Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-023-02182-5

Swarm intelligence-based framework for accelerated and optimized assembly line design in the automotive industry

Anass El Houd[1,2] · Benoit Piranda[1] · Raphael De Matos[2] · Julien Bourgeois[1]

Received: 30 March 2023 / Accepted: 7 July 2023

Abstract

This study proposes a dynamic simulation-based framework that utilizes swarm intelligence algorithms to optimize the design of hybrid assembly lines in the automotive industry. Two recent discrete versions of Whale Optimization Algorithm (named VNS-DWOA) and Gorilla Troops Optimizer (named DGTO) were developed to solve the assembly line balancing problem. The effectiveness of these algorithms was compared to six conventional meta-heuristics as well as the solution proposed by process design experts. The experimental results show that our methods outperform the conventional meta-heuristics and achieve comparable or better results than the experts' solution. Particulary, VNS-DWOA, being the top performer, has consistently provided averagely remarkable enhancements of cycle time, ranging from 7% when compared to the process expert's solution to 20% maximum improvement compared to all other methods. The findings of this study highlight the effectiveness of utilizing swarm intelligence algorithms and dynamic simulation-based frameworks as well as the potential benefits of implementing these digital methods in industrial settings, as they can significantly accelerate and enhance the optimization of assembly line design particularly and reduce time to market generally.

Keywords Industry 4.0 · Assembly line balancing · Swarm intelligence · Discrete metaheuristics · Combinatorial optimization · Distributed algorithms

Introduction

The design of an assembly line is a complex process that consists in finding an optimal solution among almost infinite combinations of scenarios. This procedure carries considerable weight, as it directly affects both manufacturing costs and the quality of the end product. In essence, Assembly Line Design comprises three key components: the Manufacturing Bill Of Material (MBOM), which is the product's recipe specifying the list of components, assembly tasks and precedence graph; the layout, which encompasses the physical manufacturing environment such as workstations, storages, and operators; and the assembly scenario, which depicts the sequence of assembly tasks and their allocation to specific workstations and workers. Until now, the traditional manual way of designing an assembly line involves gathering a team of experts from various departments, such as production, manufacturing, engineering and logistics, to brainstorm and design the optimal layout of the assembly line. The team members typically rely on their expertise and experience to propose potential solutions and debate the pros and cons of each alternative. The team may iterate over multiple designs until reaching a consensus on the final assembly line solution. While this approach can leverage the collective expertise of the team members and allow for valuable insights and recommendations, it may also suffer from subjectivity, bias, and limited scope due to the team members' experience and background. Moreover, the manual process can be time-consuming, labor-intensive and prone to errors, espe-

✉ Anass El Houd
anass.elhoud@forvia.com

Benoit Piranda
benoit.piranda@univ-fcomte.fr

Raphael De Matos
raphael.dematos@forvia.com

Julien Bourgeois
julien.bourgeois@univ-fcomte.fr

[1] FEMTO-ST Institute, CNRS, University of Franche-Comte, 25200 Montbeliard, France

[2] R&D Center, Forvia Clean Mobility, 25250 Bavans, France

cially when dealing with complex assembly line balancing problems in large-scale production environments.

This problem is known as the general formulation of the Assembly Line Balancing Problem (ALBP), a term regularly employed to refer to the decision-making process that aims to optimally partition and balance the assembly tasks among the workstations and workers in relation with some objectives in a continuous production system. ALBP handles the distribution of tasks among workstations and workers ensuring that a minimum number of resources have approximately the same amount of work to do. This strategy improves different key performance indicators (KPI) such as the cycle time, the overall equipment effectiveness, and production costs (Adnan et al., 2016; Kumar & Mahto, 2013). The configuration of an assembly line is a multiplex process considered NP-hard (Hosseini & Al Khaled, 2014). The optimization of this system is an important part of many commercial manufacturing models and has gained attention from both industrial and research partners.

The ALBP has three main classes: ALBP-1 minimizes resources for a given cycle time, ALBP-2 finds a balanced assignment solution with minimum cycle time for a fixed number of resources, and ALBP-E maximizes total line efficiency through task and workstation combinations. ALBP-1 is less significant in current industrial contexts since resource numbers are predetermined based on machine time and have minimal variation. ALBP-3 is more complex as it aims to simultaneously minimize cycle time and resource numbers. We chose ALBP-2 for our study due to its variability and optimization potential. Optimizing resource allocation using ALBP-2 allows us to balance complexity and practical applicability. Industrial gains focus on optimizing allocation scenarios rather than layout changes due to existing machine investments.

To summarize, given a set of tasks, the objective is to minimize the cycle time, or equivalently, to maximize the production rate by finding the best feasible scenario assigning the tasks to the workstations and workers while managing other constraints. Different types of assignment restrictions or constraints are considered:

- Precedence constraints: the tasks are partially ordered by anteriority relations defining a precedence graph that should be respected.
- Incompatibility restrictions: some assembly tasks cannot be carried out on the same workstation to prevent distortion of parts or robot's incompatible trajectories. Assignments not allowed for these reasons are commonly referred to as forbidden assignments.
- Manufacturing rules: they include all the company standard process rules to be respected such as Hoshin rules (Giordani da Silveira et al., 2017) and one piece flow concept.

This work aims to perform a comprehensive experimental study of enhanced state-of-the-art swarm intelligence-based meta-heuristics to solve the constrained ALBP-2 for predefined layouts in the automotive industry.

Our research actively supports the digital transformation of the industry by developing an intelligence-based tool specifically designed for manufacturing line design workshops and digital twin tools (Stan et al., 2023; Prashar, 2023; Tliba et al., 2022). This tool enables faster and more efficient creation of assembly lines, ultimately reducing the time it takes to bring new products to market. Additionally, our research provides valuable insights and recommendations for optimizing existing assembly lines, leading to increased productivity and efficiency. On the academic side, our work significantly contributes to the evaluation and application of swarm intelligence-based methods in a real industrial setting. We focus on a specific use case and thoroughly investigate the effectiveness of these methods, particularly their impact on search efficiency. Furthermore, we explore various techniques such as discretization, utilization of different transfer functions, and hybridization with other heuristics to enhance the performance and effectiveness of swarm intelligence-based methods in solving complex optimization problems.

Background

The development of search and optimization algorithms has been a significant area of computer science, artificial intelligence and operations research over the past few decades. In particular, exact optimization algorithms are methods that guarantee to find the optimal solution to a given problem within a finite number of steps. These algorithms rely on mathematical programming and linear algebra techniques to explore the search space and systematically identify the best possible solution. On the other hand, meta-heuristic algorithms are general-purpose search algorithms that can be applied to a wide range of optimization problems. They are designed to find high-quality solutions in a shorter amount of time, even when dealing with very large search spaces (Salhi & Thompson, 2022).

Swarm intelligence, which has inspired the development of several meta-heuristic algorithms, is based on the study of collective behavior in decentralized, self-organized systems, called search agents, such as groups of animals or social insects. Particle Swarm Optimizer (PSO) (Katoch et al., 2021) is recognized as one of the earliest swarm-based algorithms. It involves a group of particles that moves around in a search space and communicates with each other to find the optimal solution to a given problem. Another example is Ant Colony Optimization (ACO) (Dorigo & Di Caro, 1999), a family of algorithms that are inspired by the foraging behavior of ants. Grey Wolf Optimizer (GWO) (Mirjalili et al., 2014) is inspired by the social hierarchy and hunting behavior of grey wolves, while the Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016) is inspired by the hunting behavior of humpback whales. These different algorithms have been used to solve a variety of optimization problems, including the traveling salesman problem, routing and scheduling problems (Alorf, 2023).

The No-Free-Lunch theorem (Wolpert & Macready, 1997) asserts that even though there are various meta-heuristic algorithms, the demand for additional ones exists. This is because a meta-heuristic algorithm that performs the most on a particular type of problem may not achieve the same level of performance when applied to a different type of problem. Therefore, it is essential to create more problem-oriented meta-heuristics that are customized to each problem's specific features. Thus, a meticulous examination and comprehension of the problem's structure is a mandatory prerequisite, along with the invention of innovative search techniques.

