Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Solving mixed-model two-sided u-type assembly line balancing problem using a hybrid GA-VNS

Yılmaz Delice[a*]

[a]*Department of Industrial Engineering, Faculty of Engineering, Architecture and Design, Kayseri University, 38280 Talas, Kayseri, Turkey*

C H R O N I C L E

Article history:
Received January10 2026
Received in Revised Format March 12 2026
Accepted April 16 2026
Available online April 16 2026

Keywords:
Two-sided U-type assembly lines
Mixed-model
Genetic Algorithm
Variable Neighborhood Search

A B S T R A C T

This study investigates the mixed-model two-sided U-type assembly line balancing (MMTsUtALB) problem and proposes a hybrid solution approach based on Genetic Algorithm (GA) and Variable Neighborhood Search (VNS). Unlike existing studies on two-sided U-type assembly line balancing (TsUtALB), the mixed-model structure is explicitly considered. In the proposed approach, GA is used to explore the solution space through evolutionary operators, while VNS is applied as a local improvement procedure to refine promising solutions and improve convergence. A problem-oriented priority rule–based encoding method is adopted to represent solutions, which are transformed into feasible two-sided U-type assembly line configurations using a decoding-based task assignment procedure. This structure allows the algorithm to balance diversification and intensification during the search process. The effectiveness of the GA–VNS hybrid algorithm is evaluated using benchmark test problems. Since this study represents the first attempt to solve the MMTsUtALB problem, the obtained results are compared with closely related mixed-model two-sided assembly line balancing studies. Computational results show that GA–VNS achieves competitive performance, particularly for larger instances, by reducing the number of stations and positions with lower computational times than SA and PSO.

© 2026 by the authors; licensee Growing Science, Canada

1. Introduction

Assembly lines (ALs) are widely used in manufacturing environments where high quantities of similar products must be produced in a consistent manner. Their popularity largely stems from the ability to achieve stable quality levels while keeping production costs under control. In such systems, work is distributed among a series of stations, and each station performs a set of assigned tasks one after another, without exceeding a specific cycle time. The assignment of tasks to stations along the assembly line must be carried out in accordance with priority relationships and the various technological constraints imposed by the production system. The resulting coordination of these requirements is often known as assembly line balancing (ALB). ALB aims to achieve a functional distribution of tasks that ensures the line operates stably and efficiently.

The importance of this problem has made it a constantly addressed topic in production and operations management studies. While the initial studies by Salveson (1955) and Baybars (1986) provided an understanding of the problem, later research offered alternative contributions that better represented real-world situations. Due to the combinatorial nature of the problems arising from the increasing scope of proposed models to meet real-life needs, Assembly Line Balancing (ALB) is classified among NP-hard problems (Gutjahr & Nemhauser, 1964). Generally, exact solution methods are insufficient for solving large-scale problems. In recent years, a wide variety of ALB variants have been discussed in the literature, considering different line configurations, mixed model generation, stochastic task times, and walking and setup times. Extensive surveys covering various formulations and solution approaches of the ALB problem have been conducted by Rashid, et al. (2012), Sivasankaran and Shahabudeen (2014), Abdullah Make et al. (2017), Li et al. (2017), Razali et al. (2019), Chutima (2020), Eghtesadifard et al. (2020), Jiao et al. (2021), Boysen et al. (2021) and Çimen et al. (2022).

Traditionally, ALs operate on a single side. However, in industries producing large-sized products such as automobiles, buses, and heavy machinery, operators frequently work on both sides of the product simultaneously. These systems are referred to as two-sided ALs, where tasks may be restricted to the left or right side of the line or be executable on either side depending on technological requirements. Early work on two-sided ALs can be traced back to Bartholdi (1993), who provided one of the first formal descriptions of this type of production system. Subsequent studies focused on the practical difficulties that

* Corresponding author
E-mail ydelice@kayseri.edu.tr (Y. Delice)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2026 Growing Science Ltd.
doi: 10.5267/j.ijiec.2026.4.006

arise when tasks must be assigned to specific sides of the line and when synchronization between opposite stations is required (Kim et al., 2000). Because two-sided lines are frequently used in the assembly of large products, such as those found in the automotive sector, they have remained an active research topic for many years. At the same time, their combinatorial structure has made exact optimization difficult, which explains why a broad range of solution strategies has been explored in the literature. These include both classical optimization approaches and a variety of heuristic and metaheuristic techniques–based methods (Baykasoğlu & Dereli, 2008; Kim et al., 2009; Özcan, 2010; Khorasanian et al., 2013; Li et al., 2016; Kucukkoc & Zhang, 2016; Delice et al., 2017a; Kucukkoc et al., 2018; Yang et al., 2019).

Another alternative layout to traditional straight ALs is an arrangement where the assembly line is placed in a U-shape. In a U-type arrangement, stations are positioned so that operators may work on tasks located near both ends of the line, which changes the way labor and material flow are organized. Previous studies have shown that this structure is advantageous in terms of flexibility, operational visibility, and shorter material handling (Miltenburg & Wijngaard, 1994; Scholl & Klein, 1999). These lines have been shown to utilize labor more efficiently compared to straight-line layouts (Sparling & Miltenburg, 1998).

Another factor affecting assembly line balancing is the increase in product variety. On lines suitable for single-model production, identical products are produced efficiently in high volumes. However, they are insufficient when customer requirements vary significantly. Therefore, mixed-model ALs have been developed to allow the production of multiple product types on the same line. Although this increases flexibility, it also complicates the balancing problem, since task times and precedence relations may depend on the selected model (Kucukkoc & Zhang, 2017). To cope with this added complexity, many studies have relied on metaheuristic solution approaches, including GA, simulated annealing, ant colony optimization, and particle swarm optimization (Simaria & Vilarinho, 2009; Hamzadayi & Yildiz, 2013; Kucukkoc & Zhang, 2014; Zhang et al., 2018).

Despite the breadth of existing research, most studies have examined only limited combinations of these characteristics. For example, mixed-model U-type lines and mixed-model two-sided lines have been studied separately, while production systems that integrate all of these features remain relatively underexplored. Although TsUtALB problems have been addressed in the literature, existing contributions are largely restricted to single-model production settings (Thomopoulos, 1970; Macaskill, 1972).

The TsUtALB problem itself represents a relatively recent extension of classical ALB, combining two-sided task execution with a U-shaped layout. One of the earliest physical configurations of such a system was presented by Yegül, Ağpak, and Yavuz (2010), who considered a U-type line in which only one branch operates as a two-sided line and proposed a multi-pass random assignment procedure for its solution. Ağpak, Yegül, and Gökçen (2012) further modified this configuration and developed a bi-objective integer programming model, demonstrating that two-sided U-type layouts can reduce both line length and the number of workstations. Subsequent studies introduced metaheuristic solution approaches under various assumptions. Delice et al. (2016) incorporated stochastic processing times and solved large-scale instances using a GA. Delice et al. (2017a) proposed a particle swarm optimization approach, while Delice et al. (2018) considered sequence-dependent setup times and applied an ant colony optimization algorithm. More recently, Jiao et al. (2025) considered heuristic factors and pheromone information together in the ant colony optimization algorithm they developed.

Similarly, while mixed-model ALB has been examined in either two-sided or U-Type configurations, the joint consideration of mixed-model production, two-sided operation, and U-type line layout has not yet been explored. The integration of these three characteristics substantially increases problem complexity due to model-dependent task times, side-dependent task restrictions, and complex task assignment decisions, which cannot be adequately handled by existing solution approaches. Therefore, a clear research gap exists in both the formulation and solution of the MMTsUtALB problem. To address the identified research gap, this study makes the following contributions. First, the MMTsUtALB problem is formally defined and investigated for the first time, providing a comprehensive problem setting that captures realistic manufacturing conditions. Secondly, a hybrid metaheuristic based on GA and VNS is proposed, combining the global exploration capability of GA with the local condensation power of VNS, effectively addressing the complexity of the problem. Third, a problem-specific priority rule–based encoding scheme and a decoding-based task assignment procedure are developed to generate feasible two-sided U-type line configurations under mixed-model production conditions.

