

Sim-heuristic Algorithm for Solving Bi-objective Stochastic Optimization Problems

Shih-Cheng Horng

schong@cyut.edu.tw

Revised and Resubmitted to **Scientific
Reports**

as a **REGULAR PAPER**

Corresponding author: Shih-Cheng Horng

Department of Computer Science & Information Engineering, Chaoyang
University of Technology, Taiwan, R.O.C.

e-mail : schong@cyut.edu.tw

Abstract

Bi-objective stochastic constrained optimization problems (BSCOP) involve bi-objective stochastic functions and deterministic constraints. BSCOP is NP-hard because of exponential growth in computational complexity. A sim-heuristic algorithm can solve NP-hard problems, where the simulation addresses model uncertainty and interacts with the metaheuristic approach, which explores the search space to identify a near-optimal solution. A sim-heuristic approach that combines leaf in wind optimization with ordinal optimization (LWOO), is presented to solve BSCOP efficiently. The LWOO approach comprises three phases: surrogate model, universal exploration, and local exploitation. The Kolmogorov-Arnold Network (KAN) serves as a surrogate model to efficiently evaluate solution performance. In universal exploration, a refined leaf in wind optimization is developed to select N promising solutions from the whole search space. An advanced optimal computing budget allocation is utilized in local exploitation to identify a near-optimal solution from the N promising solutions. The LWOO approach is employed for production planning in reconfigurable manufacturing systems. The purpose is to find the optimal system configuration, task allocation to the workstation, and buffer allocation to simultaneously maximize expected throughput while minimizing the total buffer capacity. A reconfigurable manufacturing system consisting of 3 workstations with 2 inter-station buffers serves as a practical example. The LWOO approach was compared with four heuristic methods to validate the performance. Experimental results demonstrate that the LWOO approach consistently produces optimal and reliable solutions within a reasonable timeframe.

Keywords: Bi-objective stochastic constrained optimization, leaf in wind optimization, ordinal optimization, Kolmogorov-Arnold network, optimal computing budget allocation, reconfigurable manufacturing system.

Nomenclature

BSCOP	Bi-objective stochastic constrained optimization problems
LWOO	Leaf in wind optimization with ordinal optimization
RMS	Reconfigurable manufacturing systems
OO	Ordinal optimization
LiWO	Leaf in wind optimization
KAN	Kolmogorov-Arnold network
RLWO	Refined leaf in wind optimization
AOCBA	Advanced optimal computing budget allocation
CE	Coarse evaluation
EE	Rigorous evaluation
$E[f_i(\mathbf{x})]$	The i -th expected objective value
$\mathbf{x}=[x_1, \dots, x_K]^T$	K -dimensional solution
$h(\mathbf{x}) \leq H$	Inequality constraint
$g(\mathbf{x}) = G$	Equality constraint
$\mathbf{LB}=[LB_1, \dots, LB_K]^T$	Lower limit
$\mathbf{UB}=[UB_1, \dots, UB_K]^T$	Upper limit
$\bar{f}_i(\mathbf{x})$	Sample average of i -th objective value
L	Quantity of replications
$f_i^\ell(\mathbf{x})$	Estimate of the ℓ th replication
$F(\mathbf{x})$	Penalized function
α	Weighting parameter
η	Penalty parameter
L_e	Replications of rigorous evaluation
$F_e(\mathbf{x})$	Penalized function using rigorous evaluation
R	Number of sampling solutions
$f_{\text{KAN}}(\mathbf{x})$	Predicted output of KAN
$\phi_{p,q}(\cdot)$	Univariate functions
$\Phi_p(\cdot)$	Combined univariate functions
C	Adaptive movement factor
$P1$	Threshold of breeze translation motion
$P2$	Threshold of breeze spiral motion
$P3$	Reset threshold
Ψ	Total number of leaves
$t_{\max}$	Maximum number of iterations
$\mathbf{x}_i^t$	Position of the i -th leaf
$\mathbf{x}_{gb}^t$	Global best leaf position
$C_{\min}$	Minimum of C
$C_{\max}$	Maximum of C
$P1_{\min}$	Minimum of $P1$