Recently, Swarm intelligence-based algorithms have shown remarkable progress in solving global optimization problems, with Gorilla Troops Optimizer (GTO) (Abdollahzadeh et al., 2021), Sheep Flock Optimization Algorithm (SFOA) (Kivi & Majidnezhad, 2022), Tuna Swarm Optimization (TSO) (Xie et al., 2021), Gannet Optimization Algorithm (GOA) (Pan et al., 2022) and Honey Badger Algorithm (HBA) (Hashim et al., 2022) being examples of such very recent algorithms. These algorithms have outperformed traditional and advanced optimization methods on various benchmark functions in terms of efficiency and outcomes.

Apart from designing new algorithms, researchers have also explored the hybridization of meta-heuristics with other techniques like local search, machine learning, or other meta-heuristics to enhance their effectiveness. These hybrid algorithms can be created by combining operators, parameters, or components of the parent algorithms in different ways. Several articles show that the resulting hybrid algorithm can be more robust, efficient and effective than the parent algorithms. Rajpurohit et al. (Rajpurohit & Sharma, 2022) have proposed a hybrid algorithm that combines the features of Jellyfish search optimizer (JSO) and Sine–cosine algorithm to create a new algorithm that outperforms both individual algorithms. Al-Betar et al. (2022) have combined adaptive β-hill climbing as a local search algorithm with six swarm-based meta-heuristics to boost the training process of neural networks. Another recent method for predicting a back break in open-pit blasting with high accuracy has been introduced by Dai et al. (2022). This approach combines the strengths of random forest (RF) and particle swarm optimization (PSO) to create a new hybrid intelligence approach.

We opted to further develop the Whale Optimization Algorithm (WOA) due to its consistent performance and reliability when discretized using heuristics or transfer functions. Moreover, WOA demonstrated impressive outcomes across a diverse set of combinatorial problems, as supported by the studies conducted by Becerra-Rozas et al. (2022), Wang et al. (2023), and Yu et al. (2022). On the other hand, our emphasis on the Gorilla Troops Optimizer (GTO) stemmed from its novelty and the potential it carries in addressing the limitations of conventional optimization methods. As a relatively recent approach, GTO introduces enhanced search operators, introducing fresh perspectives and innovative techniques. By leveraging the cooperative behavior observed in gorilla troops, GTO presents a compelling solution for overcoming the inherent challenges within our specific problem domain.

Related works

Most research on the Assembly Line Balancing Problem (ALBP) has mainly concentrated on a static assessment of the assembly line, assuming that the cycle time of each resource is simply the sum of the individual tasks duration of the tasks performed by that resource without considering the dynamic interactions between workers, workstations and all equipment along the line. While this static approach is quite fast, it only provides a basic approximation of the real cycle time of the assembly line. As a result, this approximation may not accurately capture the true cycle time, potentially leading to inaccuracies in the optimization process. This, in turn, could impact the effectiveness of the search and exploration process of the optimization method used.

Initially, there were numerous efforts to address ALBP through the use of exact methods, especially integer and linear programming and dynamic constrained programming (Bukchin & Raviv, 2018; Walter et al., 2021; Mehmet et al., 2020). Recently, Yadav and Agrawal (2022) proposed a mathematical model of the robotic assembly line problem to find an exact solution approach to this constrained problem. However, these methods always fail in solving complex combinatorial optimizations, their run-time experiences a significant surge and has no practical relevance when considering real-world problems (Mirjalili, 2020). Consequently, several researchers shifted their attention towards recent meta-heuristic methods due to their ability to provide near-to-optimum solutions in a reasonable time (Mutingi et al., 2017; Suwannarongsri & Puangdownreong, 2009; Rodríguez et al., 2018). Chen et al. (2023) have implemented a bi-level multi-objective genetic algorithm to optimize the number of stations and their workload smoothness. Tang et al. (2022) have proposed an improved multi-objective multifactorial evolutionary algorithm to optimize assembly production and equipment maintenance by treating it as a multi-task optimization problem.

Using swarm intelligence algorithms to solve the ALBP represents a significant challenge due to their primary focus

Table 1 Most Known Meta-heuristics & Recent Swarm-based Algorithms Summary (D: Discrete, C: Continuous)

Optimizer	Year	Citations	Type	Behavior
SA (Černý 1985)	1983	51,107	D	Physics-based
GA (Katoch et al., 2021)	1992	26,434	D	Evolution-based
PSO (Katoch et al., 2021)	1995	67,564	C	Swarm-based
ACO (Dorigo & Di Caro, 1999)	2006	13876	D	Swarm-based
GWO (Mirjalili et al., 2014)	2014	6165	C	Swarm-based
WOA (Mirjalili & Lewis, 2016)	2016	3731	C	Swarm-based
GTO (Abdollahzadeh et al., 2021)	2021	402	C	Swarm-based
TSO (Xie et al., 2021)	2021	66	C	Swarm-based
SFOA (Kivi & Majidnezhad, 2022)	2022	24	C	Swarm-based

on continuous optimization problems, making them less effective in solving discrete problems. To address this limitation, discretization strategies can be employed to modify these algorithms. A potential solution to this issue is presented by Pornsing et al. (2022), who introduced a new approach for discretizing a variant of Particle Swarm Optimization (PSO), resulting in improved efficiency and performance when tackling discrete problems. Similarly, a new hybrid binary whale-hawks representation was presented by Alwajih et al. (2022) using a transfer function to convert the continuous attributes in binary to solve the features selection problem. The reverse problem, called the disassembly line balancing problem (DLBP), was also tackled by Yao et al. (xxxx), as they have focused on the implementation of a novel metaheuristic algorithm called cat swarm optimization (CSO) and have demonstrated its effectiveness in solving type-1 DLBP.

Owing to the mixed-integer, nonlinear and dynamic behavior of our problem, previous adaptations are no longer useful. Proper changes and problem-oriented modifications should be included to address our real-world discrete problem (Mirjalili, 2020).

To summarize, we are encountering three main issues in directly applying state-of-the-art meta-heuristics:

- They are generally problem-independent methods, with no prior knowledge about the real problem as a guide.
- They are generally continuous and cannot be directly applied to discrete problems.
- Search space knowledge is not fairly propagated through the search agents of the methods.

For that, we propose enhanced, discrete and problem-oriented versions of swarm intelligence-based optimizers based on Whale Optimizer Algorithm and Gorilla Troops Optimizer and compare them with six conventional benchmark algorithms.

Mathematical model formulation

Given a list of tasks $T = \{t_1, t_2, \ldots, t_n\}$, a list of identical parallel workstations $S = \{s_1, s_2, \ldots, s_m\}$ and a list of workers $W = \left\{W_1, W_2, \ldots, W_p\right\}$. We note P the set of pairs of tasks (i,k) such as i precedes k, and F is the set of pairs of tasks (i,k) such as i and k cannot be assigned to the same workstation. An assembly scenario is mathematically modeled by the $(n \times m)$ assignment matrix A_s:

$$A_s = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots \\ \vdots & \ddots & \\ a_{m,1} & & a_{m,n} \end{bmatrix} \tag{1}$$

where $a_{i,j} = 1$ if the task t_i is assigned to workstation s_j and 0 otherwise.

n: total number of tasks.
m: total number of workstations.
p: total number of manual workers.
i: the index of tasks, i=1,...,n
j: the index of workstations, j=1,...,m
P: the set of precedence pairs.
F: the set of restricted pairs.
CT: the overall cycle time of the assembly line.
MT: the machine time (automatic).
WC: the work content (manual).
TT: the technical time (additional).

Each task t_i is composed of three time components: machine time (MT), technical time (TT), and work content (WC). MT is the time required for machines or robots to perform the task, while TT includes technical actions like launching the operation or opening the workstation door. WC pertains to human actions, such as handling and transferring parts between workstations.