The remainder of this paper is organized as follows. Section 2 presents the definition of the MMTsUtALB problem. Section 3 describes the proposed GA-VNS algorithm in detail. Section 4 provides illustrative examples, and Section 5 discusses the computational results. Section 6 concludes the paper and outlines directions for future research.

2. Mixed-Model Two-Sided U-Type Assembly Line Balancing Problem

2.1. Problem Definition

In MMTsUtALB, the ALB problem is defined by the simultaneous consideration of mixed-model production, two-sided operations, and a U-Type line layout, reflecting the structural and operational requirements of modern assembly systems. The combination of these features significantly increases problem complexity compared to traditional ALB problems.

2.1.1 Assembly Line Structure

This study examines an assembly line structure that combines the characteristics of a U-shaped layout with two-sided operations. One arm of the U shape is called the entrance arm, while the other is called the exit arm. The resulting structure enables the simultaneous execution of tasks on both sides of the product. Such a configuration is well suited to the assembly of large products, including buses and trucks. Within this layout, stations are assigned to four distinct locations, with each location arranged along the production track. The resulting structure can be represented as a four-row layout scheme, in which rows correspond to locations and columns indicate positions (Delice et al., 2017a).

This arrangement increases flexibility in station design, as tasks performed on opposite sides of the product can be allocated to stations facing each other along the two arms of the U-shaped line. Tasks assigned to facing locations within the same position are processed simultaneously on the same product unit. Such facing stations are referred to as mated stations. Stations facing each other at the center of the U-shape may, depending on the line balancing process, be separate (distinct stations) in some cases, or be combined into a single crossover station in others (Delice et al., 2017a). However, depending on precedence constraints and task-side restrictions, some locations within a position may remain idle (Bartholdi, 1993; Kim et al., 2000).

2.1.2 Task Characteristics and Precedence Relations

Each assembly task is associated with a deterministic processing time and may require execution on a specific side of the product (left or right) or may be flexible and performed on either side. Task precedence relationships are represented by a precedence diagram, which determines the feasible task assignment sequences.

In the two-sided U-type structure, different task assignment logics apply to the two arms of the line. On the entrance arm, assignable tasks are those whose predecessors have already been completed, whereas on the exit arm, tasks whose successors have been assigned become eligible. Due to the U-Type configuration, crossover positions may occur at the center of the line, where tasks from both arms are assigned to the same station, as described by Delice et al. (2017a). Furthermore, in the proposed assembly line, the tasks that can be assigned to the central stations may differ depending on whether the flow starts from the entrance or exit arm of the U-line, leading to alternative solution configurations. Therefore, both cases are considered in the present study.

2.1.3 Mixed-Model Production Environment

Unlike single-model ALs, the MMTsUtALB problem considers the simultaneous production of multiple product models with similar structures but model-dependent task times. All models follow the same precedence relationships; however, task times vary across models. In this variation, line balancing becomes significantly more difficult because a task assignment that satisfies the cycle time constraint for one model may not satisfy it for another. In the hybrid model, idle time generation is largely unavoidable.

This is due not only to priority constraints (Delice et al., 2017b) as in single-model two-sided U-shaped lines, but also to differences in task times between models. Furthermore, as the number of product models increases, synchronizing the task execution work of paired and crossover stations becomes even more difficult. This further increases waiting times between positions.

2.1.4 Problem Objective

The objective of the MMTsUtALB problem is to assign the tasks of all models to stations along the two-sided U-type assembly line in such a way that cycle time limitations, task-side restrictions, and precedence constraints are satisfied, while achieving an efficient balance of the line. In this context, the objective function given in Eq. (1) is employed to primarily minimize the total number of stations (X) and secondarily minimize the number of positions (Y). The weighting coefficients are selected to enforce a lexicographic preference, assigning a significantly higher priority to minimizing X. Due to the combined effects of mixed-model production, two-sided operation, and U-type layout, the problem is NP-hard, rendering exact optimization approaches computationally impractical for realistic problem sizes. Therefore, this study focuses on developing an effective metaheuristic solution approach capable of handling the high level of combinatorial complexity inherent in the MMTsUtALB problem.

$$min\ Z\ =\ 1.000X\ +\ 10Y \tag{1}$$

2.2. Assumptions

To formulate the MMTsUtALB problem, the following assumptions are adopted. These assumptions are consistent with those commonly used in two-sided and U-type assembly line studies (e.g., Delice et al., 2017a), with necessary extensions to accommodate mixed-model production.

- The assembly line has a two-sided U-type configuration consisting of an entrance arm and an exit arm, and assembly operations can be performed on both sides of the line.

- Each position on the U-type assembly line consists of four possible locations, and some locations may remain idle depending on task-side and precedence constraints.
- Operators perform tasks in parallel at stations located at facing locations on the U-line; however, parallel tasks or parallel stations are not allowed.
- A mixed-model production environment is considered, where multiple product models are assembled on the same line. All models share the same task precedence structure, while task processing times may differ among models.
- Task processing times are deterministic and independent of the assigned station or location but are model-dependent.
- Some tasks are restricted to a specific side of the product (left or right), whereas others can be performed on either side.
- Preemption of tasks is not allowed, and each task must be assigned entirely to a single station.
- No work-in-process inventory is permitted between stations, and products move sequentially along the U-type line.
- Operator travel times and material handling times between adjacent stations are assumed to be negligible.
- The cycle time is fixed and identical for all stations and all product models.

3. Solution Approach: GA-VNS Algorithm

Given the NP-hard nature of the MMTsUtALB problem, exact optimization methods become computationally impractical for realistic problem sizes. Consequently, metaheuristic algorithms have become the dominant solution approaches in the ALB literature due to their ability to generate high-quality solutions within reasonable computational times (Eghtesadifard et al., 2020; Mohammed et al., 2021). Therefore, in this study, a hybrid approach is developed by combining the GA and VNS. Within this scope, this section first presents fundamental information regarding the basic structure of the GA, followed by that of the VNS.

GA is utilized as the optimization method due to its effectiveness in solving complex and nonlinear problems. GA work with a set of candidate solutions, where each solution is encoded as a chromosome representing the decision variables of the problem. Originally introduced by Holland (1975) and later extended by Goldberg (1989), GA belong to the class of population-based metaheuristics and are mainly valued for their ability to explore the search space through evolutionary mechanisms. In a typical GA procedure, an initial population is generated randomly in order to create diversity among solutions. Each individual is then evaluated according to a fitness measure that reflects the objective of the optimization problem.

During the evolutionary process, individuals with better fitness values are more likely to be selected for reproduction. New solutions are produced by applying crossover operators, which combine information from parent chromosomes and allow promising regions of the search space to be explored further. To avoid loss of diversity, mutation is applied with a relatively small probability. An elitist replacement strategy is commonly used so that the best-performing solutions are retained in subsequent generations. This process continues until a stopping condition is reached, such as a convergence criterion or the maximum number of iterations.

GA-based methods have been frequently applied to different variants of the assembly line balancing problem, including mixed-model, two-sided, or U-type layouts (Kim et al., 2009; Delice et al., 2016; Zhang et al., 2018). However, performance can decrease as the problem becomes more constrained. In the early stages of the search process, clustering around local optimal solutions can prevent reaching the optimum (Pandey et al., 2014). Such obstacles have made the hybrid use of local optimization mechanisms in addition to population-based search necessary (Blum & Roli, 2003).

