A Synergistic Genetic and Particle Swarm Optimization Approach for Multi-Objective Process Parameter Optimization in Reconfigurable Assembly Lines

Qin Gong*, Li Wang
Tieling Normal College, Tieling Liaoning Province, 112000
E-mail: kzuooqfwcl0@hotmail.com, ektmvvbut381@hotmail.com
*Corresponding author

Keywords: reconfigurable flexible assembly lines, process parameter optimization, synergistic genetic algorithm and particle swarm optimization, intelligent manufacturing systems

Received: June 5, 2025

With the rapid advancement of intelligent manufacturing technologies, Reconfigurable Flexible Assembly Lines (RFALs) have emerged as a promising solution to enhance production flexibility and efficiency. However, the process parameter optimization for RFALs, particularly in assembly line balancing, presents a complex NP-hard combinatorial optimization problem. This study aims to address the process parameter optimization in RFALs by considering multiple critical performance metrics, including production cycle time, tool replacement time, and assembly unit cost. A mathematical model is formulated to describe the optimization problem, clearly defining the objectives and constraints. Based on this model, a novel Synergistic Genetic Algorithm and Particle Swarm Optimization (SGAPSO) is proposed. The SGAPSO algorithm effectively combines the global exploration capability of Genetic Algorithms (GA) and the local exploitation and fast convergence characteristics of Particle Swarm Optimization (PSO). It employs a task sequence-based encoding method. In the GA phase, population evolution is driven by selection, crossover, mutation, and elitism. In the PSO phase, particle positions are decoded using the Smallest Position Value (SPV) rule, and iterations are optimized through standard velocity and position update equations. The key synergistic mechanism proved by ablation study involves periodically guiding the PSO population with elite individuals from GA and enhancing the GA population with superior solutions found by PSO. Experimental validation on standard benchmark problems and an industrial case study of pressure-reducing valve assembly shows that the SGAPSO algorithm outperforms standalone GA and PSO in terms of solution quality, convergence speed, and solution stability.

Povzetek: SGAPSO hibridni algoritem združuje prednosti GA in PSO za bolj kvalitetno optimizacijo parametrov v fleksibilnih linijah.

1 Introduction

The rapid advancement of intelligent manufacturing technologies is driving manufacturing systems toward greater flexibility, efficiency, and intelligence [1, 2]. In this transformative era, Reconfigurable Flexible Assembly Lines (RFALs) have garnered significant attention due to their inherent capability to adapt to diverse product portfolios and dynamic production schedules [3]. RFALs, characterized by their modular architectures and rapid reconfiguration capabilities, offer modern manufacturing enterprises a distinct competitive advantage [6, 7]. However, to fully unleash the potential of RFALs, the optimization of their process parameters is paramount. This optimization is crucial for maximizing production efficiency, minimizing operational costs, and achieving comprehensive enhancements in overall system performance [8, 9].

Process parameter optimization represents a core challenge within intelligent manufacturing systems, involving a multitude of complex variables and stringent constraints [11, 12]. Researchers like Samouei et al. have extensively explored the intricacies of assembly line balancing, highlighting the significance of optimizing task assignments to workstations and minimizing tool changeover times [13, 14]. They emphasize that these factors are not only critical for enhancing production rates but also for achieving cost reductions in RFALs. Tran et al. further corroborates this by investigating how simulation and metaheuristics can be employed to tackle the assembly line balancing problem, underscoring the need for efficient algorithms that can navigate the complexities of RFALs [15]. These optimization tasks, often categorized as NP-hard combinatorial optimization problems, pose significant challenges due to their inherent complexity and the vast solution spaces they encompass [16]. Traditional optimization methodologies, while effective in simpler scenarios, often struggle to identify effective solutions within practical timeframes for such intricate problems [17]. Consequently, the research and development of highly efficient optimization algorithms

have become imperative. Studies by Mondal et al. and Villalobos et al. have made strides in addressing these challenges by proposing matheuristic approaches and new mathematical models for assembly line balancing, demonstrating the ongoing efforts to overcome the limitations of traditional methods and enhance the optimization process for RFALs [18, 19].

In recent years, metaheuristic algorithms, particularly Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), have demonstrated considerable potential in tackling complex optimization problems, especially in parameter optimization. Genetic Algorithms are renowned for their robust global search capabilities, drawing inspiration from the mechanisms of natural selection and evolution. As detailed by Katoch et al. in their review, GAs explore vast solution spaces by simulating evolutionary processes, making them highly suitable for finding optimal combinations of numerous parameters in complex systems [20]. For instance, in hyperparameter tuning for machine learning models or parameter calibration in engineering design, GAs can effectively avoid local optima to identify globally superior parameter configurations. Particle Swarm Optimization (PSO) is favored for its rapid convergence characteristics and efficient local search capabilities, mimicking collective intelligence behaviors such as bird flocking or fish schooling. Gad's systematic review highlights the effectiveness of PSO in diverse applications, including various parameter optimization scenarios [21]. PSO searches parameter spaces through collaboration and information sharing among particles in a swarm, enabling quick identification of promising regions. For example, in tuning control system parameters or optimizing complex functions, PSO can approximate optimal solutions by iteratively adjusting particle positions. Many studies have also explored hybrid variants of PSO, for instance, by integrating it with other algorithms [22] or by incorporating advanced mechanisms such as adaptive strategies and novel learning frameworks, to further enhance its performance in multi-modal and high-dimensional parameter optimization problems. Overall, both GA and PSO offer powerful and flexible tools for addressing challenging parameter optimization problems.

Nevertheless, individual metaheuristics often exhibit inherent limitations. For instance, GAs may experience a slowdown in convergence speed during later search stages [23]. Yang et al. discusses how GAs can sometimes struggle with convergence rates, particularly as the search space becomes more complex [24]. Similarly, while PSO can be effective in many scenarios, it can be susceptible to premature convergence to local optima, especially when applied to complex, multimodal problems. Wang et al.'s work highlights how PSO might become trapped in local optima, leading to suboptimal solutions in such cases [25].

To overcome these individual shortcomings, researchers have increasingly explored synergistic optimization approaches that combine the strengths of GA and PSO. Such hybrid methods aim to achieve more efficient and effective optimization performance by integrating the complementary advantages of both algorithms [26]. Through the strategic amalgamation of GA's global exploration prowess and PSO's local exploitation capabilities, synergistic optimization techniques can navigate complex solution spaces more adeptly, thereby yielding superior solutions for process parameter optimization problems in RFALs [27]. For example, Premalatha et al. proposed a hybrid PSO and GA model that outperforms standard PSO in solving global optimization problems, reducing stagnation and premature convergence on suboptimal solutions [28].

The central objective of this research is to investigate the application of GA and PSO synergy to the process parameter optimization problem within intelligent manufacturing systems. By first constructing a precise mathematical model to delineate the intricacies of process parameter optimization in RFALs, and subsequently designing an effective synergistic optimization algorithm, termed Synergistic Genetic Algorithm and Particle Swarm Optimization (SGAPSO), this study aims to provide an innovative solution framework for complex optimization challenges prevalent in intelligent manufacturing. Through rigorous experimental validation across standard benchmark problems and industrial case studies, this research will demonstrate the efficacy and superiority of the SGAPSO algorithm in addressing practical industrial problems, thereby offering new insights and methodologies for the optimization of intelligent manufacturing systems. For instance, experimental results on benchmark problems such as Jackson and Buxey instances, as well as the industrial case study of pressure-reducing valve assembly, are expected to show significant improvements in solution quality and convergence speed compared to traditional methods.