The optimization problem is expressed by Equation (2a):

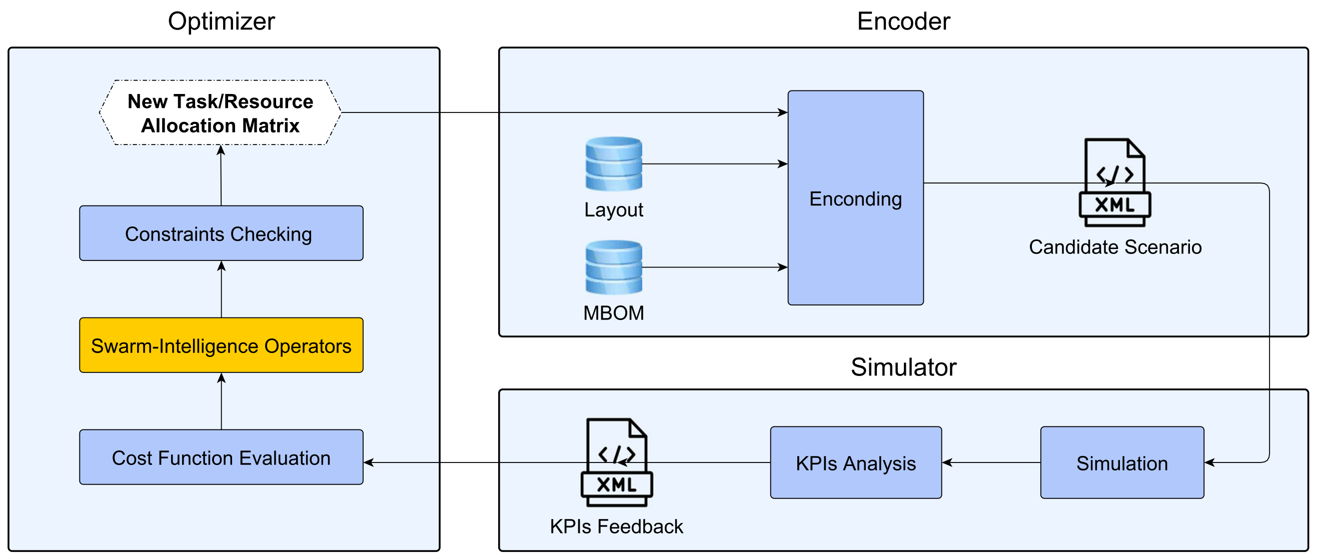


Fig. 1 Our proposed simulation-based optimization framework

$$\min_{(a_{i,j})} \quad CT(A_s, W) \tag{2a}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} a_{i,j} = 1, \quad i = 1, 2, \ldots, n, \tag{2b}$$

$$\sum_{j=1}^{m} j \cdot a_{i,j} \leq \sum_{j=1}^{m} j \cdot a_{k,j} \quad \forall (i,k) \in P \tag{2c}$$

$$a_{i,j} \neq a_{k,j} \quad \forall (i,k) \in F \tag{2d}$$

$$a_{i,j} \in \{0,1\} \quad \forall i \in \{1, 2, \ldots, n\} \tag{2e}$$

Constraint 2b expresses that every task is assigned to a unique workstation, Eq. 2c refers to the precedence constraint. Constraint 2d, also known as the incompatibility constraint, ensures that incompatible tasks are not assigned to the same workstation. The goal of the optimization problem stated in equation 2a is to minimize the global cycle time of the assembly line, which refers to the time taken to finish a single unit of the product. This cycle time is influenced by both the assignment of tasks to workstations and the assignment of workers to workstations, as each task consists of three components: MT, TT, and WC. To simplify the problem, each operator is assigned to a fixed set of workstations.

$$CT(A_s, W) = \max_{j=1,..,m} \left\{ \sum_{i=1}^{n} a_{i,j} \left(MT_i + TT_{i,j} + WC_{i,j} \right) \right\} \tag{3}$$

The machine time (MT_i) is deterministic and exclusively determined by the task t_i, while the technical time ($TT_{i,j}$) and work content ($WC_{i,j}$) are subject to fluctuations throughout the assembly cycle due to their dependence on both the task and the workstation, making the problem more complicated. Our objective is to assign tasks to workstations in a way that minimizes the total time needed to perform all three types of actions (MT, TT, WC). Prior research in this area has generally employed objective functions that consider only machine times of the tasks, and such functions have demonstrated efficacy in estimating cycle time for static assembly lines that involve only one type of assembly action. Nevertheless, in our specific use case, which encompasses three different types of actions per task, computing the global cycle time is more complex. To address this challenge, we have created and integrated a sophisticated manufacturing process simulator into our framework.

End-to-end framework

Our proposed global framework, named Automatic Manufacturing Design Optimizer (AMDO), consists of three primary blocks as demonstrated in Fig. 1. Firstly, the optimizer generates improved solutions and explores new assembly scenarii. Then, the decoder converts the mathematical representation of the solution proposed by the optimizer (A_s) into an XML file format, which is transmitted to the simulator. Solution evaluation is conducted using a discrete event simulator, specifically customized for industrial use, called ManufactSim (Piranda et al., 2022) and showcased in Fig. 2. The simulator estimates the assembly line's cycle time by dynamically simulating the assembly process, including the machine time of each workstation, the working time of the operators per workstation and all related technical interactions. The results are then returned to the optimizer to enhance the previously proposed candidates.

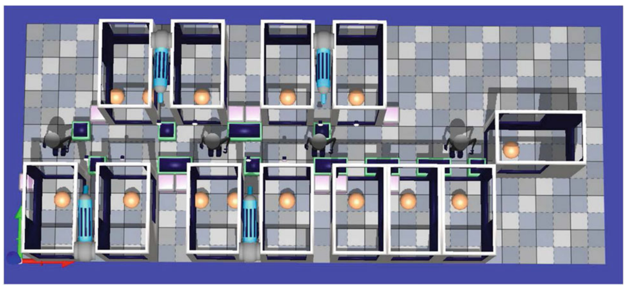


Fig. 2 Graphical interface of ManufactSim (Manufacturing Line Simulator)

As previously stated, determining the assembly cycle time can be difficult using traditional analytical methods due to the complexity and flexibility of hybrid assembly lines. These methods often simplify the process and do not account for dynamic interactions and fluctuations. However, using a simulator can provide an advantage as it allows for the consideration of these factors. Simulators are designed to capture the behavior of complex systems over time, including the dynamic interactions and fluctuations that occur within the system. They provide a more accurate representation of the system by incorporating a greater level of detail and complexity. Our internal studies have shown that it is typically necessary to produce at least 30 parts on average to achieve a stable cycle time of the assembly line. Our simulator offers a low-cost, secure and efficient way to estimate KPIs of a given assembly scenario. It takes 2 s on average to simulate the assembly of 30 products. The development of a fast simulator was the main inspiration behind the creation of this entire framework.

VNS-discrete whale optimizer

Original WOA

The Whale Optimizer Algorithm (WOA) (Mirjalili & Lewis, 2016) is inspired by the hunting behavior of humpback whales. Humpback whales adopt an attacking method called the bubble-net attacking method which includes three steps: encircling prey, spiraling update position, and searching for the prey. The whale's population size is set at the beginning of the optimization.

Encircling prey

In this process, the location of the so-far best solution (the prey) is identified and surrounded. The whales move closer and progressively to the location of the prey to encircle it starting from an initial position that can be given or selected randomly. This behavior is expressed in Equation (4).

$$\begin{aligned} X(t+1) &= X^*(t) - A \cdot D \\ D &= \mid C \cdot X^*(t) - X(t) \mid \end{aligned} \tag{4}$$

$$\begin{aligned} A &= 2 \cdot a \cdot r_1 - a \\ C &= 2 \cdot r_2 \end{aligned} \tag{5}$$

where X and X^* denote respectively the position of a whale and the so-far best solution, A and C coefficient vectors, r_1, r_2 uniform random numbers between 0 and 1, and $a = 2 \cdot (1 - \frac{iter}{iter_{max}})$.

Spiral update behavior

During the hunting phase, each humpback whale updates its position following a spiral path using the mathematical model in Equation (6):

$$\begin{aligned} X(t+1) &= D' \cdot e^{bl} \cdot \cos(2\pi l) + X^*(t) \\ D' &= \mid X^*(t) - X(t) \mid \end{aligned} \tag{6}$$

where X is the position vector of a whale, X^* is the so-far best solution (or the prey), l is a uniform random number between [-1, 1].

Searching for prey

To avoid getting stuck in local optima, the random search approach is proposed as an imitation of the mechanism of searching for the prey using the following equations:

$$\begin{aligned} D &= \mid C \cdot X_{rand}(t) - X(t) \mid \\ X(t+1) &= X_{rand}(t) - A \cdot D \end{aligned} \tag{7}$$

X_{rand} is a random whale from the population at the current iteration.