$P1_{\max}$	Maximum of $P1$
$P2_{\min}$	Minimum of $P2$
$P2_{\max}$	Maximum of $P2$
$P3_{\min}$	Minimum of $P3$
$P3_{\max}$	Maximum of $P3$
CB	Allowable simulation budget
N	Number of promising solutions
L_0	Initial replications
L_i	Replications assigned for the i th solution
Δ	Specific given computing budget
ν	Speedup factor
S	Number of workstations
T	Number of tasks
M	Available number of machines
$M_{\min}$	Minimum quantity of machines
$M_{\max}$	Maximum quantity of machines
$B_{\min}$	Safety buffer capacity
$B_{\max}$	Maximum buffer capacity
W_s	Quantity of machines for workstation s
x_s	Quantity of tasks for workstation s
B_s	Buffer capacity for buffer s
$E[TH(\mathbf{w}, \mathbf{x}, \mathbf{b})]$	Expected throughput
$\mathbf{w}^*$	Optimal system configuration
$\mathbf{x}^*$	Optimal task allocation
$\mathbf{b}^*$	Optimal buffer capacity
$F(\mathbf{w}, \mathbf{x}, \mathbf{b})$	Penalized function of RMS
Ω	Representative subset
rk	Rank of a near-optimal solution in Ω

1. Introduction

Bi-objective stochastic constrained optimization problems (BSCOP) involve bi-objective stochastic functions and deterministic constraints [1]. The goal of BSCOP is to seek an optimal design for optimizing the bi-objective stochastic functions subject to all constraints [2]-[3]. BSCOP occurs across different applications, including reverse supply chain networks, facility location, and production planning for reconfigurable manufacturing systems (RMS). The BSCOP is a typical example of an NP-hard class. NP-hard problems are highly challenging as they are not typically solvable within polynomial time, although any proposed solutions can be efficiently verified in polynomial time.

Solving BSCOP is challenging due to its large search space, the computational effort needed to estimate random bi-objective values, and the requirement to satisfy all constraints. Real-life BSCOP are often large-scale and require high-quality solutions in a short time. Metaheuristic approaches can generate near-optimal solutions for NP-hard problems in a short computing time [4]-[9]. Metaheuristic methods implement global search to explore near-optimal solutions. Most metaheuristics are stochastic, mimicking natural, physical, or biological principles and resembling an optimization search process. Although metaheuristic methods benefit from parallelization paradigms and different random search, they usually assume that the objective function and constraints are deterministic. Mixing metaheuristics with other methodologies is a good approach to resolve difficult BSCOP [10]-[11]. Among them, sim-heuristic approaches provide a general technique integrating metaheuristics with simulation to solve BSCOP [12]-[13].

Sim-heuristic approaches combine simulation techniques within a metaheuristic optimization framework to address BSCOP. In sim-heuristic approaches, simulation replications are performed to evaluate the objective values of solutions obtained by the metaheuristic. Then, the feedback from the simulation is used to guide the metaheuristic

search process. However, the simulation is time-consuming during the metaheuristic search. Two challenges arise regarding the solution provided by the metaheuristic. (i) Is this solution good enough to invest an additional computing budget in a stochastic environment? (ii) How much of the computing effort should be provided to achieve the required minimum level of accuracy? Fortunately, ordinal optimization (OO) proposed by Y.C. Ho [14] is becoming a relevant framework to address these two challenges. For the alignment probability equals 0.99, the OO framework guarantees that an optimal solution will be provided, which is among the top 3.5% of all solutions. OO frameworks have many applications, such as the sorting conveyor systems [15], medical resource allocation [16], container freight station allocation [17], and multiple-item inventory system [18].

In study [15], the process of optimizing shortcut layouts within a sorting conveyor system is presented as a constrained optimization problem involving binary variables. To address this, tensor-product B-splines is incorporated into the binary seagull optimization algorithm to efficiently identify the optimal shortcut layout that maximizes system capacity. In study [16], allocating hospital medical resources is modeled as a constrained bi-objective stochastic optimization problem. A metamodel-based optimization approach that combines regularized minimal-energy tensor-product splines with a driving training-based optimization is proposed. The approach aims to determine the optimal allocation of medical resources, targeting the simultaneous reduction of both the expected patient length of stay and the average cost of medical waste. The study [17] aims to enhance the efficiency of shipping and receiving docks at a container freight station within the air cargo sector, formulating the challenge as a constrained integer optimization problem. Polynomial chaos expansion is incorporated into golden jackal optimization to identify the optimal shipping/receiving docks, aiming to minimize average truck waiting times while ensuring queue stability. Study [18] presents a simulation-based optimization framework that embeds Monte Carlo simulation within a skill

optimization algorithm to identify the optimal stock levels of a multiple-item inventory system, thereby minimizing the expected total cost.