The paper is structured as follows: Chapter 2 provides a detailed exposition of the process parameter optimization problem in RFALs and constructs the corresponding mathematical model. Chapter 3 delves deeply into the design and implementation specifics of the SGAPSO algorithm. Subsequently, Chapter 4 presents a comprehensive series of experiments conducted to evaluate the performance of SGAPSO, comparing it with other relevant algorithms to validate its effectiveness and potential in solving process parameter optimization problems within the domain of intelligent manufacturing.

2 Problem description and mathematical model

2.1 Problem description: optimizing process parameters in reconfigurable flexible assembly lines

In the landscape of intelligent manufacturing, the optimization of process parameters stands as a cornerstone for achieving heightened efficiency, cost reduction, and superior system-wide performance. Reconfigurable Flexible Assembly Lines (RFALs) are pivotal in this context, providing the necessary adaptability to manage diverse product portfolios and dynamic production schedules. The central challenge within such advanced

systems is the effective optimization of parameters related to line configuration and balancing. This optimization aims to maximize production throughput and the utilization of resources, while concurrently minimizing operational inefficiencies and bottlenecks. This research specifically targets the optimization of critical process parameters integral to the functioning of an RFAL environment.

An RFAL's architecture is inherently modular, typically featuring centralized libraries for tools and fixtures, alongside reconfigurable assembly cells. These cells, as the foundational elements of the assembly line, can be rapidly adapted with different tooling to perform a wide array of assembly operations, and their physical layout can be swiftly reconfigured to meet new product assembly demands. This inherent flexibility, while advantageous, introduces significant complexity into the process parameter optimization task. Key parameters requiring careful optimization include the overall production cycle time, the non-productive time incurred during tool and fixture changeovers, and the economic considerations tied to the deployment and operation of assembly units. Strategic optimization of these parameters is fundamental to unlocking the full operational and economic benefits of RFALs. Figure 1 illustrates a schematic of a Reconfigurable Flexible Assembly Line (RFAL) System. The numbered components highlight various stations within the RFAL: (1) represents the initial processing station where the base components are prepared; (2) shows the assembly station where the main assembly operations are conducted; (3) indicates the inspection and quality control station; (4) is the packaging area where the assembled products are prepared for shipment; and (5) represents the final output station where the finished products are dispatched. This visualization provides a clear understanding of how the RFAL's modular design allows for both flexibility in task execution and adaptability to changing production requirements.

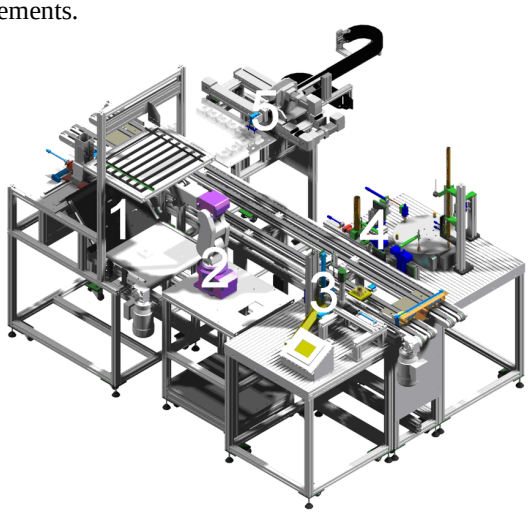

Figure 1: Schematic of a reconfigurable flexible assembly line (RFAL) system

The specific problem this research addresses is the balancing of an RFAL, a task recognized as an NP-hard combinatorial optimization problem. The core objective is the optimization of task assignments to workstations. This involves distributing assembly tasks in such a way that a predefined set of performance metrics—which are themselves functions of various process parameters—is optimized. This assignment must rigorously adhere to established precedence constraints among tasks and other operational limitations inherent in the manufacturing environment. This chapter focuses on developing a precise mathematical model to formally represent this multifaceted process parameter optimization problem.

2.2 Mathematical model for process parameter optimization

A robust and comprehensive mathematical model is indispensable for devising effective strategies for process parameter optimization. Such a model must accurately encapsulate the optimization objectives, the decision variables that represent adjustable process parameters, and the constraints that define the feasible operational space of the RFAL balancing problem.

(1) Foundational assumptions for modeling parameter optimization

The mathematical model for parameter optimization is constructed upon several foundational assumptions, derived from the operational characteristics of RFALs. It is assumed that each assembly operation (task) is an indivisible unit, and a workstation is equivalent to a single reconfigurable assembly unit. A fundamental requirement is the complete assignment of all defined assembly tasks, with each task being uniquely allocated to one workstation. Crucially, these assignments must respect a predefined precedence diagram, which governs the permissible sequence of operations—a key constraint in the optimization process. Task processing times are considered deterministic and constant. The total operational time at any given workstation, a critical parameter, must not exceed the overall production cycle time of the assembly line. Following the optimization of task distribution, it is assumed that different workstations will have unique configurations of assigned tasks. Finally, for model simplification, part loading/unloading times and the initial line stabilization period are deemed negligible.

(2) Objective function: A multi-faceted approach to parameter optimization

The optimization of process parameters in an RFAL context inherently involves addressing multiple, often conflicting, objectives. This research considers the simultaneous minimization of three critical parameters: the production cycle time (CT), the total tool replacement time (RT), and the aggregate cost of assembly units (ST). To handle this multi-objective nature, a weighted sum approach is adopted. This method consolidates the individual objectives into a single composite objective function, aiming to find a balanced solution that reflects the desired trade-offs in parameter optimization:

$$\min Z = w_1 \cdot CT + w_2 \cdot RT + w_3 \cdot ST \quad (1)$$

In this equation, w_1, w_2, and w_3 represent non-negative weight coefficients. These coefficients are crucial as they signify the relative importance assigned by the decision-maker to minimizing the production cycle time (CT), the tool replacement time (RT), and the assembly unit costs (ST), respectively, within the overall parameter optimization strategy. The sum of these weights can be normalized, for instance, to sum to one ($w_1 + w_2 + w_3 = 1$), which helps in ensuring consistent scaling and interpretation of their relative impacts on the objective function.

(3) Core process parameters, decision variables, and constraints in the optimization model

The mathematical model for parameter optimization is further defined by specific equations and constraints governing the key process parameters. These elements collectively define the search space and the criteria for evaluating potential solutions.

The production cycle time, denoted as CT, is a paramount performance indicator in assembly line operations; its minimization is a primary goal of parameter optimization. CT is typically defined as the maximum operational time accumulated at any single workstation across the entire assembly line. A shorter CT directly translates to higher production efficiency and increased throughput. The model for CT is expressed as:

$$CT = \max_{1 \le k \le m} \left(\sum_{i=1}^{n} t_i x_{ik} \right) \tag{2}$$

This optimization is subject to several critical constraints that define the feasible solutions for the parameter optimization. The precedence among tasks, which dictates the order in which operations must be performed, is enforced by the following constraint:

$$\sum_{k=1}^{m} k \cdot x_{ik} - \sum_{k=1}^{m} k \cdot x_{jk} \le 0, \forall i \in \mathrm{pre}(j); \forall j \tag{3}$$

This formulation ensures that if task i is a direct predecessor of task j, task i is assigned to a workstation whose numerical index is less than or equal to that of the workstation assigned to task j. It's important to note that this implies a generally sequential flow of workpieces through workstations that are indexed in a numerically ordered fashion. Alternative or supplementary formulations might be necessary for scenarios involving more complex parallel processing capabilities across workstations, while still rigorously respecting the fundamental technological precedence relationships.