Our proposed VNS-DWOA

The original WOA was designed for continuous optimization problems and is not suitable for addressing our problem directly. We present a new discrete WOA by modifying its operators and adding VNS as a local heuristic search. This approach combines the global search of WOA with VNS's local search capability, providing a powerful and effective hybrid optimizer to our problem.

Discrete prey encirclement

As mentioned above, Whale Optimizer Algorithm adopts shrinking encirclement to update the positions of the whales. For our case, we mimic this behavior by following the so far best solution to help the whales approach the global optimal solution from their original position.

This continuous operation is replaced by a new discrete exploitation operator, modeled by the following equation:

$$\begin{aligned} a_{i,j_{min}}(t+1) &\leftarrow a_{i,j_{max}}(t) \\ a_{i,j_{max}}(t+1) &\leftarrow a_{i,j_{min}}(t) \end{aligned} \tag{8}$$

where

$$\begin{cases} j_{min} = \arg\min(CT_1(t), \cdots, CT_m(t)) \\ j_{max} = \arg\max(CT_1(t), \cdots, CT_m(t)) \end{cases} \tag{9}$$

We denote that i is the index of a randomly selected task from the list of tasks done in the workstation with the highest workload in the previous iteration t ($a_{i,j}(t) = 1$).

The whales adjust their positions using Equation (9), which incorporate information about the workload of each workstation from the previous scenario. The aim is to re-balance and re-distribute tasks efficiently. If the workload variance is high, the algorithm selects a couple of tasks at random from the bottleneck workstation (i.e., the one with the highest workload) and reassigns them to the workstation with the lowest workload. This strategy mimics the encircling of prey by humpback whales, who use their prior knowledge to explore promising sub-optimal regions that may contain good solutions (preys).

Discrete spiral update

Discretizing the spiral behavior of the original WOA is not straightforward. To preserve the effectiveness of this operator, we propose an initial step of encoding the assignment vector A_s as a continuous matrix B_s on which will be applied the spiral update. This relationship between the discrete matrix A_s and the continuous position matrix B_s is given by Equation (10)

$$A_s = \Phi(B_s) = \begin{bmatrix} \phi(b_{1,1}) & \phi(b_{1,2}) & \cdots \\ \vdots & \ddots & \\ \phi(b_{m,1}) & & \phi(b_{m,n}) \end{bmatrix} \tag{10}$$

where

$$a_{i,j} = \phi(b_{i,j}) = \begin{cases} 1, & \text{If rand } > \frac{1}{1+e^{-b_{i,j}(t)}} \\ 0, & \text{else} \end{cases} \tag{11}$$

The original spiral update is applied on the continuous values of the position matrix B_s as written in Equation (12)

$$\begin{aligned} B_s(t+1) &= D' \cdot e^{bl} \cdot \cos(2\pi l) + B_s^*(t) \\ D' &= \mid B_s^*(t) - B_s(t) \mid \end{aligned} \tag{12}$$

The continuous position matrix B_s is therefore squashed and converted back to the corresponding discrete assignment vector A_s using the transfer function Φ. The discrete spiral operator is encapsulated in Equation (13)

$$\begin{aligned} A_s(t+1) &= \Phi(D' \cdot e^{bl} \cdot \cos(2\pi l) + B_s^*(t)) \\ D' &= \mid B_s^*(t) - B_s(t) \mid \\ B_s(t) &= \Phi^{-1}(A_s(t)) \end{aligned} \tag{13}$$

Using this approach, we are able to apply the original spiral update on our discrete solution matrix A_s through the intermediary of the position matrix B_s.

Seaching for the prey with VNS

Searching for the prey is done randomly in the original version. To further improve the computational performance, we replace the random search with Variable Neighborhood Search (VNS) developed by Mladenović and Hansen (1997). This mechanism is based on systematic changes of neighborhoods to escape from local minimum (Hansen et al., 2017). Three neighborhood search operators are used:

- Swapping: randomly select two elements corresponding to different tasks and exchange their positions.
- Insertion: randomly select two elements corresponding to different tasks and then insert one to the front of another.
- Randomizing: select randomly a possible task to be assigned to a randomly selected workstation.

The pseudo-code of VNS-DWOA is given as follows (Algorithm 1)

Algorithm 1 VNS-DWOA

```
Input: $N_p$, MaxIter
Output: Best Solution
  procedure VNS- DWOA
    Generate random population of $N_p$ whale
    while $i < MaxIter$ do
      for each whale $A_s$ in population do
        Calculate the objective function of the whale
      end for
      $A^*$ is the so far best solution
      for each whale $A_s$ in population do
        Update a, A and C using Equation (5)
        if $rand(0, 1) > 0.5$ then
          if $\mid A \mid < 1$ then
            Update the whale position using Eq. (9)
          else
            Update the whale using VNS Operators
          end if
        else
          Update the whale by discrete spiral by Eq. (13)
        end if
      end for
    end while
    return $A^*$ Best Solution
  end procedure
```

Discrete Gorilla troops optimizer (DGTO)

Original GTO

Artificial Gorilla Troops Optimizer (GTO) is a recent meta-heuristic developed by Abdollahzadeh et al. (2021) and inspired by the social behavior of the gorilla troops in which five strategies are imitated, including moving to an unknown area, migrating to other gorillas, moving to known places, following the silver-back, and competing for adult females. These different strategies illustrate the process of exploration and exploitation in the gorilla kingdom.

Exploration phase

During the exploration phase, three mechanisms are employed. The first mechanism of moving to unknown places is selected when rand < p. If rand ≥ 0.5, the mechanism of movement towards other gorillas is chosen. Otherwise, the mechanism of migration to a known location is to be selected. This exploration phase is modeled in Equation (14)

$$GX(t+1) = \begin{cases} (UB - LB) \cdot r_1 + LB, & \text{rand } < p \\ (r_2 - C) \cdot X_r(t) + L \cdot H, & \text{rand } \geq 0.5 \\ X(i) - L \cdot (L \cdot (X(t) - GX_r(t)) + r_3 \cdot (X(t) - GX_r(t))), & \text{rand } < 0.5 \end{cases} \quad (14)$$

where $GX(t+1)$ shows the candidate gorilla's position vector in the next iteration. X(t) is the current gorilla's position vector, r_1, r_2, r_3, and rand are uniform random values between 0 and 1, and p is a hyper-parameter that determines the probability of selecting random migration. UB and UL are the upper and lower bounds of our solution variables. X_r is a randomly selected gorilla from the entire population. C, L and H are calculated using the following equations

$$C = (\cos(2 \cdot r_4) + 1) \cdot \left(1 - \frac{It}{MaxIt}\right) \quad (15)$$

$$L = C \cdot l \quad (16)$$

$$H = Z \cdot X(t) \quad (17)$$

where MaxIt is the total value of iterations to perform the optimization, l is a random value in the range of -1 and 1, r_4 is a uniform random value between 0 and 1 while Z is a uniform random value in the range of [-C, C].

Exploitation phase

The exploitation phase in GTO implies two strategies: following the silver-back and competing for adult females. The silver-back is considered the troop leader who takes decisions. The W parameter should be specified before starting the process. If C ≥ W, the strategy of following the silver-back is selected. This behavior is mathematically expressed

$$GX(t+1) = L \cdot M \cdot (X(t) - X_{silver-back}) + X(t) \quad (18)$$

$$M = \left(\left|\frac{1}{N}\sum_{i=1}^{N} GX_i(t)\right|^g\right)^{\frac{1}{g}} \quad (19)$$

$$g = 2^L \quad (20)$$

$X(t)$ represents the position vector of the gorilla, $X_{silverback}$ is the position vector of the silver-back gorilla. $GX(t+1)$ is the position vector of the candidate gorilla. L can be calculated using Equation (16). If C ≤ W, the strategy of competing for adult females is used. This mechanism imitates the competition of adult gorillas with other males for mature females and is coded as follows

$$GX(t+1) = X_{\text{silver-back}} - (X_{\text{silver-back}} \cdot Q - X(t) \cdot Q) \cdot A$$
$$Q = 2 \cdot r_5 - 1 \quad (21)$$

$$A = \beta \cdot E \quad (22)$$

A is a coefficient vector to determine the degree of violence in conflicts and is calculated using β, a parameter to be given value before the optimization operation while r_5 is a uniform random value between 0 and 1.