To reduce the computational costs associated with BSCOP, a sim-heuristic approach that combines leaf in wind optimization (LiWO) [19] with ordinal optimization (OO), abbreviated as LWOO, is proposed to obtain a near-optimal solution in a reasonable time. The LWOO approach consists of three phases: surrogate model, universal exploration, and local exploitation. The Kolmogorov-Arnold Network (KAN) [20] serves as a surrogate model to efficiently evaluate solution performance. A refined leaf in wind optimization (RLWO) is developed to perform universal exploration and select N promising solutions from the entire search space. An advanced optimal computing budget allocation (AOCBA) is developed in local exploitation to identify a near-optimal solution from the N promising solutions. In universal exploration, the RLWO provides greater exploration because solutions are attracted to different regions of the search space. In local exploitation, AOCBA provides more accurate performance estimates for currently promising solutions. These three phases can effectively reduce the computing budget needed to solve BSCOP.

The advantage of sim-heuristics is that only a subset of promising solutions needs to be simulated. This benefit is significant in population-based algorithms. Because they are based on fast constructive heuristics, they can generate a large number of high-quality promising solutions in a very short time. The proposed LWOO approach effectively balances the computational budget of sim-heuristics between optimization and simulation. Consequently, RLWO assisted by KAN can quickly guide optimization toward more promising solutions. However, when RLWO stagnates, switching from optimization to simulation may be more beneficial. The key to the practical application of sim-heuristic lies in determining when to switch from optimization to simulation. Switching too early reduces the likelihood of finding a better solution, while switching too late increases the likelihood of selecting a poorer one.

Accordingly, the AOCBA is developed to enhance simulation efficiency within a given computing budget.

The LWOO approach is employed for RMS production planning [21]-[22]. In a small and diverse consumer market, manufacturers' adaptability to diverse demand, speed of market response, and product quality are essential. When facing the needs of different customers, how should manufacturers respond to this market environment? The most crucial factor is the flexibility of manufacturing equipment. The most important feature of the RMS is its reconfigurability, which enables the system to quickly adjust output products or production capacity in response to customer needs or market changes. Therefore, the RMS provides manufacturers with a high degree of flexibility, helps the manufacturing industry transform in the post-epidemic era, and enables quick response to market changes. The production planning of RMS can be mathematically modeled as a BSCOP. This study makes the following two contributions. The *first* is to propose an LWOO approach for BSCOP to find a near-optimal solution quickly. The *second* is applying the LWOO approach to determine the optimal task assignment to the workstation and the buffer allocation to maximize expected throughput while minimizing RMS's total buffer capacity.

The structure of the paper is organized as follows. Section 2 introduces the LWOO approach, designed to identify near-optimal solutions for BSCOP. Section 3 focuses on the production planning within the RMS, which is formulated as a BSCOP and addressed using the LWOO method. Section 4 outlines the conducted experiments, analyzes the results, and evaluates performance against four heuristic approaches. Lastly, Section 5 concludes the paper and highlights directions for future research.

2. Sim-heuristic Approach

2.1 BSCOP

A typical BSCOP has the following form.

$$\min \{E[f_1(\mathbf{x})], E[f_2(\mathbf{x})]\} \quad (1)$$

$$\text{subject to } h(\mathbf{x}) \leq H, \quad (2)$$

$$g(\mathbf{x}) = G, \quad (3)$$

$$\mathbf{LB} \leq \mathbf{x} \leq \mathbf{UB}. \quad (4)$$

where $\mathbf{x} = [x_1, \dots, x_K]^T$ denotes a K -dimensional solution, $E[f_i(\mathbf{x})]$ depicts the i -th expected objective value, $h(\mathbf{x}) \leq H$ depicts the inequality constraint, $g(\mathbf{x}) = G$ depicts the equality constraint, and $\mathbf{LB} = [LB_1, \dots, LB_K]^T$, $\mathbf{UB} = [UB_1, \dots, UB_K]^T$ represent the lower and upper limit, respectively. An adequate number of replications is conducted to estimate $E[f_i(\mathbf{x})]$ in an exact model. The sample average approximation is an approach that is frequently employed to evaluate $E[f_i(\mathbf{x})]$, where L depicts the number of replications, and $f_i^\ell(\mathbf{x})$ is the performance of the ℓ th replication.