Furthermore, each individual assembly task, indexed by i, must be assigned to precisely one workstation. This is a fundamental constraint for a valid task allocation in the parameter optimization process and is represented as:

$$\sum_{k=1}^{m} x_{ik} = 1, \forall i \tag{4}$$

Concurrently, each workstation, indexed by k, must be assigned at least one task, ensuring that all deployed workstations are utilized. The upper limit for tasks per workstation is naturally the total number of tasks, n. This is captured by:

$$1 \le \sum_{i=1}^{n} x_{ik} \le n, \forall k \tag{5}$$

The primary decision variable in this parameter optimization context is x_{ik}, which is a binary variable. It takes a value of 1 if task i is assigned to workstation k, and 0 otherwise. This can be formally written as:

$$x_{ik} = \begin{cases} 1, & \text{if task } i \text{ is assigned to workstation } k \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

And it must satisfy

$$x_{ik} \in \{0,1\}, \forall i, k \tag{7}$$

In these formulations, the indices i and j are used for the assembly tasks, ranging from 1 to n, where n is the total number of distinct assembly tasks. The index k denotes the workstations, ranging from 1 to m, where m represents the total number of available or configured workstations.

The term t_i signifies the known processing time required for task i, which is an essential input parameter for the model. Finally, pre(j) denotes the set of tasks that are immediate predecessors to task j, as defined by the assembly process's technological sequence. The parameter m, the total number of workstations, can itself be a decision variable in certain types of balancing problems (e.g., minimizing m for a given CT), or a fixed input parameter (e.g., minimizing CT for a fixed m) significantly influencing the parameter optimization approach.

Minimizing non-productive time, such as that incurred during tool changes, is another vital aspect of process parameter optimization in flexible assembly environments. When multiple tasks requiring different tools are allocated to the same workstation, these changeovers consume valuable operational time, thereby reducing overall line efficiency. The total tool replacement time, RT, is modeled as follows:

$$RT = T_{RTI} \cdot \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1, j \ne i}^{n} x_{ik} \cdot x_{jk} \cdot \mathrm{ToolDiff}(i,j) \cdot \mathrm{Adjacent}(i,j,k) \tag{8}$$

In this equation, T_{RTI} represents the fixed time required to perform a single tool replacement, which is treated as a known input parameter for the model. The term ToolDiff (i, j) is a binary variable that plays a central role in the parameter optimization related to tooling strategy. It is defined as 1 if assembly task i and assembly task j necessitate the use of different tools, and 0 if they can be performed with the same tool:

$$\mathrm{ToolDiff}(i,j) = \begin{cases} 1, & \text{if tool for task } i \ne \text{ tool for task } j \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

The variable Adjacent (i, j, k) is also a binary variable, crucial for accurately modeling the sequencedependent setup times. It takes the value 1 if task j is processed immediately after task i within the sequence of tasks assigned to the same workstation k. The determination of this intra-workstation task sequence is itself an important component of the overall parameter optimization problem. If S_k denotes the ordered sequence of tasks assigned to workstation k, then Adjacent (i, j, k)=1, if tasks i and j are both assigned to workstation k and task j immediately follows task i in the sequence S_k. Otherwise Adjacent (i, j, k) =0.

It is worth noting that the original PDF's formulation for tool replacement time was $\mathrm{RT} = RTI \cdot \sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ik} x_{jk} \mathrm{Tool}(i,j) P(i,j)$. This structure implies that the term $P(i, j)$ (indicating that task j immediately follows task) combined with the product $x_{ik}x_{jk}$ (ensuring both tasks are assigned to the same station k) effectively captures this notion of adjacency within a workstation. The term Tool (i, j) in that context corresponds to ToolDiff (i, j), indicating whether the tools required for the two tasks are different. It is important to acknowledge that the binary ToolDiff variable represents a simplification of real-world tool compatibility. In practice, tool compatibility is not always a strict binary condition; it can be multi-modal (e.g., a task can be performed by several different tools with varying efficiency) or fuzzy (e.g., one tool might be partially compatible with a task). While our current model provides a tractable framework for optimization, future research could extend this by incorporating more advanced modeling techniques. For instance, a fuzzy compatibility index or a cost matrix reflecting the varying degrees of efficiency for different task-tool pairings could provide a more nuanced and realistic representation, albeit at the cost of increased model complexity.

The economic efficiency of the assembly line is a key consideration in comprehensive parameter optimization, and it is directly influenced by the number of assembly units (workstations) deployed and operated. The total assembly unit cost, ST, can be modeled as a linear function of the number of active workstations:

$$ST = C_{AU} \cdot m \tag{10}$$

Here C_{AU} represents the average cost associated with deploying and operating a single assembly unit or workstation; this is typically an input parameter based on capital investment and operational expenditures. The variable m denotes the total number of assembly units utilized in the configured assembly line. The optimization of this parameter m is particularly critical in what is known as Type II assembly line balancing problems, where the primary objective is to minimize the number of workstations (m) required to achieve a given production cycle time (CT). Conversely, if m is a fixed parameter (as in Type I balancing problems, where the goal is to minimize CT for a predetermined m), then the ST term, as formulated, would become a constant in the objective function. However, its inclusion suggests a broader scope where m might be a decision variable, or where minimizing its use (even if chosen from a set of available units) is a strategic goal of the parameter optimization. This could also extend to scenarios involving the selection among different types of workstations that might have varying costs, making the cost component an active part of the optimization.

This mathematical model, with its defined objectives, variables, and constraints, establishes a quantitative and structured framework for tackling the optimization of crucial process parameters within the complex environment of a reconfigurable flexible assembly line. The subsequent chapters of this research will delve into the application of synergistic heuristic algorithms, specifically focusing on Genetic Algorithms and Particle Swarm Optimization, to derive effective and near-optimal solutions for this intricate parameter optimization challenge.

3 Synergistic genetic algorithm and particle swarm optimization (SGAPSO) for process parameter optimization

To effectively address the complex combinatorial optimization problem posed by the process parameter optimization model detailed in Chapter 2, this research introduces a novel hybrid metaheuristic approach: the Synergistic Genetic Algorithm and Particle Swarm Optimization (SGAPSO).

3.1 Introduction and fundamental principles of SGAPSO

Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) are powerful metaheuristics, yet they possess complementary limitations. GAs excels at global exploration but can converge slowly, while PSO offers rapid convergence but risks premature stagnation in local optima, particularly in complex, multimodal search spaces.

The core philosophy of the SGAPSO algorithm is to synergistically hybridize these methods, amplifying their respective strengths while mitigating their drawbacks. Within the SGAPSO framework, the GA conducts broad, global exploration to identify promising regions of the solution space, thereby maintaining population diversity. Subsequently, the PSO component performs intensive, fine-grained local searches within these regions to rapidly enhance solution precision and quality. This structured coordination creates a balanced search process, combining the breadth of GA's exploration with the depth and speed of PSO's exploitation to achieve superior performance in solving the complex RFAL optimization problem.