Our proposed DGTO

The original version of GTO is performed as well on a continuous space. So, many issues need to be resolved regarding its implementation to our discrete problem. At the time of writing this article, there is no published discrete version of this algorithm. Because of its superior results in benchmarks, it was more relevant to keep the original model of GTO and use the discrete mapping function Θ expressed in Equation (23).

$$A_s = \Theta(X) = \begin{cases} s_1, & 0 \leq x_i < r_1 \\ s_2, & r_1 \leq x_i < r_2 \\ \vdots \\ s_m, & r_{n-1} \leq x_i < 1 \end{cases} \quad (23)$$

We also present the discrete distance between the search agents (gorillas), calculated using Equation (24)

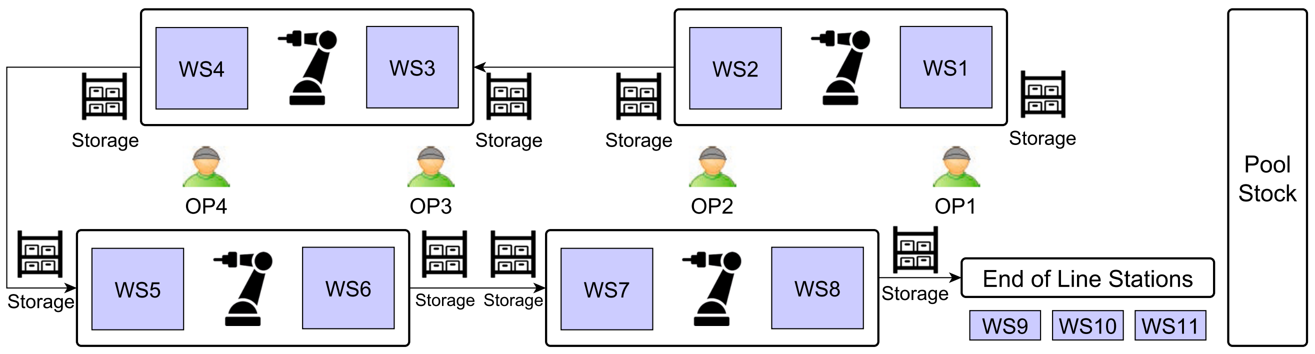


Fig. 3 Our use case's U-shape assembly line layout

$$< A_s^p, A_s^q >= \sum_{j=1}^{n} d_j,\ d_j = \begin{cases} 1, \mid a_{pj} - a_{qj} \mid \neq 0 \\ 0, \mid a_{pj} - a_{qj} \mid = 0 \end{cases}, p \neq q \tag{24}$$

where A_s^p and A_s^q are respectively the p-th and q-th search agents, a_{pj} refers to the j-th element of the p-th search agent.

Based on that, the discrete exploration phase is mathematically modeled by Equations (25) and (26)

$$A_s(t+1) = \begin{cases} \Theta[(UB - LB) \cdot r_1 + LB], & \text{rand } < p \\ \Theta[(r_2 - C) \cdot X_r(t) + L \cdot H], & \text{rand } \geq 0.5 \\ \Theta[X(t) - L \cdot \big(L \cdot < A_s(t), A_s^r(t) > & \\ \quad + r_3 \cdot < A_s(t), A_s^r(t) >\big)], & \text{rand } < 0.5 \end{cases} \tag{25}$$

where

$$X(t) = \Theta^{-1}(A_s(t)) \tag{26}$$

$A_s(t)$ and $X(t)$ are respectively the gorilla assignment vector and its continuous position vector at the iteration t, while $A_s^r(t)$ and $X_r(t)$ are respectively a random gorilla vector and its corresponding position vector. C, L and H are calculated using Equations (15), (16), and (17).

For the discrete exploitation phase of our DGTO, the competition for gorilla females is done using Equation (27)

$$A_s(t+1) = \Theta(X_{\text{silver-back}} - < A_s^{\text{silver-back}}, A_s(t) > \cdot Q \cdot A) \tag{27}$$

On the other hand, following the silver-back strategy by Equation (28)

$$A_s(t+1) = \Theta(L \cdot M \cdot < A_s^{\text{silver-back}}, A_s(t) > + X(t)) \tag{28}$$

where

$$\begin{aligned} X(t) &= \Theta^{-1}(A_s(t)) \\ X_{\text{silver-back}} &= \Theta^{-1}(A_s^{\text{silver-back}}) \end{aligned} \tag{29}$$

The pseudo-code of DGTO is presented below (Algorithm 2)

Algorithm 2 DGTO

```
Input: p, $\beta$, W, $N_p$, MaxIter
Output: Best Solution
 procedure DGTO
   Generate random population of $N_p$ gorilla
   while $i < MaxIter$ do
     for each gorilla $A_s$ in population do
       Update each gorilla position by exploring (Eq.25)
       Calculate the objective function of the gorilla
     end for
     $A^*$ is the so far best solution (silver-back)
     for each gorilla $A_s$ in population do
       Update C, L and H using Eqs. (15), (16), (17).
       if $C > W$ then
         Update gorillas by conflicting for females (Eq. 25)
       else
         Update gorillas by following silver-back (Eq. 28)
       end if
       Calculate the objective function of the new gorilla
     end for
     $A^*$ is the so far best solution (silver-back)
   end while
   return $A^*$ Best Solution
 end procedure
```

Experimental results

Use case

Our approach is tested on a use case involving a hybrid assembly line comprising four welding machines (V-cells), each equipped with a welding robot and has two identical workstations, and four manual workers. Followed by 3 end-of-line stations (Fig. 3).

The exhaust system, which is the product being manufactured, is comprised of 24 components that require 35 welds to be assembled as shown in Fig. 4.

Table 2 Parameters of the methods (Population Size, Hyper-parameters and number of iterations)

Optimizer	Pop. Size	Init. Parameters	Iterations
SA	1	$T_0 = 1, \alpha = 1 - it/it_{max}$	1000
GA	25	$p_{mut} = 0.2, p_{cros} = 0.8$	200
PSO	100	$C_1 = C_2 = C_3 = 0.5$	1000
ASrm	100	$PE = 0.01$	1000
ACSrm	100	$PE = 0.01$	1000
DGWO	25	None	100
VNS-DWOA	25	None	100
DGTO	25	p = 0.03, $\beta = 3$, W = 0.8	200

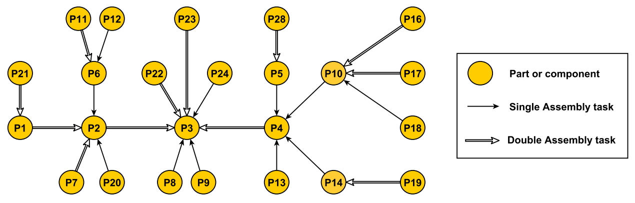


Fig. 4 Task Sequence of the use case's product: 24 parts assembled by 35 welding task

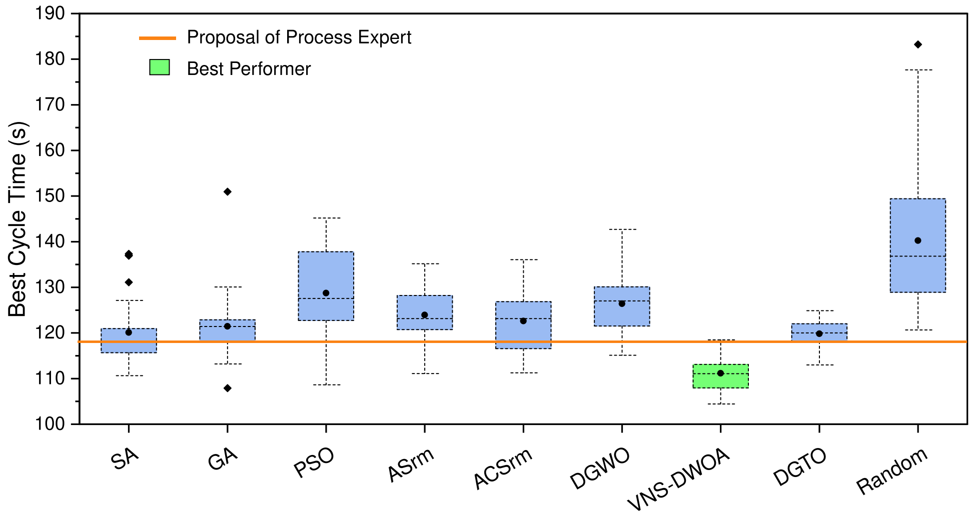
We note that the best assembly scenario proposed by the process expert (PE) has a cycle time of **118 s** and an average balancing error between resources of 6 s.