$$\bar{f}_i(\mathbf{x}) = \frac{1}{L} \sum_{\ell=1}^L f_i^\ell(\mathbf{x}) \quad (5)$$

A large value of L can evaluate $E[f_i(\mathbf{x})]$ more accurately. Therefore, we consider two targets: (i) L should be large enough to obtain an accurate evaluation of $E[f_i(\mathbf{x})]$, (ii) $\bar{f}_i(\mathbf{x})$ can be estimated for different solutions to select the best solution.

A common way to combine bi-objective values into a single objective value is the weighted sum method [23]. A common method for managing constraints in stochastic optimization involves the use of penalty functions [24]. These functions incorporate a penalty into the objective function whenever constraints are violated. As a result, we derive the following penalized function.

$$\min F(\mathbf{x}) = \frac{1}{L} \cdot \left[\alpha \cdot \sum_{\ell=1}^L f_1^\ell(\mathbf{x}) + (1-\alpha) \cdot \sum_{\ell=1}^L f_2^\ell(\mathbf{x}) \right] + \frac{\eta}{2} \cdot \left[\left(\max\{0, h(\mathbf{x}) - H\} \right)^2 + (g(\mathbf{x}) - G)^2 \right] \quad (6)$$

where α denotes a weighting parameter, η indicates a penalty parameter,

$(\max\{0, h(\mathbf{x}) - H\})^2$ is the quadratic penalty function for inequality constraint, $(g(\mathbf{x}) - G)^2$ is the quadratic penalty function for equality constraint, and $F(\mathbf{x})$ depicts a penalized function.

The superiority of the OO framework is that it can distinguish good solutions from bad solutions even when using a surrogate model. Thus, the rank of performance remains largely unaffected by substantial approximation errors. However, the penalty method is the simplest for constrained optimization; an enormous value η can guarantee an apparent discontinuity in the penalized function for bad solutions. Let L_e represents the tremendous number of replications, and the rigorous evaluation of (6) is obtained when $L = L_e$. Let $F_e(\mathbf{x})$ denote the penalized function of a solution $\mathbf{x}$ through rigorous evaluation. To efficiently identify N promising solutions within the search space in a reasonable timeframe, a metaheuristic method assisted by a surrogate model is presented. The surrogate model utilizes KAN, and the metaheuristic method adopts RLWO.

2.2 KAN

A surrogate model is a simplified model to approximate the behavior of a more complex system or process. There are several methods for constructing surrogate models, such as polynomial chaos expansion [25], extreme learning machine [26], support vector regression [27], physics-informed neural networks [28], kriging [29], and KAN [20]. Kernel networks map data to higher-dimensional feature spaces using a kernel function, such as a radial basis function. Unlike traditional neural networks, which learn representations across multiple layers, kernel networks focus on computing similarities between data points to generate predictions. KAN shifts the learning paradigm from traditional, static node weights to dynamic, learnable activation functions on the edges. Rather than relying on fixed scalar multiplication, KAN employs a collection of univariate functions, such as B-splines, to transform features as they propagate through layers. KAN was proposed by a research team from MIT, Caltech,

Northeastern University, and the Ohio State University, led by authors Z.M. Liu, Y.X. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark [20]. KAN has recently emerged as a novel approach to function approximation, offering interpretability, efficiency, and adaptability. KAN employs learnable activation functions directly on the edges. Basis functions, including B-splines, wavelets, and radial basis functions, are used to construct activation functions. Somvanshi *et al.* demonstrated that KAN can enhance computational efficiency, adaptability, and interpretability for complex function approximation tasks, and that its applicability is growing in dynamic environments [30]. KAN incorporates learnable activation functions at the edges of the network, which help adapt to the data distribution and dynamically improve generalization capabilities. It can improve scalability and optimize parameter efficiency across various applications, such as function approximation, classification, curve fitting, prediction, and time series forecasting. Thus, a KAN serves as a surrogate model designed to evaluate the performance of a given solution efficiently.