In the proposed approach, VNS is used precisely in this way. The search strategy explores the vicinity of promising solutions by employing predefined neighborhood structures throughout different phases of the solution procedure. Starting from an initially good solution, successive improvement steps are performed through controlled modifications. Based on the quality of the resulting solution, the search either proceeds from the updated solution or switches to an alternative neighborhood structure. This adaptive use of multiple neighborhoods supports diversification and reduces the likelihood of premature convergence (Mladenović & Hansen, 1997; Hansen & Mladenović, 2001; Hansen et al., 2008).

In this study, a hybrid GA–VNS framework is used to address the MMTsUtALB problem. GA is used to create a pool of various candidate solutions and to direct the search to different regions of the solution space. VNS, on the other hand, is applied to refine these candidate solutions through local modifications. This combination of GA and VNS reveals a powerful solution mechanism for the complex and highly constrained MMTsUtALB problem.

3.1 General Framework of the GA-VNS Algorithm

The proposed solution framework adopts a hybrid search scheme that combines population-oriented mechanisms with neighborhood-based refinement steps. This combination enables the model to address multiple constraint types inherent to the line balancing problem, including precedence relations, heterogeneous task times, and side-specific assignment restrictions. During the search, solution candidates are iteratively generated and updated, while selected solutions are further improved through targeted local adjustments when required.

GA is used mainly to maintain a diverse set of solutions and to guide the search toward alternative task assignments rather than to perform detailed local improvements. VNS is then applied as a refinement procedure, where neighborhood changes are used to improve solution quality and to avoid getting trapped in locally optimal solutions. In this way, exploration and improvement are handled in separate but connected stages of the algorithm.

The main components of the proposed algorithmic framework are summarized in Fig. 1 and Fig. 2 and can be described through the following phases:

Initialization Phase: During the initialization phase, the fundamental parameters of the GA and VNS, along with all variables required throughout the algorithm, are defined. The parameters of the GA and VNS adopted in the proposed GA–VNS algorithm are summarized in Table 1, and parameter tuning was performed experimentally. An initial population of feasible solutions is then generated using a problem-specific priority rule–based encoding scheme. Each chromosome does not directly represent a task sequence. Instead, it encodes priority information that guides the selection of tasks from a candidate list during the decoding-based task assignment procedure. This information applies to all product models.

Table 1
Parameters of the GA and VNS used in the proposed GA-VNS algorithm

Component	Parameter	Description	Value
GA	Population size	Number of individuals in the population	500
	Selection method	Parent selection strategy	Tournament selection
	Crossover probability	Probability of applying crossover	0.80
	Mutation probability	Probability of mutation	0.05
	Elitism	Number of elite individuals preserved	10
	Maximum generations	Upper bound on GA iterations	1000
VNS	Number of neighborhood structures (k_{max})	Total number of neighborhood structures	4
	Shaking Limit	Number of shaking moves per neighborhood	5
GA–VNS	VNS application strategy	Best Individual of GA is improved by VNS	Best individual
Termination	Time limit	Maximum computational time	3600 seconds
	Iteration Limit	Maximum generations of GA	1000

Priority Rules–based Encoding and Decoding-Based Task Assignment Phase

An effective encoding and decoding scheme is a critical component of metaheuristic algorithms for complex ALB problems, particularly for the MMTSUTALB problem. Due to the presence of precedence constraints, side-dependent task restrictions, U-type task execution characteristics, and model-specific processing times, conventional encoding approaches are insufficient to guarantee feasible solutions. Therefore, a problem-specific priority rule–based encoding and decoding-based task assignment framework is developed in this study.

Within the proposed framework, priority rules are embedded into the GA–VNS structure to address the MMTSUTALB problem. These rules are employed during the initialization phase to generate the required representations and during the decoding phase to guide the selection of tasks from the candidate tasks pool. In the proposed encoding scheme, a chromosome does not represent a fixed task order or explicit task assignments. Instead, it defines a sequence of heuristic priority rules that are used during the decoding phase of the solution construction. The chromosome is therefore modeled as an integer sequence, where each gene refers to a specific priority rule that becomes active at a particular stage of task assignment. This type of representation allows different task selection strategies to be combined within a single solution and supports a broader exploration of alternative assignments.

During the decoding process, the priority rule indicated by the current gene is applied to select a task from the available candidate set. The selected task is then assigned, and the procedure continues with the next gene in the sequence. Since rules are applied dynamically during decoding, the same priority rule may be used more than once within a chromosome. In the proposed approach, the set of priority rules is chosen from those that were shown to perform well in previous studies, particularly the rules reported in Özcan et al. (2022), and these rules are summarized below.

- Rule 1: Maximum ranked positional weight
- Rule 2: Longest processing time
- Rule 3: Maximum average ranked positional weight
- Rule 4: Minimum task number
- Rule 5: Maximum total number of successor tasks
- Rule 6: Random assignment

This encoding structure enables the GA to efficiently apply crossover and mutation operators without violating precedence constraints, while also providing sufficient flexibility for local search operations within the VNS framework.

The decoding-based task assignment procedure transforms a given priority rule sequence into a feasible two-sided U-type assembly line configuration for all product models. Initially, the first position is opened, and the corresponding workstations are created at the four available locations associated with this position. The available processing time for each workstation is

set equal to the predefined cycle time. As a general characteristic of U-type ALs, an assignable task pool is initially formed consisting of all tasks with no predecessors or no successors. Subsequently, tasks are considered according to the priority rules specified in the chromosome. For each selected task, its feasibility of assignment to a workstation at one of the available locations is evaluated based on side-dependent task restrictions (left-only, right-only, or either side), remaining available processing time for all models, and U-type flow direction constraints that allow both forward and backward task assignments. In mixed-model production, a task can be assigned to a workstation only if it does not violate the cycle time constraint for any model; otherwise, the task is removed from the candidate pool. When the pool becomes empty, a new position is opened and the candidate pool is updated. This process continues until all tasks are assigned, resulting in a complete and feasible two-sided U-type assembly line configuration for all models.

The proposed encoding and decoding-based task assignment structure guarantees full compliance with complex problem constraints while maintaining flexibility in the search process. Moreover, this structure enables seamless integration with GA operators and VNS neighborhood structures, thereby enhancing the robustness and overall effectiveness of the proposed GA–VNS algorithm.

Fitness Evaluation: Each individual in the population is evaluated using the objective function of the MMTSUTALB problem, as defined in Eq. (1). The resulting fitness value reflects the assembly line balancing performance of the decoded solution across all product models.

Global Search via GA Operators: The GA component of the proposed approach is responsible for global exploration. Selection, crossover, and mutation operators are applied to the population to generate new offspring solutions. The selection operator favors high-quality individuals, while crossover and mutation introduce diversity and enable exploration of new regions of the solution space.

The algorithm makes use of standard GA operators, which are adapted to the proposed chromosome representation. Parent solutions are selected using a tournament-based mechanism. This process involves choosing the best-performing individual from a small subset of randomly selected individuals from the population. This selection process is repeated until a sufficient number of parents is obtained. Tournament selection is adopted mainly because it is simple to implement and allows selection pressure to be adjusted without eliminating population diversity, which is an important consideration for combinatorial problems.

New solutions are generated through a one-point crossover operation applied to selected parent chromosomes. A crossover point is randomly chosen along the priority rule sequence, and segments of the parent chromosomes are exchanged to form offspring. Since chromosomes encode priority rules rather than explicit task orders, this operation does not directly conflict with precedence relations and can be combined naturally with the decoding-based assignment procedure to produce feasible solutions.

