181(2):1471–81.
- [14] Boysen N, Flidner M, Scholl A. Sequencing mixed-model assembly lines: Survey, classification and model critique. *European J Oper Res* 2009;192(2):349–73.
- [15] Bautista J, Alfaro R, Batalla C. Modeling and solving the mixed-model sequencing problem to improve productivity. *Int J Prod Econ* 2015;161:83–95.
- [16] Miltenburg J, Steiner G, Yeomans S. A dynamic programming algorithm for scheduling mixed-model, just-in-time production systems. *Math Comput Modell* 1990;13(3):57–66.
- [17] Ng W, Mak K. A branch and bound algorithm for scheduling just-in-time mixed-model assembly lines. *Int J Prod Econ* 1994;33(1–3):169–83.
- [18] Bolat A, Savsar M, Al-Fawzan MA. Algorithms for real-time scheduling of jobs on mixed model assembly line. *Comput Oper Res* 1994;21(5):487–98.
- [19] Bard J, Shtub A, Joshi S. Sequencing mixed-model assembly lines to level parts usage and minimize line length. *Int J Prod Res* 1994;32(10):2431–54.
- [20] Pinedo M, Chao X. Operations scheduling with applications in manufacturing and services. McGraw Hill; 1999.
- [21] Xanthopoulos A, Koulouriotis D, Gasteratos A, Ioannidis S. Efficient priority rules for dynamic sequencing with sequence-dependent setups. *Int J Ind Eng Comput* 2016;7(3):367–84.
- [22] Kasemset C, Pintaruean E. Application of job sequencing policies in refrigerated truck sequencing: A case study. In: 2017 international conference on industrial engineering, management science and application (ICIMSA). IEEE; 2017, p. 1–5.
- [23] Prandtstetter M, Raidl G. A variable neighborhood search approach for solving the car sequencing problem. *na*; 2005.
- [24] Yazdani M, Amiri M, Zandieh M. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Syst Appl* 2010;37(1):678–87.
- [25] Driessl R, Mönch L. Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints and ready times. *Comput Ind Eng* 2011;61(2):336–45.
- [26] Zhang G, Zhang L, Song X, Wang Y, Zhou C. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem. *Cluster Comput* 2019;22(5):11561–72.
- [27] Adamo T, Ghiani G, Guerriero E, Manni E. Automatic instantiation of a variable neighborhood descent from a mixed integer programming model. *Oper Res Perspect* 2017;4:123–35.
- [28] Bierwirth C, Mattfeld DC. Production scheduling and rescheduling with genetic algorithms. *Evol Comput* 1999;7(1):1–17.
- [29] Cochran JK, Horng S-M, Fowler JW. A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. *Comput Oper Res* 2003;30(7):1087–102.
- [30] Mansouri SA. A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines. *European J Oper Res* 2005;167(3):696–716.
- [31] Lei D. Multi-objective production scheduling: a survey. *Int J Adv Manuf Technol* 2009;43(9–10):926.
- [32] Rebai M, Kacem I, Adjallah KH. Earliness–tardiness minimization on a single machine to schedule preventive maintenance tasks: metaheuristic and exact methods. *J Intell Manuf* 2012;23(4):1207–24.
- [33] Huang J, Süer GA, Urs SB. Genetic algorithm for rotary machine scheduling with dependent processing times. *J Intell Manuf* 2012;23(5):1931–48.
- [34] Moradi H, Zandieh M. An imperialist competitive algorithm for a mixed-model assembly line sequencing problem. *J Manuf Syst* 2013;32(1):46–54.
- [35] Gen M, Lin L. Multiobjective evolutionary algorithm for manufacturing scheduling problems: state-of-the-art survey. *J Intell Manuf* 2014;25(5):849–66.
- [36] Rabbani M, Aliabadi L, Farrokhi-Asl H. A multi-objective mixed model two-sided assembly line sequencing problem in a make-to-order environment with customer order prioritization. *J Optim Ind Eng* 2019;12(2):1–20.