Experiments

To evaluate the effectiveness of the proposed algorithms, the following methods are being considered for comparison:

- Simulated Annealing (SA): the search space is explored by randomly generating new candidate solutions and performing small changes over the so-far best solution. A better solution is always accepted, while a worse solution might be accepted with a certain probability that decreases over time and is controlled by a temperature parameter (Vincent et al., 2022).
- Genetic Algorithm (GA): the placement and allocation decisions for all tasks in this method are determined by the GA operators (mutation, crossover, and selection).
- Particle Swarm Optimizer (PSO): the solution generation is based on the movement of particles through the search space and the update of their position in the direction of the best solution found so far by the swarm (Singh & Singh, 2023).
- Ant System with Random Search (ASrm): The solution space is inhabited by a group of ants that modify the overall level of pheromone during each iteration. An improved solution is selected by the quantity of accumulated pheromones (Zhou et al., 2022). We use a random search for local search.
- Ant Colony System with Random Search (ACSrm): A variant of ASrm is employed to reach the optimal solution by updating the pheromone level both locally and globally. Local pheromone updates are permitted for all ants, whereas only the so-far best solution can perform global updates (Chen et al., 2022).

Fig. 5 Box-and-whiskers plot comparing the performance of the algorithms for the Assembly Cycle Time Minimization over 40 runs



- Discrete Grey Wolf Optimizer (DGWO): the solution space consists of a population of grey wolves, where each wolf represents a possible solution. Then, the population is iteratively updated, using a transfer function, by following the guidance of the three fittest solutions, referred to as alpha, beta, and delta (Sharma et al., 2022).
- Random Guessing: the solution is constructed at random without any specific strategy or guidance. This approach involves randomly selecting a solution from a set of possible solutions and repeating the process until an acceptable solution is found.

All experiments were conducted on a Windows computer with Intel(R) Core(TM) CPU i9 3.50GHz and 128 GB RAM. All methods were launched for the same execution time of 1 h for 40 runs. For each of the 40 optimization trials, an independent initial population is randomly generated. The table 2 restates the parameters of the implemented methods.

T_0 and α are respectively the initial temperature and the cooling factor of the simulated annealing (SA). p_{mut} and p_{cros} are respectively the probability of mutation and the crossover of the genetic algorithm (GA). C_1 is the inertia weight and C_2, C_3 are the acceleration coefficients of PSO. PE is the pheromone evaporation (known also as the learning rate) of both ASrm and ACSrm. p, β and w are the parameters of DGTO presented in section 7.

Table 3 The best, worst, mean, and standard deviation (SD) values of assembly cycle time produced by all optimizers in seconds

Optimizer	Best	Worst	Mean	Median	SD
SA	110.64	137.33	120.10	118.05	6.37
GA	107.91	150.96	121.44	121.41	6.38
PSO	108.63	145.20	128.73	127.55	6.06
ASrm	111.10	135.17	123.94	123.14	5.34
ACSrm	111.26	136.07	122.60	123.14	6.11
DGWO	115.10	126.40	126.40	127.03	5.40
VNS-DWOA	**104.45**	**118.50**	**111.16**	**111.07**	3.84
DGTO	113.00	124.32	119.80	120.00	**2.90**
Random	120.66	183.22	140.24	136.82	15.01

Results & analysis

We showcase the outcomes of 40 optimization runs conducted to compare the performance of the different algorithms that were evaluated. The aim is to identify the most promising algorithms based on their performance.

As shown in Table 3, the optimization method VNS-DWOA exhibits superior performance compared to the other methods in terms of average cycle time, and best and worst solutions, with only a slightly higher deviation rate than DGTO. Additionally, VNS-DWOA was able to generate assembly scenarios with lower cycle times than those proposed by the process expert (PE) in the use case. Notably, all of the optimization methods tested outperform random guessing, indicating that even the simplest of methods can provide significant improvements over purely random selection.

These findings are further reinforced by the box-and-whisker plot presented in Fig. 5, which provides clear evidence that throughout all 40 experiments, VNS-DWOA consistently generates an assembly scenario with a cycle time comparable to, if not better than, the one proposed by the process expert within an hour of computation time.

When conducting experiments that involve multiple optimization algorithms, it is necessary to compare their performances accurately and reliably. Statistical tests provide a rigorous approach to evaluating the differences between the performances of these algorithms. However, the normality assumption that underlies many commonly used statistical tests is often violated when analyzing such data. To overcome this issue, a non-parametric statistical test analysis is recommended.

One of the most commonly used non-parametric tests for multiple comparisons is the Friedman test (Derrac et al., 2011). This test is used to compare the mean ranks of three or more groups and is robust to the presence of outliers or skewed data. The null hypothesis H_0 in the Friedman test is that there is no significant difference in the performance between all the optimizers.

If the Friedman test reveals a significant difference between the algorithms, it is important to conduct further analysis to determine which pairs of algorithms are different. The Bonferroni-Dunn test (Kashiwado et al., 2023) is a suitable posthoc test that allows for the comparison of all algorithms to one another using adjusted significance levels based on the number of pairwise comparisons being made. The Wilcoxon rank-sum test (Garren & Davenport, 2022) is another pairwise comparison test that compares the ranks of two groups to determine if there is a significant difference between them.

The results of the Friedman test, as reported in Table 5, indicate a significant difference between the performances of some of the algorithms being tested, as evidenced by a p-value of significantly lower than 0.05. This means that we can reject the null hypothesis H_0 and conclude that at least some of the algorithms have significantly different performances.

However, the Friedman test is unable to provide information regarding which pairs of algorithms exhibit significant differences. Therefore, we use the Bonferroni-Dunn test, a posthoc test that was applied to every pair of algorithms, resulting in 36 unique pairs. Table 4 summarizes the p-values obtained from this test.

Table 4 P-values of Bonferroni-Dunn test: Italic refers to the significant difference, Bold reflects no significant difference ($p \leq 0.05$)

P-value	**SA**	**GA**	**PSO**	**ASrm**	**ACSrm**	**DGWO**	**VNS-DWOA**	**DGTO**	**Random**
SA	–	**1.0**	*1E-4*	**0.08**	**1.0**	*5E-4*	*7E-04*	**1.0**	*9E-13*
GA	**1.0**	–	*0.01*	**1.0**	**1.0**	*0.04*	*3E-06*	**1.0**	*2E-09*
PSO	*1E-04*	*0.01*	–	**1.0**	**0.38**	**1.0**	*2E-17*	*5E-4*	**0.09**
ASrm	**0.08**	**1.0**	**1.0**	–	**1.0**	**1.0**	*9E-12*	**0.19**	*1E-04*
ACSrm	**1.0**	**1.0**	**0.38**	**1.0**	–	**0.80**	*9E-09*	**1.0**	*1E-06*
DGWO	*5E-4*	*0.04*	**1.0**	**1.0**	**0.80**	–	*2E-16*	*1E-3*	*0.03*
VNS-DWOA	*7E-04*	*3E-06*	*2E-17*	*9E-12*	*9E-09*	*2E-16*	–	*2E-4*	*6E-31*
DGTO	**1.0**	**1.0**	*5E-4*	**0.19**	**1.0**	*1E-3*	*2E-4*	–	*7E-12*
Random	*9E-13*	*2E-09*	**0.09**	*1E-4*	*1E-06*	*0.03*	*6E-31*	*7E-12*	–

Table 5 Friedman test for the comparison between the optimizers

Optimizer	Avg. Ranking	χ^2 statistic	p-value
VNS-DWOA	**1.33**		
SA	3.75		
DGTO	3.98		
GA	4.43		
ACSrm	4.98	167.53	4.22E-32
ASrm	5.45		
DGWO	6.25		
PSO	6.55		
Random	8.28		

Based on the average ranking, the results of Bonferroni-Dunn test (Table 4) indicate that only VNS-DWOA algorithm is able to significantly perform differently than all the other 8 methods with a very low p-value. If we take the highest p-values calculated for VNS-DWOA, which is 2.10^{-4}, this means that the probability of observing such a difference in performance by chance is less than 0.02%, which is a strong indication of the superior performance of VNS-DWOA. Moreover, DGTO, SA, and GA exhibit significantly better performance than PSO, DGWO, and Random Guessing. On the other hand, ASrm and ACSrm only present better results than Random Guessing. PSO is the only algorithm that does not demonstrate a significant difference when compared to random guessing. However, this does not imply that random guessing is superior, but rather that we lack sufficient statistical evidence to reject the null hypothesis for PSO/Random Guessing comparison. The Bonferroni-Dunn test is known for being a conservative test, meaning that it controls the family-wise error rate by adjusting the significance level for multiple comparisons. Therefore, the significant differences found between VNS-DWOA and the other algorithms are reliable and robust.