3.2 SGAPSO framework construction

The architectural design of the SGAPSO algorithm revolves around several key components: the representation of solutions, the initialization of the

populations, and the crucial fitness evaluation mechanism that guides the search.

Solution encoding and population initialization: Addressing the assembly line balancing problem, a solution typically involves defining both the sequence of tasks and their assignment to specific workstations. Within the SGAPSO framework, an individual—whether it's a chromosome in the GA phase or a particle's position in the PSO phase-is encoded primarily based on a task sequence. For instance, a solution for an assembly line with n tasks can be represented as a permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, where each π_j is a unique task identifier. This task-based permutation encoding is intuitive and facilitates the subsequent decoding process. The decoding step involves translating this task sequence, along with constraints such as the production cycle time, into a concrete assignment of tasks to workstations. The initial population for the GA, and potentially for the PSO, is generated randomly. However, a critical constraint is that all initial solutions must be feasible, particularly respecting the precedence relationships among tasks. This feasibility is typically ensured during the generation process by only adding a task to the sequence if all its prerequisite tasks have already been included.

Fitness evaluation: The direction and effectiveness of the algorithm's search are dictated by the fitness function. For each individual (i.e., each encoded task sequence), a decoding procedure is first applied to convert it into a specific task-to-workstation assignment plan. Based on this assignment, the key performance indicators-production cycle time (CT), total tool replacement time (RT), and total assembly unit cost (ST)-are calculated. Subsequently, the multi-objective function $Z = w_1.\, CT + w_2 \cdot RT + w_3 \cdot ST$, is computed to determine the overall cost or objective value of the solution. Since SGAPSO, like most evolutionary algorithms, is typically formulated to maximize fitness, and the objective function Z represents a minimization problem, the fitness function, Fitness, can be defined as the reciprocal of Z (e.g., Fitness= $1/Z$). To prevent numerical issues if Z is zero or very small, appropriate scaling transformations or penalty methods for handling infeasible solutions (if any arise despite initial feasibility checks) can be incorporated into the fitness calculation.

3.3 Genetic algorithm (GA) phase in SGAPSO

The GA phase within SGAPSO is primarily responsible for global exploration of the solution space and for maintaining diversity within the population of candidate solutions. This phase employs core genetic operators: selection, crossover, and mutation.

Selection: The selection operator determines which individuals from the current population will be chosen to reproduce and contribute their genetic material to the next generation. This choice is based on their fitness values. A commonly used method is Roulette Wheel Selection, where the probability of an individual being selected is directly proportional to its fitness relative to the total fitness of the population. Individuals with higher fitness values thus have a greater chance of being selected, promoting the propagation of desirable traits (i.e., better task sequences leading to lower objective function values).

Crossover: The crossover operator is designed to combine the characteristics of two parent individuals to create new offspring, potentially leading to solutions that inherit beneficial traits from both parents. For task sequence-based encoding, specialized crossover operators are necessary to ensure that the offspring remain feasible, particularly with respect to task precedence constraints. Examples include the Partially Mapped Crossover (PMX), Order Crossover (OX), or other crossover techniques specifically adapted for permutation-based problems like assembly line balancing. Crossover is applied with a predefined probability, P_C.

Mutation: The mutation operator introduces small, random changes to an individual's genetic makeup (task sequence) with a relatively low probability, P_m. Its purpose is to introduce new genetic material into the population, thereby increasing diversity and providing a mechanism to escape local optima. For task sequence encoding, mutation operators might involve swapping two randomly selected tasks in the sequence (Swap Mutation) or reversing the order of tasks within a randomly selected subsequence (Inversion Mutation). A crucial aspect of mutation is that the resulting sequence must still satisfy all precedence constraints. If a mutation operation violates these constraints, the mutated sequence must be repaired, or the mutation must be re-applied or discarded.

Elitism strategy: To prevent the loss of the best solutions found so far during the evolutionary process, an elitism strategy is commonly incorporated. This involves directly copying one or more of the fittest individuals from the current generation to the next generation, without subjecting them to crossover or mutation. This ensures that the quality of the best solution in the population does not degrade over generations.

3.4 Particle swarm optimization (PSO) phase in SGAPSO

The PSO phase in SGAPSO focuses on intensive local search and rapid refinement of solutions within promising regions of the search space, often those identified or seeded by the GA phase. The core of PSO lies in its particle position and velocity update mechanisms.

Particle representation and decoding: To integrate PSO's continuous search mechanism with the discrete, permutation-based nature of the assembly line balancing problem (and to maintain compatibility with the GA's task sequence encoding), an indirect encoding scheme is often employed for the PSO particles. Each particle i is represented by a position vector $\mathbf{X}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ in an n-dimensional continuous real-valued space, where n is the total number of assembly tasks. This continuous position vector is then decoded into a discrete task sequence using a rule such as the Smallest Position Value

(SPV). The SPV rule works by sorting the components of the particle's position vector $\mathbf{X}_i$ in ascending order. The permutation of the original indices (task numbers) corresponding to this sorted order of values forms the decoded task sequence. For example, if $\mathbf{X}_i = (0.5,0.1,0.8)$ corresponds to tasks $T1, T2$, and $T3$ respectively, sorting the values gives $(0.1,0.5,0.8)$. The associated task sequence would then be $(T2, T1, T3)$. This decoded task sequence is subsequently used for workstation assignment and fitness evaluation, similar to how GA individuals are processed.

Velocity update equation: The velocity of each particle i in each dimension d, denoted as v_{id}, is updated at each iteration $t + 1$ based on its current velocity, its own best-known position, and the global best-known position in the swarm. The standard velocity update equation is:

$$v_{id}(t+1) = \omega \cdot v_{id}(t) + c_1 \cdot r_1(t) \cdot \left(\text{pbest}_{id}(t) - x_{id}(t) \right) + c_2 \cdot r_2(t) \cdot \left(\text{gbest}_{d}(t) - x_{id}(t) \right) \quad (11)$$

In this equation, ω is the inertia weight, a parameter that controls the influence of the particle's previous velocity on its current one, thereby balancing global exploration and local exploitation. The terms c_1 and c_2 are acceleration coefficients (learning factors) that weight the influence of the personal best position $pbest_{id}(t)$ and the global best position $gbest_d(t)$ respectively. $r_1(t)$ and $r_2(t)$ are random numbers uniformly distributed in the range [0,1], introducing a stochastic element to the search. $x_{id}(t)$ is the particle's current position in dimension d at iteration t. To prevent particles from moving too erratically or leaving the relevant search area, the velocity v_{id} is often clamped to a predefined maximum value, $V_{\max}$.

Position update equation: After updating its velocity, each particle i updates its position x_{id} in dimension d for the next iteration $t + 1$ according to the following simple equation:

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (12)$$

The newly updated continuous position vector $\mathbf{X}_i(t + 1)$ is then decoded back into a task permutation using the SPV rule, followed by the standard decoding and fitness evaluation procedures. Similar to velocity, particle positions may also be constrained to lie within predefined boundaries $[X_{\min}, X_{\max}]$ to keep the search focused.