Variation in the population is further supported through a mutation operation. Mutation is implemented by randomly replacing a rule index in the chromosome. A gene in the chromosome is selected and replaced by another priority rule chosen from the available rule set. This simple modification introduces stochastic changes into the population and helps prevent the search from stagnating at early stages. Because mutation affects only rule indices, the feasibility of the decoded solution is preserved. This step is mainly intended to introduce variation into the population and is consistent with standard GA implementations (Goldberg, 1989; Deb, 2004).

The parameter values associated with the selection, crossover, and mutation operators are summarized in Table 1. After applying these operators, newly generated offspring are decoded and evaluated before being introduced into the population.

Local Improvement via VNS: Following the application of genetic operators, some of the promising solutions undergo an additional optimization step based on VNS. This step is used to locally modify solutions by applying different neighborhood moves, rather than performing a complete search from scratch. In this phase, alternative neighborhood structures are evaluated one after another, and the current solution is adjusted when an improvement is achieved. If no improvement is observed, a different neighborhood structure is tested. This procedure supports the search continuing without remaining fixed around a single solution structure.

The neighborhood moves used in the VNS component are summarized in Table 2. A total of four neighborhood structures ($k_{max} = 4$) are defined, all of which operate directly on the priority rule sequence. This choice ensures compatibility with chromosome representation and prevents inconsistencies in the decoding-based task assignment process.

Table 2
Neighborhood structures (kmax = 4) employed in the VNS component (priority-rule–based)

k	Neighborhood Structure	Description
k_1	Priority Swap	Swap the relative order of two priority rules
k_2	Priority Insertion	Relocate a priority rule within the chromosome
k_3	Partial Priority Reordering	Reorder priority rules related to a task
k_4	Priority Rule Set Perturbation	Simultaneous modification of multiple rules

Population Update: The improved solutions obtained after the application of genetic operators and the VNS-based local search are merged into the current population. A replacement strategy is then employed in which individuals are ranked according to their fitness values, and the best-performing 10 solutions are retained for the next generation. To preserve population diversity and avoid premature convergence, duplicate or highly similar individuals are discarded, and a subset of inferior but diverse solutions is allowed to remain in the population.

Termination Criterion: The algorithm iterates through the GA–VNS cycle until a predefined termination condition is satisfied. The termination criteria include reaching a maximum number of generations (set to 1000) or exceeding a predefined computational time limit of 3600 seconds. The best solution obtained during the entire search process is retained and reported as the final output. The overall procedure of the proposed solution approach is summarized in the following pseudocode and flow chart in Fig. 1 and Fig. 2.

```
Input: Problem data (task information, precedence relationships, processing times of each task for each model)
Output: Best feasible solution found

Initialize GA and VNS parameters (see Table 1)
Generate an initial population using a priority rule–based encoding scheme

while termination condition is not met do
    for each chromosome do
       Consider individual using a decoding-based task assignment procedure to construct an assembly line layout that satisfies all
         constraints
       Evaluate the fitness of individual and update the best-so-far solution set
    End for

    for each selected offspring solution do
       Apply VNS local search:
         k ← 1
         while k ≤ k_max do
            Modify the priority rule sequence according to neighborhood structure k
            Consider individual using a decoding-based task assignment procedure to construct an assembly line layout that satisfies all
              constraints
            Evaluate the fitness of individual
            if improved solution found then
               Replace current solution with improved solution
               k ← 1  // restart from first neighborhood
            else
               k ← k + 1  // proceed to next neighborhood
            end if
         end while
    end for
    Merge improved solutions into the best-so-far solution set
    Apply a replacement strategy to retain the best-so-far solution set and preserve population diversity

    Select parent solutions using the GA selection strategy
    Apply crossover and mutation operators to generate each offspring
end while
Return best solution found
```

Fig. 1. Pseudo Code of Proposed Algorithm

4. An Illustrative Example

To illustrate the main characteristics and search behavior of the proposed GA-VNS algorithm, a numerical example is considered. In this example, two product models are assembled concurrently on a two-sided U-type assembly line with a cycle time of 5 seconds. The presented example is solved for two different scenarios that may arise depending on which arm of the U-shaped line the product enters. The P12 problem (Kim et al., 2000) consists of 12 tasks, whose combined precedence relations, task side assignment requirements and the task processing times corresponding to both models (Özcan & Toklu, 2009) are provided in Table 3. Fig. 3 illustrates the optimal assembly line balancing solution obtained by the proposed GA-VNS for the P12 problem when the right side of the product is located at the center of the U-Type line. According to Fig. 3, the resulting MMTsUtALB layout consists of two positions, each comprising four locations. In the first position, two non-crossover stations are placed on the entrance and exit arms, respectively (Station 1 at Location 1 and Station 3 at Location 4), along with one crossover station (Station 2) shared between Locations 2 and 3. In the second position, two non-crossover stations are formed at Locations 1 and 2, whereas Locations 3 and 4 remain unoccupied. The resulting line layout utilizes a total of five stations.

Table 3
Task processing times for the numerical example

Task Number	Task Side	Successor Task	Model 1	Model 2
1	L	4	2	3
2	R	5	3	3
3	E	6	2	0
4	L	7	3	2
5	E	7, 8, 9	1	2
6	L	9	1	0
7	E	10	3	2
8	R	10	3	1
9	E	11	2	1
10	L	-	2	3
11	E	12	0	2
12	R	-	0	1