In order to confirm the best performers, we conduct multiple pair-wise comparisons using the Wilcoxon sum-rank test. Since we want to know which method has the lowest sample of cycle time, we use the alternative hypothesis "less" which means that we are testing whether the first sample is significantly less than the second sample. In this case, the null hypothesis H_0 would be that the median difference between the two samples is not less than zero.

Wilcoxon sum-rank test results, recapitulated in Table 6, confirm the superiority of VNS-DWOA over all other eight methods with very low p-values. Additionally, the test results reveal that SA and DGTO are both on the same level, and they significantly outperform GA, ACSrm, ASrm, PSO, DGWO, and Random Guessing, with p-values less than 0.05, in finding minimized assembly cycle time.

To sum up, the statistical analysis performed on the experimental data using the Freidman Test, Bonferroni-Dunn test, and Wilcoxon sum-rank test indicate that the VNS-DWOA method outperforms all other methods tested. Specifically, the VNS-DWOA method consistently achieved higher quality solutions than the other methods, and its performance remained stable and correct even as it converged to high-quality solutions. These results suggest that the VNS-DWOA method is a reliable and effective optimization technique for the given problem domain. Therefore, it may be a promising approach to consider for future studies or applications in this field.

VNS-DWOA, as a hybrid optimization algorithm, combines the strengths of both Variable Neighborhood Search (VNS) and Whale Optimization Algorithm (WOA). VNS is known for its ability to escape local optima and explore different regions of the search space, while WOA's discrete encircling behavior can efficiently converge to high-quality solutions by re-balancing iteratively the workload taking into account its overall impact on all equipment. In contrast, the statistical study revealed that certain techniques, such as Particle Swarm Optimization (PSO) and Grey Wolves Optimizer (GWO), exhibited poor performance, possibly due to their difficulty in handling discrete variables and executing discrete search operations.

Table 6 The Wilcoxon sum-rank test results for the comparison of the algorithms ($p \leq 0.05$) -the row method better than the column method-

Better than	VNS-DWOA	SA	DGTO	GA	ACSrm	ASrm	DGWO	PSO	Random
VNS-DWOA	–	*1E-10*	*2E-12*	*3E-08*	*3E-11*	*2E-11*	*9E-13*	*1E-11*	*9E-13*
SA		–	**0.20**	**0.09**	*0.03*	*9E-04*	*1E-05*	*3E-06*	*6E-12*
DGTO			–	**0.12**	*9E-03*	*1E-04*	*2E-07*	*1E-06*	*9E-13*
GA				–	**0.12**	*0.01*	*2E-04*	*7E-05*	*6E-10*
ACSrm					–	**0.18**	*3E-03*	*1E-03*	*1E-10*
ASrm						–	**0.07**	*1E-03*	*8E-10*
DGWO							–	**0.12**	*1E-07*
PSO								–	*1E-04*
Random									–

Lastly, the dynamic simulation-based framework used in the study may have also contributed to the superior performance of VNS-DWOA. By incorporating realistic constraints and interactions between different components of the hybrid assembly line, the framework has provided a more realistic and accurate representation of the optimization problem, facilitating the search agents' ability to navigate through their exploration and exploitation processes with greater precision and accuracy.

Conclusion

In conclusion, we presented two recent swarm intelligence-based methods, VNS-DWOA and DGTO, to solve a general version of type-2 hybrid assembly line balancing problem. We have used a discrete event simulator to accurately estimate the cycle time of the assembly line based on a given scenario. The optimization algorithms were tested on a real industrial use case and compared with six other state-of-the-art meta-heuristics as well as random guessing. The performance of the methods was assessed using three different statistical tests on 40 independent runs: Friedman test, Bonferroni-Dunn test and Wilcoxon sum-rank test.

The results obtained from our study clearly demonstrate the superior performance of the VNS-DWOA method compared to all other methods, including the state-of-the-art approaches. In terms of average cycle time, best solutions, and worst solutions, the VNS-DWOA consistently outperforms its counterparts without any exceptions. Remarkably, within just approximately 1 h of execution, the VNS-DWOA method is capable of generating assembly scenarios that exhibit a 7% lower cycle time compared to the scenarios proposed by the process experts. This highlights the remarkable efficiency and effectiveness of the VNS-DWOA method.

Furthermore, our study shows that the DGTO method performs significantly better than several other methods, namely GA, ACSrm, ASrm, PSO, DGWO, and Random Guessing. With an average achieved cycle time of 119 s, the DGTO method outperforms these competitors. However, it falls short of surpassing the performance of the process experts and the VNS-DWOA method, as it has not consistently achieved a lower cycle time on average. These findings have been validated through rigorous statistical analysis, as indicated by the low p-values obtained from the three statistical tests.

The research can be further extended to include the analysis of optimal parameters adjustment to enhance the performance of the proposed algorithms. Furthermore, we plan to incorporate a global cost function that takes into account equipment costs and other relevant factors. Moreover, we aim to develop a continuous learning block that can add the concept of lessons learned to the assembly line design. This will enable the correction of design mistakes made previously and allow for dynamic adjustment of constraints based on the feedback dataset.

Overall, our research provides a valuable contribution to the field of hybrid assembly line design and optimization. Our swarm intelligence-based framework offers an effective and efficient approach to solving assembly line design problems, which has significant implications for the automotive industry and beyond. We believe that the proposed methods can be adapted and extended to various manufacturing and production settings, offering practical and actionable insights for process improvement and optimization.

Author Contributions A. El Houd conceptualized and implemented the proposed optimizers, conducted statistical tests, and authored the main manuscript text and figures. B. Piranda provided supervision, contributed to the development of the simulator, and assisted with manuscript editing. R. De Matos provided supervision, reviewed and edited the manuscript, validated the findings, and secured the testing of the framework on a real industrial use case. J. Bourgeois provided supervision, reviewed and edited the manuscript, validated the findings, and acquired funding for the project.

Funding This work was done as a research collaboration between FORVIA and FEMTO-ST Institute, funded by the Ministry of Higher Education and Research of France and managed by the Association

Nationale de la Recherche et de la Technologie (ANRT) as a part of a CIFRE (N 2021/1302).

Data Availability The data underlying this article, including the developed methods and test data, are available upon request. However, due to confidentiality considerations, detailed information regarding the assembly line and welding process, detailed process durations cannot be publicly disclosed.

Declarations

Conflict of interest All authors declare that they have no conflicts of interest to disclose.

References

Abdollahzadeh, B., Soleimanian Gharehchopogh, F., & Mirjalili, S. (2021). Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems, 36*(10), 5887–5958.

Adnan, A. N., Arbaai, N. A., & Ismail, A. (2016). Improvement of overall efficiency of production line by using line balancing. *ARPN Journal of Engineering and Applied Sciences, 11*(2), 7752–7758.

Al-Betar, M. A., Awadallah, M. A., Doush, I. A., Alomari, O. A., Abasi, A. K., Makhadmeh, S. N., and Alyasseri, Z. A. A. (2022). Boosting the training of neural networks through hybrid metaheuristics. *Cluster Computing* , 1–23.

Alorf, A. (2023). A survey of recently developed metaheuristics and their comparative analysis. *Engineering Applications of Artificial Intelligence, 117*, 105622.

Alwajih, R., Jadid, A., & Al Hussain, H. (2022). Hybrid binary whale with harris hawks for feature selection. *Neural Computing and Applications, 07*, 1–19.