- [37] Zhang B, Xu L, Zhang J. A multi-objective cellular genetic algorithm for energy-oriented balancing and sequencing problem of mixed-model assembly line. *J Cleaner Prod* 2020;244:118845.
- [38] Costa MT, Ferreira JS. A simulation analysis of sequencing rules in a flexible flowline. *European J Oper Res* 1999;119(2):440–50.
- [39] Zangiacomi A, Zhijian L, Sacco M, Boër CR. Process planning and scheduling for mass customised shoe manufacturing. *Int J Comput Integr Manuf* 2004;17(7):613–21.
- [40] Süer GA, Subramanian A, Huang J. Heuristic procedures and mathematical models for cell loading and scheduling in a shoe manufacturing company. *Comput Ind Eng* 2009;56(2):462–75.
- [41] Chen JC, Wirasambada S, Pujawan IN, Wang K, Wu C, Lin C, et al. Minimizing total tardiness and makespan for hybrid flow shop problem with resource constraint in footwear manufacturing. In: *Asia pacific industrial engineering & management systems conference*; 2012. p. 1093–9.
- [42] Guimarães L, Ribeiro J, Renner J, Oliveira PD. Worker evaluation of a macroergonomic intervention in a Brazilian footwear company. *Applied Ergon* 2014;45(4):923–35.
- [43] Chen JC, Putra AP, Anggono N, Chen J, Su Y-S. Simulation modeling and analysis for stitching line of footwear industry. In: *International conference on industrial engineering and operations management*; 2014. p. 1099–106.
- [44] Chen JC, Wu C-W, Thao TD, Su L-H, Hsieh W-H, Chen T. Hybrid genetic algorithm for solving assembly line balancing problem in footwear industry. *Adv Mater Res* 2014;(939).
- [45] Ulutas B, Islier AA. Dynamic facility layout problem in footwear industry. *J Manuf Syst* 2015;36:55–61.
- [46] Dang Q-V, Pham K. Design of a footwear assembly line using simulation-based ALNS. *Proc CIRP* 2016;40:596–601.
- [47] Lopes IC, Lopes SO, Pereira RM, Teixeira S, Vaz AIF. A scheduling application to a molding injection machine: a challenge addressed on the 109th European Study Group with Industry. In: *European consortium for mathematics in industry*. Springer; 2016. p. 525–32.
- [48] Quyen NTP, Chen JC, Yang C-L. Hybrid genetic algorithm to solve resource constrained assembly line balancing problem in footwear manufacturing. *Soft Comput* 2017;21(21):6279–95.
- [49] Fani V, Bindi B, Bandinelli R. Balancing assembly line in the footwear industry using simulation: A case study. In: *ECMS*; 2020. p. 56–62.
- [50] Roshanaei V, Naderi B, Jolai F, Khalili M. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan. *Future Gener Comput Syst* 2009;25(6):654–61.
- [51] Amiri M, Zandieh M, Yazdani M, Bagheri A. A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. *Int J Prod Res* 2010;48(19):5671–89.
- [52] Kucukkoc I, Zhang DZ. A mathematical model and genetic algorithm-based approach for parallel two-sided assembly line balancing problem. *Prod Plan Control* 2015;26(11):874–94.
- [53] Yu C, Semeraro Q, Matta A. A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility. *Comput Oper Res* 2018;100:211–29.
- [54] Talbi EG. *Metaheuristics: From design to implementation*, vol. 74. John Wiley & Sons; 2009.
- [55] Fallahi M, Amiri S, Yaghini M. A parameter tuning methodology for metaheuristics based on design of experiments. *Int J Eng Technol Sci* 2014;2(6):497–521.
- [56] Boyabatli O, Sabuncuoglu I. Parameter selection in genetic algorithms. *J Syst Cybern Inf* 2004;4(2):78.