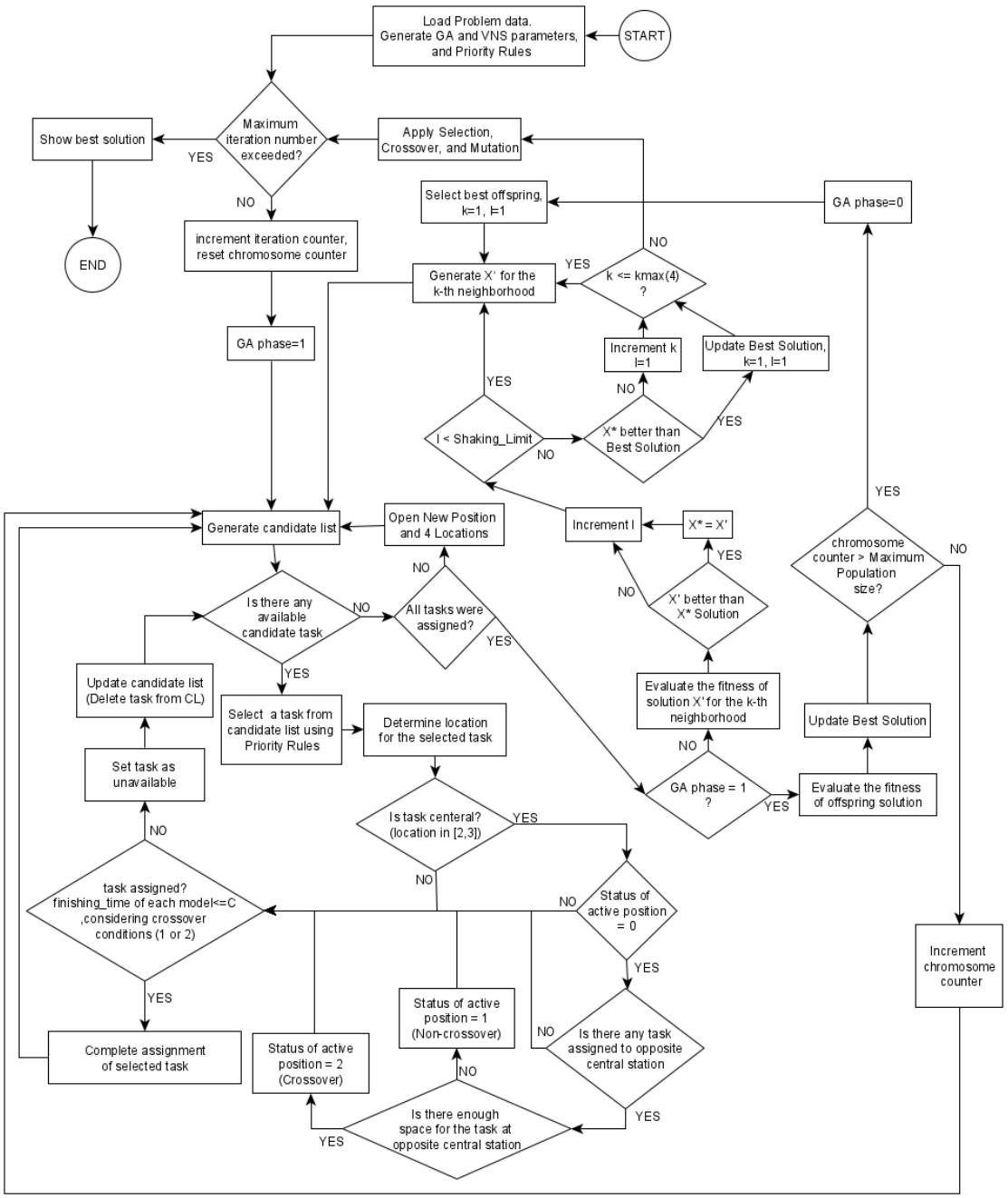
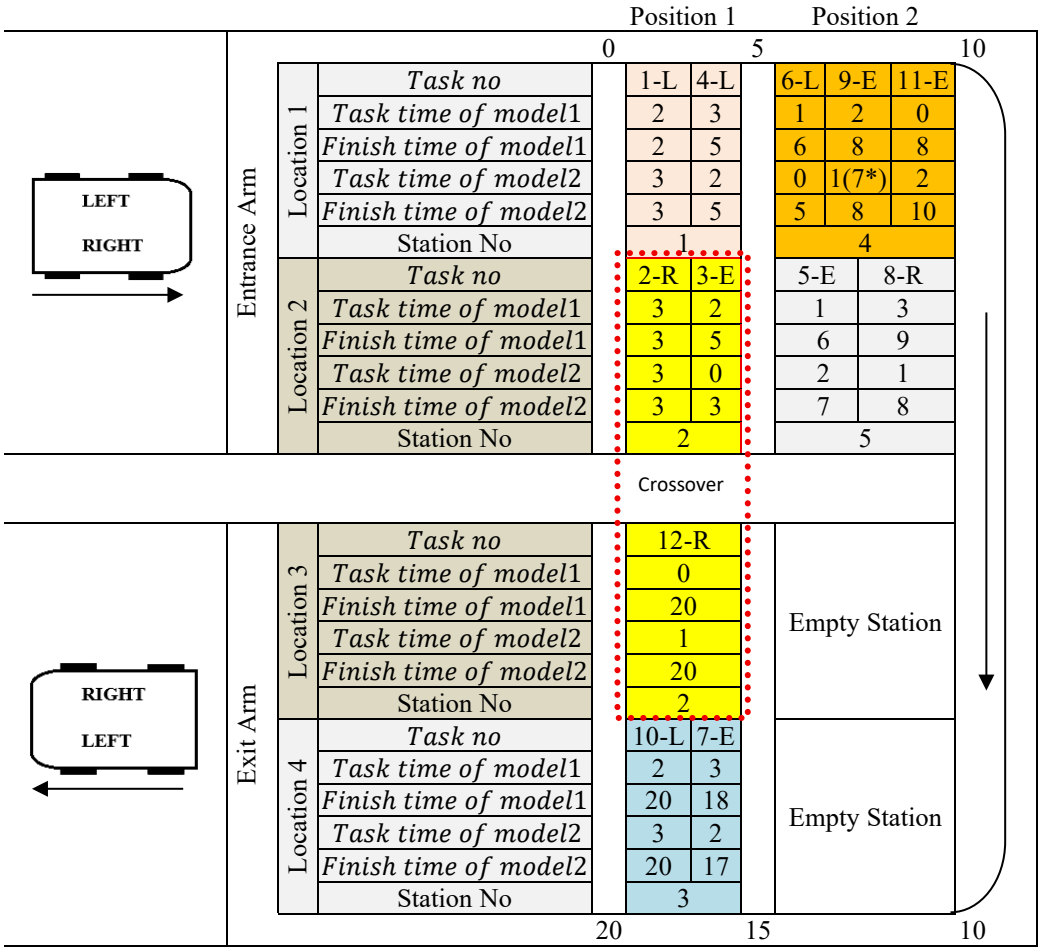


Fig. 2. General structure of the proposed GA-VNS Algorithm

A closer look at the solution shows that, in the Right-Sided Center scenario, tasks classified as R and E are placed in the central locations, namely Locations 2 and 3, whereas tasks of types L and E are assigned to the other available locations. The

crossover station formed at Locations 2 and 3 in the first position is highlighted using red dashed lines. At this station, the operator initially performs Tasks 2 and 3 and subsequently crosses to the opposite side of the line to carry out Task 12 at the third location. For both models, the cycle time constraint of 5 seconds is satisfied.

In certain cases, the inclusion of such crossover stations strengthens the proposed algorithm by enabling more efficient layout solutions that require fewer operators. Therefore, the algorithm is designed to consider, in the same position, both solutions that include crossover stations and alternative solutions composed solely of non-crossover stations, depending on the problem conditions. The task number, location, processing time, completion time, and station number of each task are shown in detail in Fig. 3. The direction of operation of the assembly line is symbolized by arrows. The product moves from left to right at the entrance arm and from right to left at the exit arm.