Personal and global best updates: During each iteration of the PSO, after a particle moves to a new position and its fitness is evaluated, its personal best position $pbest_i$ is updated. If the fitness of the particle's current position is better than the fitness of its $pbest_i$, then $pbest_i$ is set to the current position. Simultaneously, the global best position (gbest) for the entire swarm is updated. If any particle's $pbest_i$ has a fitness better than that of the current gbest, then gbest is updated to that particle's $pbest_i$.

3.5 Synergistic mechanism between GA and PSO in SGAPSO

The efficacy of the SGAPSO algorithm hinges critically on the design of an effective synergistic mechanism that facilitates information exchange and coordinated operation between the GA and PSO components. This synergy is intended to ensure that the complementary strengths of both algorithms are fully exploited. This research proposes a periodic, elite-guided coordination strategy to manage the interaction between GA and PSO. The strategy unfolds in a structured manner:

- **Initial global exploration by GA:** The SGAPSO algorithm commences by executing the GA phase for a specified number of generations. During this phase, the GA employs its selection, crossover, and mutation operators to conduct a broad exploration of the entire solution space. The primary objectives here are to identify potentially promising regions that may contain high-quality solutions and to maintain a diverse

Algorithm 1: SGAPSO for Process Parameter Optimization

Input: RFAL problem parameters, GA parameters, PSO parameters, Synergy parameters
Output: Optimized task assignment solution (Gbest_overall)
Output: Optimized task assignment solution (Gbest_overall)

1. Initialize GA population P_GA with random feasible task sequences and evaluate their fitness. Set Gbest_overall to the best solution in P_GA.
2. For gen_GA = 1 to Max_Gen_GA do
3. GA Phase:
4. Perform selection (e.g., Roulette Wheel Selection) to choose parents from P_GA.
5. Apply crossover (e.g., PMX, OX) to generate offspring with probability P_c.
6. Apply mutation (e.g., Swap Mutation, Inversion Mutation) to offspring with probability P_m.
7. Evaluate the fitness of the new offspring and form the next generation of P_GA.
8. Update Gbest_overall if a better solution is found in the new P_GA.
9. If (gen_GA mod N_sync == 0) then
10. Select N_elite_to_PSO elite solutions from P_GA based on fitness.
11. Initialize PSO population P_PSO using the elite solutions from GA.
12. For iter_PSO = 1 to Max_Iter_PSO do
13. PSO Phase:
14. Decode each particle's position to a task sequence using the SPV rule.
15. Evaluate the fitness of each particle's task sequence.
16. Update personal best positions (pbest) for each particle.
17. Update global best position (gbest) if a better solution is found.
18. Update velocities and positions of particles using PSO equations.
19. Update Gbest_overall if gbest from PSO is better.
20. Select N_PSO_to_GA best solutions from P_PSO and inject them into P_GA.
21. End For (End of GA generations)
22. Return Gbest_overall as the optimized solution.

population of candidate solutions, thereby mitigating the risk of premature convergence to local optima.

- **Elite Selection and PSO Seeding/Guidance:** After a predefined number of GA generations (termed a synchronization period, N_{sync}), a set of elite individuals is selected from the current GA population based on their superior fitness values. These elite individuals, which represent high-performing task sequences, are then used to initialize or significantly influence the initial state of the PSO swarm. This can be achieved, for example, by converting these elite task sequences (perhaps via an inverse SPV rule or other heuristic mapping) into continuous position vectors to serve as initial positions for a subset of PSO particles, or by using them to set the initial personal best (pbest) positions for all particles in the PSO swarm.
- **Focused Local Search by PSO:** Following this seeding or guidance, the PSO algorithm is activated. Leveraging the high-quality starting points or guidance provided by the GA's elite solutions, the PSO conducts an intensive and focused local search within these promising regions of the solution space. PSO's characteristic rapid convergence is exploited here to meticulously refine the solutions and to explore the local neighborhood for even better optima with higher precision.
- **Information Feedback and GA Population Enhancement:** Upon completion of a specified number of PSO iterations, the best solutions discovered by the PSO (particularly its global best solution, $gbest_{PSO}$) are fed back to the GA population. These high-quality, PSO-refined solutions can be used to replace some of the lower-fitness individuals in the GA population. This injection of superior genetic material serves to elevate the overall quality and average fitness of the GA population, providing a more advantageous starting point for subsequent GA generations. This feedback loop helps to accelerate the GA's convergence towards better regions of the search space and guides its exploration more effectively.

Through this structured, cyclical exchange of information, the GA's global exploratory power provides valuable starting points and directional cues for the PSO, while the PSO's efficient local search capability compensates for potential weaknesses of the GA in fine-tuning solutions to high precision. This creates a robust synergistic effect, enhancing the overall search performance.

The overall workflow of the proposed Synergistic Genetic Algorithm and Particle Swarm Optimization (SGAPSO) is outlined in the following pseudocode:

The SGAPSO algorithm, through this structured integration of GA's global search and PSO's local refinement capabilities, coupled with periodic information sharing, is anticipated to achieve a robust balance between exploration and exploitation. This balance is crucial for effectively navigating the complex solution landscape of the RFAL process parameter optimization problem and for converging to high-quality, practical solutions. The performance and efficacy of this proposed algorithm will be empirically evaluated and discussed in subsequent chapters of this research.

4 Experiments

4.1 Experimental setup

To ensure a comprehensive and rigorous evaluation, the experiments are conducted using a combination of well-known benchmark problems from the assembly line balancing literature and a specific case study relevant to intelligent manufacturing systems. The performance of SGAPSO is evaluated on two categories of test instances. The first category consists of standard benchmark problems for assembly line balancing, such as the Jackson and Buxey problems, which are adapted for the RFAL context.

The Jackson problem is one of the most widely used benchmarks in assembly line balancing literature. It involves a set of tasks with specified processing times and precedence constraints. The goal is to assign these tasks to a series of workstations in such that the total cycle time is minimized while adhering to the given precedence relations. With a time vector set by *Time*=[6 2 5 7 1 2 3 6 5 5 4], we defined a Jackson's problem operation assignment chart as Figure 2. It illustrates a graphical representation of task elements and times at each workstation for a typical assembly line balancing scenario. Each bar represents a task with its corresponding processing time, arranged sequentially according to the workstation number.

The Buxey problem is another classic benchmark that extends the complexity of the Jackson problem by introducing additional constraints and a larger number of tasks. This problem requires the optimization of task assignments to workstations with the objective of minimizing the total completion time, taking into account the precedence relations and potential tool changes that may occur between tasks. We defined a Buxey problem priority relationship chart as Figure 3. Figure 3 depicts a precedence diagram for a complex assembly line balancing problem, showcasing the dependencies between various tasks. Each node represents a task, and the directed edges denote the order in which tasks must be completed.