Becerra-Rozas, M., Cisternas-Caneo, F., Crawford, B., Soto, R., García, J., Astorga, G., & Palma, W. (2022). Embedded learning approaches in the whale optimizer to solve coverage combinatorial problems. *Mathematics, 10*(23), 4529.

Bukchin, Y., & Raviv, T. (2018). Constraint programming for solving various assembly line balancing problems. *Omega, 78*, 57–68.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications, 45*, 41–51.

Chen, J., Jia, X., & He, Q. (2023). A novel bi-level multi-objective genetic algorithm for integrated assembly line balancing and part feeding problem. *International Journal of Production Research, 61*(2), 580–603.

Chen, J., Ling, F., Zhang, Y., You, T., Liu, Y., & Du, X. (2022). Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system. *Swarm and Evolutionary Computation, 69*, 101005.

Dai, Y., Khandelwal, M., Qiu, Y., Zhou, J., Monjezi, M., and Yang, P. (2022). A hybrid metaheuristic approach using random forest and particle swarm optimization to study and evaluate backbreak in open-pit blasting. *Neural Computing and Applications*, 1–16.

Derrac, J., García, S., Molina, D., & Herrera F, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1, 1*, 3–18.

Dorigo, M., and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, Vol. 2, IEEE, pp. 1470–1477.

Garren, S. T., and Davenport, G. H. (2022). *Using Kurtosis for Selecting One-Sample T-Test or Wilcoxon Signed-Rank Test*, vol. 41. pp. 46–55.

Giordani da Silveira, W., Pinheiro de Lima, E., Gouvea da Costa, S. E., & Deschamps, F. (2017). Guidelines for hoshin kanri implementation: Development and discussion. *Production Planning & Control, 28*(10), 843–859.

Hansen, P., Mladenović, N., Todosijević, R., & Hanafi, S. (2017). Variable neighborhood search: Basics and variants. *EURO Journal on Computational Optimization, 5*(3), 423–454.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S., & Al-Atabany, W. (2022). Honey badger algorithm: New metaheuristic algorithm for solving optimization problems. *Mathematics and Computers in Simulation, 192*, 84–110.

Hosseini, S., & Al Khaled, A. (2014). A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research. *Applied Soft Computing, 24*, 1078–1094.

Kashiwado, Y., Kimoto, Y., Sawabe, T., Irino, K., Nakano, S., Hiura, J., Wang, Q., Kawano, S., Ayano, M., Mitoma, H., et al. (2023). Antibody response to sars-cov-2 mrna vaccines in patients with rheumatic diseases in japan: Interim analysis of a multicentre cohort study. *Modern Rheumatology, 33*(2), 367–372.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications, 80*, 8091–8126.

Kivi, M. E., & Majidnezhad, V. (2022). A novel swarm intelligence algorithm inspired by the grazing of sheep. *Journal of Ambient Intelligence and Humanized Computing, 13*(2), 1201–1213.

Kumar, N., & Mahto, D. (2013). Assembly line balancing: A review of developments and trends in approach to industrial application. *Global Journal of Researches in Engineering Industrial Engineering, 13*(2), 29–50.

Mehmet, H., et al. (2020). Constraint programming model for resource-constrained assembly line balancing problem. *Soft Computing, 24*(7), 5367–5375.

Mirjalili, S. (2020). Special issue on"real-world optimization problems and meta-heuristics". *Neural Computing Applications, 32*, 11965–11966.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

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

Mutingi, M., Mbohwa, C., & Mutingi, M. (2017). *Mbohwa, C* (pp. 183–197). Assembly Line Balancing: A Hybrid Grouping Genetic Algorithm Approach. Springer.

Pan, J.-S., Zhang, L.-G., Wang, R.-B., Snášel, V., & Chu, S.-C. (2022). Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems. *Mathematics and Computers in Simulation, 202*, 343–373.

Piranda, B., Gautam, I., Meyer, J., El Houd, A., and Bourgeois, J. (2022). Manufactsim: Manufacturing line simulation using heterogeneous distributed robots. In *International Conference on Advanced Information Networking and Applications*, Springer, pp. 130–140.

Pornsing, C., Sangkhiew, N., Sakonwittayanon, P., Jomtong, P., and Ohmori, S. (2022). A new discretization technique for enhancing discrete particle swarm optimization's performance. *Science & Technology Asia,* pp. 204–215.

Prashar, A. (2023). Production planning and control in industry 4.0 environment: A morphological analysis of literature and research agenda. *Journal of Intelligent Manufacturing, 34*(6), 2513–2528.

Rajpurohit, J., and Sharma, T. K. (2022). A hybrid metaheuristic for transmission tower design optimization. In *Soft Computing: Theo-*

ries and Applications: Proceedings of SoCTA 2021 (pp. 857–868). Springer.

Rodríguez, N., Gupta, A., Zabala, P. L., & Cabrera-Guerrero, G. (2018). Optimization algorithms combining (meta) heuristics and mathematical programming and its application in engineering.

Salhi, S., and Thompson, J. (2022). An overview of heuristics and metaheuristics. *The Palgrave Handbook of Operations Research* (pp. 353–403).

Sharma, I., Kumar, V., and Sharma, S. (2022). A comprehensive survey on grey wolf optimization. *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 15*(3), 323–333.

Singh, G., & Singh, A. (2023). Extension of particle swarm optimization algorithm for solving two-level time minimization transportation problem. *Mathematics and Computers in Simulation, 204*, 727–742.

Stan, L., Nicolescu, A. F., Pupăză, C., & Jiga, G. (2023). Digital twin and web services for robotic deburring in intelligent manufacturing. *Journal of Intelligent Manufacturing, 34*(6), 2765–2781.

Suwannarongsri, S., and Puangdownreong, D. (2009). Metaheuristic approach to assembly line balancing. *WSEAS Transactions on Systems, Vol 8*.

Tang, Q., Meng, K., Cheng, L., & Zhang, Z. (2022). An improved multiobjective multifactorial evolutionary algorithm for assembly line balancing problem considering regular production and preventive maintenance scenarios. *Swarm and Evolutionary Computation, 68*, 101021.

Tliba, K., Diallo, T. M., Penas, O., Ben Khalifa, R., Ben Yahia, N., & Choley, J.-Y. (2022). Digital twin-driven dynamic scheduling of a hybrid flow shop. *Journal of Intelligent Manufacturing, 34*(5), 1–26.

Vincent, F. Y., Susanto, H., Jodiawan, P., Ho, T.-W., Lin, S.-W., & Huang, Y.-T. (2022). A simulated annealing algorithm for the vehicle routing problem with parcel lockers. *IEEE Access, 10*, 20764–20782.

Walter, R., Schulze, P., et al. (2021). Salsa: Combining branch-and-bound with dynamic programming to smoothen workloads in simple assembly line balancing. *European Journal of Operational Research, 295*(3), 857–873.

Wang, W., Wang, Q., Zhong, R., Chen, L., & Shi, X. (2023). Stacking sequence optimization of arbitrary quadrilateral laminated plates for maximum fundamental frequency by hybrid whale optimization algorithm. *Composite Structures, 310*, 116764.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

Xie, L., Han, T., Zhou, H., Zhang, Z.-R., Han, B., & Tang, A. (2021). Tuna swarm optimization: A novel swarm-based metaheuristic algorithm for global optimization. *Computational intelligence and Neuroscience, 2021*, 1–22.

Yadav, A., & Agrawal, S. (2022). Mathematical model for robotic two-sided assembly line balancing problem with zoning constraints. *International Journal of System Assurance Engineering and Management, 13*(1), 395–408.

Yao, P., & Gupta, S. M. Cat swarm optimization algorithm for solving multi-objective sequence-dependent u-shaped disassembly line balancing problem.

Yu, D., Zhang, X., Tian, G., Jiang, Z., Liu, Z., Qiang, T., & Zhan, C. (2022). Disassembly sequence planning for green remanufacturing using an improved whale optimisation algorithm. *Processes, 10*(10), 1998.

Zhou, Y., Liu, X., Hu, S., Wang, Y., & Yin, M. (2022). Combining max-min ant system with effective local search for solving the maximum set k-covering problem. *Knowledge-Based Systems, 239*, 108000.