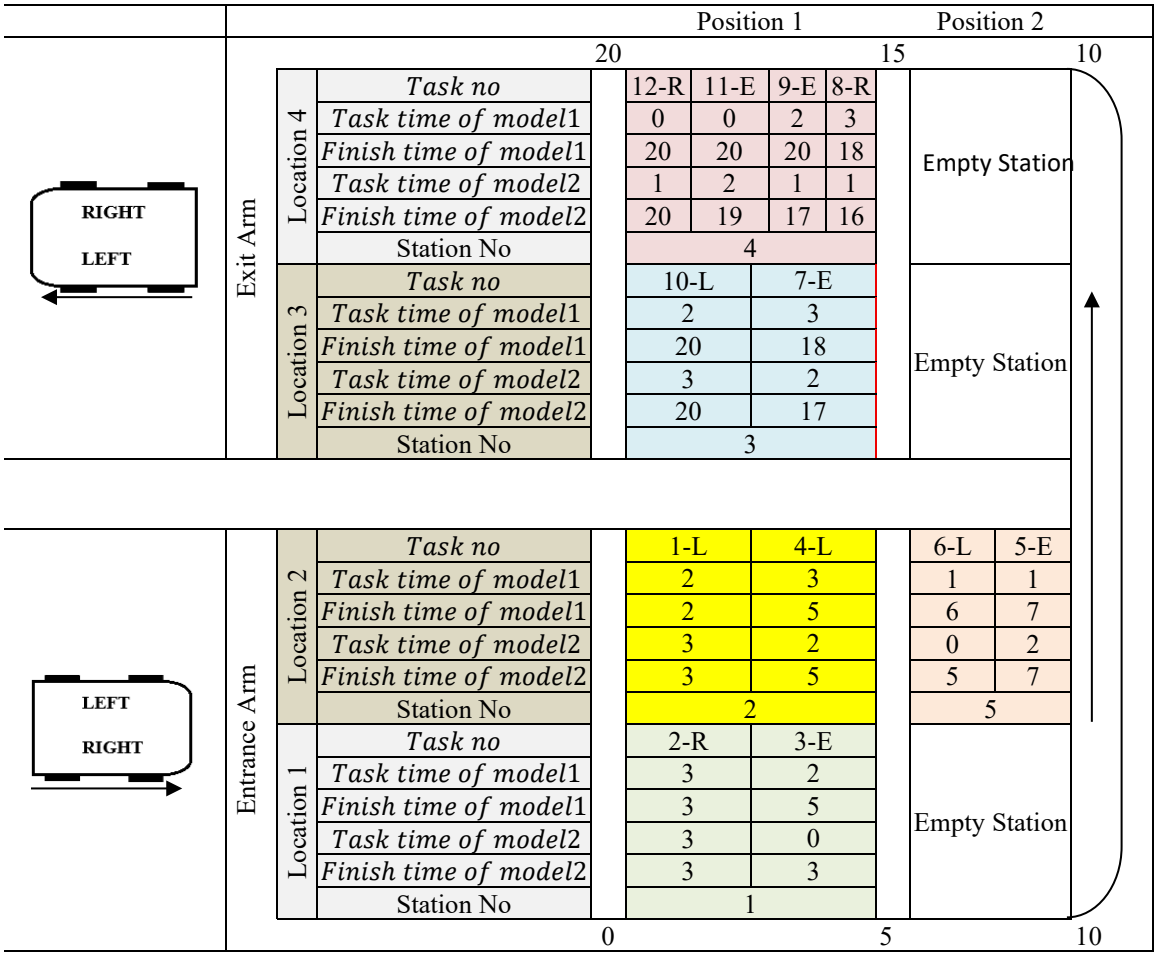


* Maximum finishing time of immediate predecessors of related task
Numberr of Stations= 5, Numberr of Positions= 2, Fitness= 5020

Fig. 3. Balanced solution results when the right side of the product is located at the center of the U-Type line

Fig. 4 illustrates the optimal assembly line balancing solution obtained by the proposed GA-VNS for the P12 problem when the left side of the product is located at the center of the U-Type line. In this case, only left-sided and either-type tasks can be assigned to the center stations of the line (Locations 2 and 3). This configuration represents the opposite case of that shown in Fig. 3. According to Fig. 4, the resulting MMTsUtALB model consists of two positions, each comprising four locations. In this solution, no crossover stations are formed. In the first position, four non-crossover stations are obtained. In the second position, only one non-crossover station is formed at Location 3, while no assignments are made to the other locations. The total number of utilized stations is again five.

An examination of the solution indicates that, under the Left-Sided Center case, L and E tasks are assigned to the central locations (Locations 2 and 3), whereas R and E-type tasks are assigned to the remaining locations. Task assignments to stations are carried out in accordance with the cycle time constraints. The task number, location, processing time, completion time, and station number of each task are shown in detail in Fig. 4. The arrow indicates the operational direction of the assembly line. Compared to the previous case, the product enters the assembly line through the opposite arm of the U-line, such that the arm previously serving as the exit arm now functions as the entrance arm. In this new entrance arm, the operation direction is from left to right, whereas it is from right to left on the exit arm. As evidenced by both cases, altering the product entry direction into the assembly line results in different solutions. Accordingly, the proposed algorithm is applied to both cases, and the obtained results are presented separately.



Numberr of Stations= 5, Number of Positions = 2, Fitness= 5020

Fig. 4. Balanced solution results when the left side of the product is located at the center of the U-Type line

5. Computational Study

A series of benchmark instances widely cited in the literature was employed to validate the performance of the proposed GA-VNS algorithm. The experimental evaluation is carried out using a collection of benchmark problems that includes four small-sized instances (P9, P12, P16, and P24) as well as two larger instances (P65 and P148). Among these, P9, P12, and P24 originate from the study of Kim et al. (2000), while P16 and P65 were later introduced by Lee et al. (2001). The largest instance in this set, P148, was originally proposed by Bartholdi (1993). To further assess the scalability of the proposed method, the P205 instance reported by Lee et al. (2001) is also incorporated into the analysis. Across all benchmark instances, the original precedence relations and task orientation constraints are preserved. Task processing times are taken from the dataset provided by Özcan and Toklu (2009) for all problems except P205. For this particular instance, task durations are instead adopted from the study of Delice et al. (2017b).

The GA-VNS algorithm was implemented using Delphi 7, and all computational experiments were conducted on a personal computer equipped with an Intel(R) Core(TM) i7-8550U processor operating at 1.8 GHz and 8 GB of RAM. All benchmark problems were solved under identical hardware and software conditions to ensure a fair and consistent evaluation. Each case was run five times, and the best results are presented. The performance of the proposed GA-VNS algorithm was comparatively analyzed against SA (Özcan & Toklu, 2009) and PSO (Delice et al., 2017b) across all test instances. These reference algorithms were originally developed for mixed-model two-sided ALs; however, no benchmark instances exist for the MMTsUtALB problem. Therefore, comparisons were conducted using studies on two-sided mixed-model ALs employing identical model task times. Accordingly, the mated-station number reported in the reference studies was considered equivalent to the number of positions in the two-sided U-Type line configuration. The lower bound (LB) values presented in the tables were obtained from Özcan and Toklu (2009) and Delice et al. (2017b). Solutions were generated for two scenarios in which the left and right sides of the product are positioned at the center of the U-Type line, and these were compared with the results of the SA and PSO algorithms. Superior solutions obtained using the GA-VNS method are highlighted in bold letters, while superior NS results are highlighted with a background marker. The resulting computational outcomes and performance indicators are presented in Tables 4, 5, and 6.

Table 4
Computational outcomes for small-scale problems

Problem	M#	Cycle Time	LB	SA			PSO			GA-VNS					
										Center of Line is Right			Center of Line is Left		
				NP	NS	CPU Time	NP	NS	CPU Time	NP	NS	CPU Time	NP	NS	CPU Time
P9	2	4	4	3	4	1.07	3	4	<1	**2**	4	<1	**2**	4	<1
		5	3	2	3	0.96	2	3	<1	**1**	3	<1	**1**	3	<1
		6	3	2	3	0.91	2	3	<1	**1**	3	<1	**1**	3	<1
P12	2	5	5	3	5	2.17	3	5	<1	**2**	5	<1	**2**	5	<1
		6	4	2	4	2.11	2	4	<1	**1**	4	<1	**1**	4	<1
		7	3	2	4	2.05	2	4	<1	**1**	4	<1	**1**	4	<1
		8	3	2	3	1.93	2	3	<1	**1**	3	<1	**1**	3	<1
P16	2	15	5	4	6	5.18	4	6	<1	**2**	6	<1	**2**	6	<1
		16	5	4	6	5.08	4	6	<1	**2**	6	<1	**3**	**5**	<1
		18	4	3	5	5.05	3	5	<1	**2**	5	<1	**2**	5	<1
		19	4	3	5	4.82	3	5	<1	**2**	**4**	<1	**2**	5	<1
		21	4	2	4	4.71	2	4	<1	**1**	4	<1	**1**	4	<1
		22	4	2	4	4.65	2	4	<1	**1**	4	<1	**1**	4	<1
P24	2	20	7	4	7	17.70	4	7	<1	**2**	7	<1	**2**	7	<1
		24	6	3	6	18.71	3	6	<1	**2**	6	<1	**2**	6	<1
		25	5	3	6	17.46	3	6	<1	**2**	6	<1	**2**	6	<1
		30	5	3	5	16.23	3	5	<1	**2**	5	<1	**2**	5	<1
		35	4	2	4	15.09	2	4	<1	**1**	4	<1	**1**	4	<1
		40	4	2	4	15.08	2	4	<1	**1**	4	<1	**1**	4	<1

M#: Number of product models

As shown in Table 4, the proposed GA-VNS algorithm demonstrated strong performance on small-scale test problems, achieving optimal solutions in under one second of CPU time for all runs. This performance is superior to SA and comparable to PSO. For the P16 instance, GA-VNS achieves a lower number of stations than PSO at cycle time values of 16 and 19, indicating that the proposed approach can yield more compact line configurations even for small-scale problems.