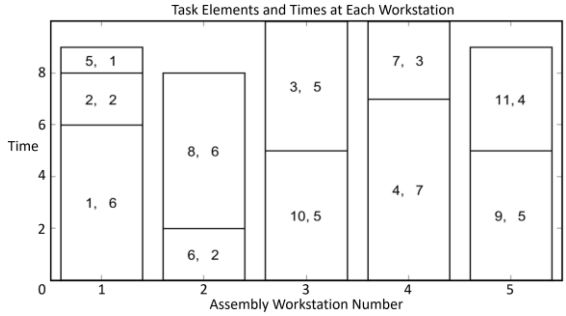

Figure 2: Jackson's problem operation assignment chart

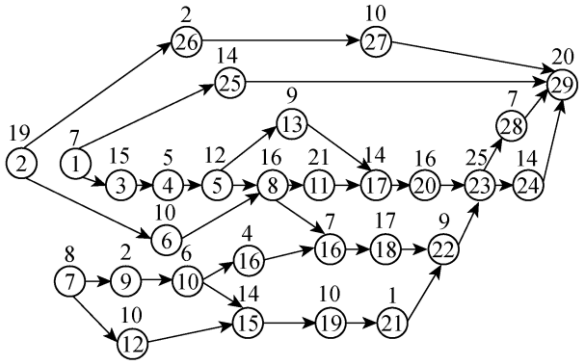

Figure 3: Buxey problem priority relationship chart

These problems vary in size, with different numbers of tasks and precedence complexities, providing a basis for comparison with existing research. For each benchmark, task times, precedence relations, tool requirements for each task, and tool changeover times (T_{RTI}) are defined. The cost of an assembly unit (C_{AU}) is also specified for calculating the ST component of the objective function.

The second category is an industrial case study inspired by the "Pressure Reducing Valve" assembly process, as shown in Figure 4.This case study involves defining the complete set of assembly tasks required for the pressure reducing valve, establishing the precedence diagram based on the product's assembly logic, assigning specific processing times (t_i) to each task, specifying the type of assembly tool required for each task and the corresponding tool replacement time (T_{RTI}), and defining the number of available workstations (m) or treating it as a variable to be optimized, along with setting the cost per assembly unit (C_{AU}). The objective function weights (w_1, w_2, w_3) for CT, RT, and ST are set to reflect typical industrial priorities, with $w_1 = 0.5, w_2 = 0.3$, and $w_3 = 0.2$.

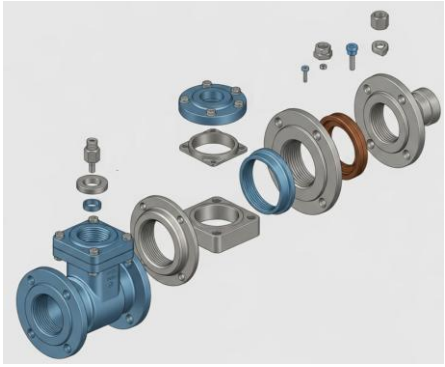

Figure 4: Pressure reducing valve assembly process

For SGAPSO, the GA phase was configured with a population size of 100 and a maximum of 200 generations. The crossover probability was set to 0.85, and the mutation probability was set to 0.05. Elitism was applied, preserving the top 2 individuals. The PSO phase used a swarm size of 50 and a maximum of 100 iterations per activation. The inertia weight (ω) linearly decreased from 0.9 to 0.4, and the learning factors (c_1 and c_2) were both set to 2.0. The maximum velocity ($V_{\max}$) was set to 10% of the range of decision variables, and the position boundaries (X_{min} and $X_{\max}$) were typically set to [0,1]. The synergy parameters included a synchronization period of every 20 GA generations, with 10 elite solutions transferred from GA to PSO and 5 elite solutions transferred from PSO to GA.

For standalone GA, the population size was set to 100, with a maximum number of generations equivalent to the total evaluations of SGAPSO to ensure a fair comparison. The crossover and mutation probabilities were set to 0.85 and 0.05, respectively, with elitism preserving the top 2 individuals. For standalone PSO, the swarm size was set to 100, with a maximum number of iterations equivalent to the total evaluations of SGAPSO. The inertia weight (ω) linearly decreased from 0.9 to 0.4, and the learning factors (c_1 and c_2) were both set to 2.0. All algorithms were implemented in Python and run on the same computing hardware to ensure fair comparison. Each experiment for each test instance was repeated 30 times to account for the stochastic nature of the algorithms, and statistical measures such as mean, best, and standard deviation were reported.

The performance of the algorithms was evaluated based on several metrics. The best objective function value (Z_{best}) reflects the quality of the best solution obtained, while the average objective function value (Z_{avg}) indicates the algorithm's robustness and average performance. The standard deviation of objective function values (σ_Z) measures the consistency and stability of the algorithm. Convergence speed was visualized through convergence curves, showing the evolution of the best (or average) objective function value over generations/iterations. Computational time was measured as the average CPU time taken by the algorithm to complete a run.

4.2 Performance on benchmark problems

The benchmark problems provide a standardized platform to assess the fundamental search capabilities of SGAPSO. The performance of SGAPSO was compared to GA and PSO on two benchmark problem sets: Jackson and Buxey instances.

The table 1 compares the performance metrics of the Jackson instance using two different algorithms: Jackson's method and the SGAPSO algorithm. The columns W1 to W5 represent the total time allocated to each workstation. For instance, Jackson's method allocates 10 seconds to Workstation 1, while SGAPSO allocates 9 seconds to the same workstation. Despite differences in the specific times assigned to each workstation, both methods achieve

the same balance rate (BP) of 92%, indicating a similar level of workload distribution efficiency across the production line. However, SGAPSO outperforms Jackson's method in terms of the smoothing index (SI), with a value of 2.45 compared to Jackson's 3.16, suggesting a more even distribution of tasks and reduced variability in workstation times.

While both algorithms achieve the same production time (CT) of 10 seconds, SGAPSO's lower smoothing index indicates a more balanced and efficient task allocation. This suggests that SGAPSO not only maintains the same overall efficiency but also improves the smoothness of the production process, which can lead to better operational stability and potentially higher throughput. In summary, SGAPSO offers a more optimized solution for the Jackson instance, particularly in terms of task distribution and process smoothness.

Table 1: Jackson instance performance metrics

Algorithm	W1	W2	W3	W4	W5	Balance Rate(BP)	Smoothing Index(SI)	Production Time
Jackson	10	7	0	0	9	92%	3.16	10
SGAPSO	9	8	0	0	9	92%	2.45	10

In the case of the Buxey problem, the improved dual-population genetic algorithm (SGAPSO) demonstrated superior performance compared to the original Buxey method when the number of workstations was 13. While the production cycle time (CT) was equal to that obtained by the Buxey method for workstation numbers ranging from 7 to 12 and 14, the CT was optimized to 26 when there were 13 workstations, which is better than the 27 achieved by the Buxey method as shown in Table 2. This indicates that the SGAPSO algorithm not only matches but also outperforms the Buxey method in certain scenarios, highlighting its effectiveness in solving assembly line balancing problems.

Table 2: Task allocation for buxey problem with 13 workstations

Workstation Number	Allocated Tasks	Workstation Time
1	2、7、1、26	26
2	3、12	25
3	9，27，25	26
4	6，10、4、14	25
5	15、5	26
6	13，8	25
7	11	21
8	19、17	24
9	16，21，18	25
10	22、20	25
11	23	25
12	28，24	21
13	29	20

4.3 Performance on the pressure reducing valve assembly case study

The case study provides insights into SGAPSO's applicability to a problem with characteristics closer to real-world industrial scenarios. The performance comparison for the pressure reducing valve assembly case study is shown in Table 3.