Table 5
Computational outcomes for P65 and P148

Problem	M#	Cycle Time	LB	SA			PSO			GA-VNS					
										Center of Line is Right			Center of Line is Left		
				NP	NS	CPU Time	NP	NS	CPU Time	NP	NS	CPU Time	NP	NS	CPU Time
P65	3	326	8	5	9	528	5	9	110	**3**	9	100	**3**	9	46
		381	7	4	8	523	4	8	<10	**3**	8	<1	**2**	8	8
		435	6	4	7	515	4	7	<10	**2**	7	8	**2**	7	9
		490	6	3	6	505	3	6	<10	**2**	6	5	**2**	6	8
		544	5	3	6	504	3	6	<1	**2**	6	<1	**2**	6	<1
P148	4	204	13	9	17	4025	9	17	<10	**4**	**16**	22	**4**	**16**	25
		255	11	7	14	3859	7	13	<10	**4**	13	9	**4**	13	7
		306	9	6	12	3845	6	11	<10	**3**	11	10	**3**	11	9
		357	8	5	10	3835	5	9	<10	**3**	9	8	**3**	9	7
		408	7	5	10	3826	4	8	<10	**2**	8	7	**2**	8	9
		459	6	4	8	3810	4	7	<10	**2**	7	6	**2**	7	6
		510	6	4	8	3829	3	6	<15	**2**	6	12	**2**	6	18

M#: Number of product models

Table 5 reports the comparative results obtained for the P65 and P148 benchmark problems using GA-VNS, PSO, and SA. For the P65 instance, GA-VNS matches the number of stations achieved by SA and PSO, while requiring less CPU time than SA and a comparable computational effort to PSO. In the case of P148, GA-VNS consistently attains fewer stations than SA, and at a cycle time of 204 it also outperforms PSO in terms of the number of stations. Across all tested cycle times, the computational effort of GA-VNS remains close to that of PSO and is consistently lower than that of SA.

Table 6
Computational outcomes for P205

Problem	M#	Cycle Time	LB	SA		PSO			GA-VNS					
									Center of Line is Right			Center of Line is Left		
				NP	NS	NP	NS	CPU Time	NP	NS	CPU Time	NP	NS	CPU Time
P205	5	1133	11	8	15	7	14	420	**4**	14	42	**4**	14	17
		1322	9	7	13	6	12	<10	**3**	12	6	**3**	12	8
		1510	8	6	12	6	11	<60	**3**	**10**	88	**3**	**10**	85
		1699	7	5	10	5	10	<10	**3**	10	8	**3**	10	8
		1888	7	5	10	4	8	<20	**2**	8	17	**2**	8	15
		2077	6	5	9	4	8	<10	**2**	8	9	**2**	8	8
		2266	6	4	8	4	8	<10	**2**	8	5	**2**	8	5
		2454	5	4	8	3	6	<25	**2**	6	21	**2**	6	22
		2643	5	4	7	3	6	<10	**2**	6	5	**2**	6	5
		2832	5	3	6	3	6	<10	**2**	6	3	**2**	6	3

M#: Number of product models

Table 6 presents the best solutions obtained for the P205 instance. The results indicate that, across seven cycle time settings, GA-VNS requires fewer stations than SA and attains these solutions with lower computational times. At a cycle time of 1510, GA-VNS also achieves a lower number of stations than PSO, while for the remaining cycle times both methods yield the same number of stations. Beyond solution quality, GA-VNS demonstrates a clear computational advantage. Considering the increased complexity of the MMTsUtALB problem in large-scale instances, these results suggest that the proposed approach remains effective under challenging problem conditions.

Table 7
Problem-Based Quantitative Improvements of GA-VNS

Problem	Scale	NP Gain vs SA	NP Gain vs PSO	NS Gain vs SA	NS Gain vs PSO	CPU Gain vs SA	CPU Gain vs PSO
P9–P12	Small	0–30%	0–25%	0–15%	0–10%	~10–20×	~1–2×
P16–P24	Small	15–35%	15–30%	5–20%	5–15%	10–20×	~1–2×
P65	Medium	25–40%	20–35%	10–20%	5–15%	≥10×	~2–5×
P148	Large	55–70%	30–50%	25–40%	15–30%	20–30×	~2–4×
P205	Very Large	40–60%	30–50%	20–35%	10–25%	≥10×	~2–3×

Table 7 presents the comparative performance of GA-VNS with respect to SA and PSO in terms of problem-specific gains. For small-sized instances, the differences among the methods in both the number of positions and stations remain limited, which can be attributed to the fact that all algorithms yield solutions close to the lower bounds. However, as the problem scale increases, GA-VNS achieves more pronounced reductions in NP and NS. At the same time, it preserves a clear advantage in computational efficiency over SA and delivers consistent, though less substantial, improvements compared to PSO. Across all instances, GA-VNS generally outperforms PSO in NS, reaching higher NS values in a total of 4 cycle time scenarios. These results indicate that the effectiveness of GA-VNS becomes particularly evident for medium, large, and very large instances, where improvements in solution quality are accompanied by shorter computation times.

6. Conclusion

This study focused on the MMTsUtALB problem by bringing together mixed-model production, two-sided operations, and a U-type assembly line layout within a single problem setting. While each of these characteristics frequently appears in real manufacturing environments, most earlier studies have treated them separately or in pairs. Addressing them jointly makes it possible to observe certain interactions that cannot be identified when simpler assembly line balancing models are used.

To solve the problem, a hybrid approach based on GA and VNS was adopted. Task assignments were handled through priority-rule representations, and a decoding procedure was used to generate feasible balances for two-sided U-shaped lines. The use of both global and local search mechanisms, rather than a single solution strategy, enabled the exploration of different assignment structures and supported gradual improvement of promising solutions during the search process.

The proposed GA–VNS method was evaluated using a number of benchmark instances. Since no standard test set exists specifically for the MMTsUtALB problem, the results were interpreted by comparing them with those reported in previous studies on two-sided assembly line balancing. The computational analysis shows that the proposed approach produces stable and competitive solutions. In particular, for larger problem instances, the number of positions was reduced by approximately 55–70% when compared with SA and by about 30–50% when compared with PSO. In addition, moderate reductions in the number of stations were obtained, together with consistently lower computational times.

Some simplifying assumptions were made in this study, which naturally limit the scope of the results. Factors such as variability in task times, setup-related effects, ergonomic issues, and energy considerations were not taken into account. Including these elements would make the problem more realistic, although also more difficult to solve. In addition, future work could consider alternative objectives or more complex line structures, such as parallel U-type layouts or environments with changing product mixes.

Acknowledgments

This work was supported by the Scientific Research Projects Coordination Unit of Kayseri University (Grant No. FBA-2026-1232).

References

Abdullah Make, M. R., Ab. Rashid, M. F. F., & Razali, M. M. (2017). A review of two-sided assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, *89*(5), 1743-1763.

Ağpak, K., Yegül, M. F., & Gökçen, H. (2012). Two-sided U-type assembly line balancing problem. *International Journal of Production Research*, *50*(18), 5035-5047.

Bartholdi, J.J. (1993). Balancing two-sided assembly lines: A case study. *International Journal of Production Research 31*, 2447–2461.

Baybars, I. (1986). An efficient heuristic method for the simple assembly line balancing problem. *International Journal of Production Research*, *24*(1), 149-166.

Baykasoğlu, A., & Dereli, T. (2008). Two-sided assembly line balancing using an ant-colony-based heuristic. *The International Journal of Advanced Manufacturing Technology*, *36*(5), 582-588.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, *35*(3), 268-308.

Boysen, N., Schulze, P., & Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years?. *European Journal of Operational Research*, *301*(3), 797-814.

Chutima, P. (2020). Research trends and outlooks in assembly line balancing problems. *Engineering Journal*, *24*(5), 93-134.

Çimen, T., Baykasoğlu, A., & Demirkol Akyol, S. (2022). A detailed review and analysis of assembly line rebalancing problems. *Assembly Automation, 42*(6), 742-760.

Deb, K. (2004). Introduction to genetic algorithms for engineering optimization. In *New optimization techniques in engineering* (pp. 13-51). Berlin, Heidelberg: Springer Berlin Heidelberg.

Delice, Y., Aydoğan, E. K., Özcan, U., & İlkay, M. S. (2017a). Balancing two-sided U-type assembly lines using modified particle swarm optimization algorithm. *4OR*, *15*(1), 37-66.

Delice, Y., Aydoğan, E. K., Söylemez, I., & Özcan, U. (2018). An ant colony optimisation algorithm for balancing two-sided U-type assembly lines with sequence-dependent set-up times. *Sādhanā*, *43*(12), 199.

Delice, Y., Kızılkaya Aydoğan, E., & Özcan, U. (2016). Stochastic two-sided U-type assembly line balancing: a genetic algorithm approach. *International Journal of Production Research*, *54*(11), 3429-3451.

Delice, Y., Kızılkaya Aydoğan, E., Özcan, U., & İlkay, M. S. (2017b). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, *28*(1), 23-36.

Eghtesadifard, M., Khalifeh, M., & Khorram, M. (2020). A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017. *Computers & Industrial Engineering, 139*, 106182.

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, *1989* (102), 36.

Gutjahr, A. L., & Nemhauser, G. L. (1964). An algorithm for the line balancing problem. *Management science, 11*(2), 308-315.

Hamzadayi, A., & Yildiz, G. (2013). A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. *Computers & Industrial Engineering*, *66*(4), 1070-1084.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of OperationalResearch*, *130*(3), 449-467.

Hansen, P., Mladenović, N., & Moreno Perez, J. A. (2008). Variable neighbourhood search: methods and applications. *4OR*, *6*(4), 319-360.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Jiao, Y. L., Jin, H. Q., Xing, X. C., Li, M. J., & Liu, X. R. (2021). Assembly line balance research methods, literature and development review. *Concurrent Engineering, 29*(2), 183-194.

Jiao, Y., Su, X., Li, L., & Wu, Z. (2025). An improved ant colony optimization algorithm for two-sided U-type assembly line balancing problems. *Engineering Optimization*, *57*(5), 1245-1262.

Khorasanian, D., Hejazi, S. R., & Moslehi, G. (2013). Two-sided assembly line balancing considering the relationships between tasks. *Computers & Industrial Engineering*, *66*(4), 1096-1105.

Kim, Y. K., Song, W. S., & Kim, J. H. (2009). A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers & Operations Research*, *36*(3), 853-865.

Kim, Y.K., Y. Kim, Y.J. Kim., 2000. Two-sided assembly line balancing: a genetic algorithm approach. Production Planning and Control, 11 (1), 44-53.

Kucukkoc, I., & Zhang, D. Z. (2014). Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International Journal of Production Research*, *52*(12), 3665-3687.

Kucukkoc, I., & Zhang, D. Z. (2016). Integrating ant colony and genetic algorithms in the balancing and scheduling of complex assembly lines. *The International Journal of Advanced Manufacturing Technology*, *82*(1), 265-285.

Kucukkoc, I., & Zhang, D. Z. (2017). Balancing of mixed-model parallel U-Type assembly lines considering model sequences. *International journal of production research*, *55*(20), 5958-5975.

Kucukkoc, I., Li, Z., Karaoglan, A. D., & Zhang, D. Z. (2018). Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm. *International Journal of Production Economics*, *205*, 228-243.

Lee, T.O., Y. Kim, Y.K. Kim., 2001. Two-sided assembly line balancing to maximize work relatedness and slackness. Computers and Industrial Engineering, 40, 273–292.

Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (2016). Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Advances in Mechanical Engineering*, *8*(9), 1687814016667907.

Li, Z., Kucukkoc, I., & Nilakantan, J. M. (2017). Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. *Computers & Operations Research, 84*, 146-161

Macaskill, J.L.C., 1972. Production-line balances for mixed model lines. *Management Science, 19*, 423-434.

Miltenburg, J., Wijngaard, J., 1994. The U-line line balancing problem. *Management Science 40*, 1378–1388.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097-1100.

Mohammed, F. D., Zakaria, M. Z., Ramli, M. F., Jusoh, M., Azizan, M., & Fadzli, N. (2021, May). Metaheuristic optimization in solving assembly line balancing problems: A short review. In *AIP Conference Proceedings* (Vol. 2339, No. 1, p. 020002). AIP Publishing LLC.

Özcan, U. (2010). Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. *European Journal of Operational Research*, *205*(1), 81-97.

Özcan, U., Aydoğan, E. K., Himmetoğlu, S., & Delice, Y. (2022). Parallel assembly lines worker assignment and balancing problem: A mathematical model and an artificial bee colony algorithm. *Applied Soft Computing*, *130*, 109727.

Özcan, U., Toklu, B., 2009. Balancing of mixed-model two-sided assembly lines. *Computers and Industrial Engineering, 57* (1), 217-227.

Pandey, H. M., Chaudhary, A., & Mehrotra, D. (2014). A comparative review of approaches to prevent premature convergence in GA. *Applied Soft Computing*, *24*, 1047-1077.

Rashid, M. F. F., Hutabarat, W., & Tiwari, A. (2012). A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology, 59*, 335-349.

Razali, M. M., Kamarudin, N. H., Ab. Rashid, M. F. F., & Mohd Rose, A. N. (2019). Recent trend in mixed-model assembly line balancing optimization using soft computing approaches. *Engineering Computations, 36*(2), 622-645.

Salveson, M. E. (1955). The assembly-line balancing problem. *Transactions of the American Society of Mechanical Engineers*, *77*(6), 939-947.

Scholl, A., & Klein, R. (1999). ULINO: Optimally balancing U-Type JIT assembly lines. *International journal of production research*, *37*(4), 721-736.

Simaria, A.S., & Vilarinho, P.M., (2009). 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers and Industrial Engineering, 56*(2), 489-506.

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology, 73*, 1665-1694.

Sparling, D., & Miltenburg, J. (1998). The mixed-model U-line balancing problem. *International journal of production research*, *36*(2), 485-501.

Thomopoulos, N.T. (1970). Mixed model line balancing with smoothed station assignments. *Management Science 16,* 593–603.

Yang, M. S., Ba, L., Liu, Y., Zheng, H. Y., Yan, J. T., Gao, X. Q., & Xiao, J. M. (2019). An improved genetic simulated annealing algorithm for stochastic two-sided assembly line balancing problem. *International Journal of Simulation Modelling*, *18*(1), 175-186.

Yegül, M. F., Ağpak, K., & Yavuz, M. (2010). A new algorithm for U-Type two-sided assembly line balancing. *Transactions of the Canadian Society for Mechanical Engineering*, *34*(2), 225-241.

Zhang, Y., Hu, X., & Wu, C. (2018). A modified multi-objective genetic algorithm for two-sided assembly line re-balancing problem of a shovel loader. *International Journal of Production Research*, *56*(9), 3043-3063.



© 2026 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).