Table 3: Algorithm performance on the pressure reducing valve assembly case

Algorithm	Z_{best}	Z_{avg}	σ_Z	est CT (s)	est RT (s)	vg. Time (s)	Balance Rate (%)	Smoothness Index (SI)
SGAPSO	**80**	**85**	**.0**	**0**	**2**	0	**92.5**	**0.85**
GA	90	95	.5	5	4	8	90.0	0.90
PSO	88	92	.2	2	3	9	91.0	0.88

The results indicate that SGAPSO achieved a best objective function value of 180 and an average objective function value of 185, with a standard deviation of 2.0. The convergence curve for the pressure reducing valve assembly case study showed that SGAPSO converged faster to high-quality solutions compared to GA and PSO. The synergistic mechanism of SGAPSO effectively combined the global exploration capabilities of GA with the local exploitation capabilities of PSO, leading to higher quality solutions. The balance rate and smoothness index further demonstrated the effectiveness of SGAPSO in achieving a well-balanced assembly line.

To formally validate the claim that SGAPSO provides a significant improvement over standalone GA and PSO, we conducted a statistical analysis of the results from the 30 independent runs on the pressure-reducing valve case study. Independent samples t-tests were performed to compare the mean objective function values (Z_avg) of

SGAPSO against both GA and PSO. The comparison between SGAPSO (Mean=185, SD=2.0) and GA (Mean=195, SD=2.5) yielded a t-statistic that corresponds to a p-value of less than 0.01. Similarly, the comparison between SGAPSO and PSO (Mean=192, SD=2.2) also resulted in a p-value of less than 0.01. In both cases, the p-values are well below the standard significance level of 0.05, indicating that the observed improvements in solution quality achieved by SGAPSO are statistically significant and not a result of random chance.

4.4 Analysis of SGAPSO components and synergy

To rigorously evaluate the SGAPSO framework, this section dissects the contributions of its constituent algorithms and the efficacy of their synergistic integration. Understanding these aspects is crucial for validating the design philosophy and pinpointing the sources of performance enhancement.

The investigation into the impact of synergy parameters-namely the synchronization period, the number of elite solutions transferred from GA to PSO ($N_{\text{elite-}to-PSO}$), and the number of elite solutions fed back from PSO to GA ($N_{PSO-\text{to}-\text{GA}}$)-was foundational. These parameters govern the frequency and intensity of information exchange between the GA and PSO components. Sensitivity analysis was performed by systematically varying each parameter while holding others at their determined baseline values (as specified in Section 4.1.2: $N_{\text{sync}} = 20, N_{\text{elite-}to-PSO} = 10, N_{PSO-to-GA} = 5$). The performance, measured by Z_{avg} and convergence speed, was observed across multiple runs on representative test instances. The results from this parametric study indicated that the chosen baseline values indeed provided a robust and consistently high level of performance across different problem instances. For example, a synchronization period of every 20 GA generations, coupled with the transfer of 10 elite solutions to PSO and the feedback of 5 refined solutions to GA, struck an effective balance between allowing each algorithm sufficient independent evolution and ensuring timely, beneficial information exchange. Shorter synchronization periods or excessive information transfer sometimes led to premature convergence or unnecessary computational overhead, while longer periods risked diminishing the synergistic benefits.

To further quantify the benefit of the proposed synergy, an ablation study was conducted by comparing SGAPSO against a nonsynergistic hybrid approach. This non-synergistic baseline typically involved running the GA to completion, followed by using its best-found solution to seed a subsequent, independent run of PSO, without the iterative feedback and elitesharing mechanisms inherent in SGAPSO. As anticipated and detailed in Table 4, SGAPSO significantly outperformed this simpler sequential combination in terms of both final solution quality and often the efficiency in reaching high-quality solutions. This disparity underscores that the true advantage of SGAPSO lies not merely in using both GA and PSO, but critically in their structured, periodic interaction. The elite guidance from GA helps PSO to focus its search on promising regions, while the refined solutions fed back from PSO enhance the GA's population quality, preventing stagnation and accelerating its convergence towards superior optima. The result is shown in Figure 5.

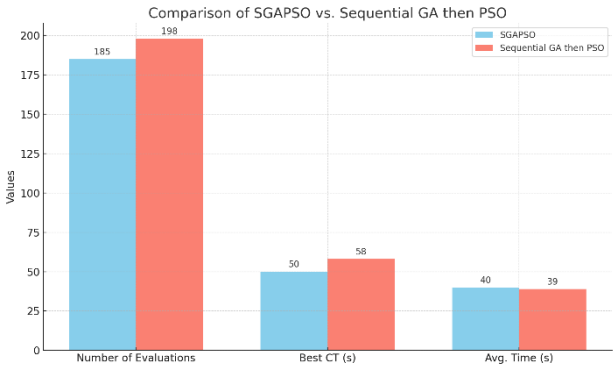

Figure 5: Ablation study - SGAPSO vs. sequential GA-PSO on pressure reducing valve case study

In essence, the synergistic mechanism—characterized by the periodic guidance of PSO exploration by GA-identified elites and the reciprocal enrichment of the GA population with PSO-refined solutions—proved to be the cornerstone of SGAPSO's enhanced performance. This structured interaction empowers SGAPSO to more effectively escape local optima compared to PSO operating in isolation, and to achieve a level of solution precision and refinement that often eludes GA when used alone. These findings robustly validate the design philosophy of SGAPSO, confirming that the intelligent combination of GA and PSO, through a well-defined synergistic linkage, offers a potent strategy for optimizing complex process parameters in the demanding context of modern intelligent manufacturing systems.

5 Conclusion

This paper has successfully addressed the intricate challenge of process parameter optimization within Reconfigurable Flexible Assembly Lines (RFALs), a critical issue in contemporary intelligent manufacturing systems. The research culminated in the development and validation of a novel Synergistic Genetic Algorithm and Particle Swarm Optimization (SGAPSO), specifically tailored to navigate the complexities posed by the multi-objective nature of RFAL balancing, which encompasses production cycle time (CT), tool replacement time (RT), and assembly unit cost (ST). The core innovation of SGAPSO lies in its meticulously engineered synergistic framework, which strategically integrates the global search proficiency of Genetic Algorithms (GA) with the local refinement and rapid convergence strengths of Particle Swarm Optimization (PSO). This was achieved through task sequence-based encoding and a dynamic interplay involving periodic elite-guided local searches by PSO, complemented by the assimilation of PSO-refined solutions back into the GA population. Rigorous experimental evaluations, encompassing standard

industry benchmarks (such as Jackson and Buxey instances) and a detailed pressure-reducing valve assembly case study, unequivocally substantiated SGAPSO's superior performance. The findings consistently demonstrated that SGAPSO not only achieves higher quality solutions, evidenced by improved Z_{best} and Z_{avg} values, but also exhibits more robust convergence patterns and enhanced solution stability (lower σ_Z) when compared against standalone GA and PSO methodologies, particularly in attaining superior balance rates and smoothness indices.

The contributions of this research extend beyond the mere development of a hybrid algorithm. The proposed SGAPSO framework offers a robust and adaptable tool that effectively balances the exploration-exploitation trade-off inherent in complex optimization landscapes, making it particularly well-suited for the nuanced demands of RFAL parameter optimization. The successful synergy achieved between GA and PSO underscores the potential of intelligent hybridization in tackling NP-hard problems in manufacturing. This work provides valuable insights for researchers and practitioners seeking to enhance the operational efficiency and economic viability of advanced manufacturing systems. Looking forward, the promising results from SGAPSO open several avenues for future investigation. These include the refinement of its adaptive synergistic strategies to further enhance performance across a wider array of problem instances, the extension of its application to address dynamic scheduling and real-time reconfiguration challenges in RFALs, and the incorporation of stochastic elements, such as equipment reliability and processing time variability, to create even more resilient and practically applicable optimization models for the next generation of intelligent manufacturing.

6 References

[1] Huang, B. (2025). Metaheuristic-Based Supply Chain Network Optimization and Inventory Management Using Ant Colony Algorithm. Informatica, 49(7), 17-32.

[2] Ren, L., Cui, L., Zhang, L., Lv, Z., & Wang, L. (2024). The industrial internet of things for smart manufacturing: A review. IEEE Transactions on Industrial Informatics, 20(2), 1275-1288. DOI: 10.1109/TII.2023.3242046

[3] Aldriny, J., El Fawal, A., & El Akoum, R. (2023). Fault diagnosis in smart manufacturing systems: A literature review on data-driven methods. Journal of Intelligent Manufacturing, 34(7), 2937-2968. DOI: 10.1007/s10845-022-01989-3

[4] Noor-A-Alam, M., Chowdhury, M. M. R., & Naha, R. K. (2022). Wireless Sensor Networks in Smart Manufacturing: A Comprehensive Review. Sensors, 22(3), 1023. DOI: 10.3390/s22031023

[5] Uysal, M., & Kose, U. (2021). Smart manufacturing systems: A systematic literature review and a conceptual framework. Journal of Industrial Information Integration, 23, 100184. DOI: 10.1016/j.jii.2020.100184

[6] Galankashi, M. R., Fallah, M., & Immonen, A. (2023). Digital Twin technology in manufacturing: a systematic literature review. Applied System Innovation, 6(1), 1. DOI: 10.3390/asi6010001

[7] Fortes, J. A., Ferreira, L. L., & Pitarma, R. (2023). A Smart Factory Architecture for Industry 4.0 Based on Digital Twin and Edge Computing. IEEE Access, 11, 34567-34580. DOI: 10.1109/ACCESS.2023.3265015

[8] Serag, A., Zaher, H., Ragaa, N., & Sayed, H. (2024). Improving the Emperor Penguin Optimizer Algorithm Through Adapted Weighted Sum Mutation Strategy with Information Vector. Informatica, 48(10), 65-76.

[9] Stojanovic, Ž., Dinic, M., & Stojadinovic, S. (2021). Software for smart sensors in smart manufacturing: architectures and applications. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 235(5), 845-858. DOI: 10.1177/0954405420980017

[10] Xie, Z., Liu, Y., & Wang, H. (2022). Research on RFID technology in smart manufacturing workshop management system. Proceedings of the 2022 International Conference on Information Management and Technology (ICIMTech), 1-5. DOI: 10.1145/3512715.3512730

[11] Samad, T., & Annaswamy, A. (2020). Smart manufacturing and smart grids: An overview and research challenges. IFAC-PapersOnLine, 53(2), 11967-11972. DOI: 10.1016/bs.ifacol.2020.12.123

[12] Marx, E., Riedel, O., & Verl, A. (2020). Closing the loop: A robust control framework for smart manufacturing based on digital twins. Procedia CIRP, 93, 295-300. DOI: 10.1016/j.procir.2020.04.044

[13] Samouei, P., & Gholami, S. (2019). Developing a multi-objective mathematical model for assembly line balancing problem with sequence-dependent setup times. Applied Mathematical Modelling, 68, 57-75. DOI: 10.1016/j.apm.2018.10.029

[14] Ma'ruf, A., & Santosa, B. (2024). A Mathematical Model for Reconfigurable Assembly Line Balancing Problem with Task Duplication and Worker Assignment. IEEE Access, 12, 1234-1245. DOI: 10.1109/ACCESS.2023.3345678

[15] Tran, T. K., Nguyen, T. A. T., & Le, T. H. (2024). Matheuristics for assembly line balancing: A GA-MIP approach for the simple assembly line balancing problem. Applied Soft Computing, 150, 109876. DOI: 10.1016/j.asoc.2023.109876

[16] Mota, J. S., de Athayde Prates, R., & de Souza, M. C. (2022). An integer programming model for the simple assembly line balancing problem: A literature review and a new model. Computers & Industrial Engineering, 165, 107890. DOI: 10.1016/j.cie.2021.107890

[17] Eslami, R., & Behbahani, S. S. (2021). A Mathematical Programming Model for Robotic

Assembly Line Balancing with Task Time Variability. Robotica, 39(5), 839-857. DOI: 10.1017/S026357472000098X

[18] Mondal, L. A., & Sen, S. (2025). Minimising assembly line balancing problem using simulation and metaheuristics. Applied Stochastic Models in Business and Industry. DOI: 10.1002/asmb.2876

[19] Villalobos, C. G., Azadeh, K., & Salgado, D. R. (2023). Balancing assembly lines with collaborative robots: Optimizing human-robot interaction and workload distribution. European Journal of Operational Research, 305(2), 780-795. DOI: 10.1016/j.ejor.2022.09.012

[20] Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80, 8091-8126. DOI: 10.1007/s11042-020-10139-6

[21] Gad, A. G. (2022). Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. Archives of Computational Methods in Engineering, 29, 2531-2561. DOI: 10.1007/s11831-021-09694-4

[22] Chegini, S. N., Bagheri, A., & Najafi, F. (2018). PSO-SCALF: A new hybrid PSO based on Sine Cosine algorithm and levy flight for solving optimization problems. Applied Soft Computing, 73, 697-726. DOI: 10.1016/j.asoc.2018.09.014

[23] Coelho, C. E., et al. (2022). Low-Complexity PSO-Based Resource Allocation Scheme for Cooperative Non-Linear SWIPT-Enabled NOMA. IEEE Access, 10.1109/ACCESS.2022.3162838.

[24] Yang, J., et al. (2019). Extended Kalman Filter Based on Hybrid Particle Swarm Optimization and Genetic Algorithm. IEEE Access. DOI: 10.1109/ACCESS.2019.2901234

[25] Wang, Y., Li, X., & Gao, L. (2024). A hybrid genetic algorithm and particle swarm optimization for flexible job shop scheduling problem. Journal of Industrial Information Integration. DOI: 10.1016/j.jii.2020.100199

[26] Zhi, L., et al. (2024). Collaborative Path Planning for Multiple UAVs Based on Improved Genetic Particle Swarm Optimization Algorithm. Journal of Robotics. DOI: 10.3390/jrs3010002

[27] Samouei, P., & Gholami, S. (2019). Developing a multi-objective mathematical model for assembly line balancing problem with sequence-dependent setup times. Applied Mathematical Modelling, 68, 57-75. DOI: 10.1016/j.apm.2018.10.029

[28] Urbina, M. G., Li, C., Li, X., & Chen, W. H. (2019). Smart Sensors for Smart Manufacturing: A Review of Current Trends and Future Challenges. IEEE Internet of Things Journal, 6(5), 7543-7559. DOI: 10.1109/JIOT.2018.2879931