The training data include $(\mathbf{x}_r, F_e(\mathbf{x}_r))$, $r=1, \dots, R$, where R denotes the quantity of random sampling solutions, and $\mathbf{x}_r$, $F_e(\mathbf{x}_r)$ denote the r -th sampling solution and the corresponding penalized function obtained by rigorous evaluation, respectively. The KAN model aims to minimize the discrepancy between the network's predictions $f_{\text{KAN}}(\mathbf{x})$ and the actual target values $F_e(\mathbf{x})$ through training on sample data. Figure 1 displays the network structure of a KAN. A typical KAN is described below.

$$f_{\text{KAN}}(\mathbf{x}) = \sum_{p=1}^{2K+1} \Phi_p \left(\sum_{q=1}^K \phi_{p,q}(x_q) \right) \quad (7)$$

where $f_{\text{KAN}}(\mathbf{x})$ is the predicted output, $\mathbf{x}=[x_1, x_2, \dots, x_K]^T$ denotes a solution, K is the dimension of a solution, $\phi_{p,q}(\cdot)$ are the univariate functions, and $\Phi_p(\cdot)$ takes the univariate functions and combines them. K neurons are in the input layer, $2K+1$ neurons in the hidden

layer, and one in the output layer. There is no linear weight in KAN. Each edge in a KAN represents a univariate function parameterized as a spline, enabling dynamic, fine-grained adjustments.

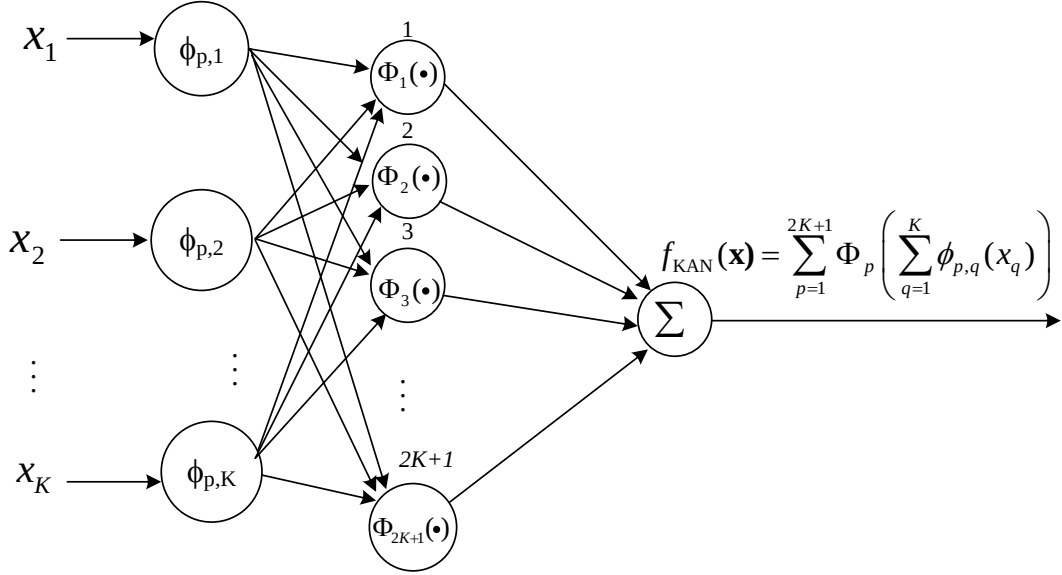


Fig. 1 Framework of a KAN.

Any splines can be adopted in the KAN. The usual choice of learnable activation function used by KAN is the B-splines. B-splines offer greater smoothness and flexibility for modeling complex functions. The B-splines are defined as follows.

$$\Phi(\mathbf{x}) = Q^T \cdot B_d(\mathbf{x}) \quad (8)$$

where Q depicts a vector of coefficients, and $B_d(\mathbf{x})$ represents a vector of functions where each component is a piecewise polynomial.

The KAN is trained off-line using the R random sampling solutions. Once the KAN is off-line trained, the network's predictions $f_{\text{KAN}}(\mathbf{x})$ can be computed simply for a given $\mathbf{x}$.

2.3 RLWO

With the assistance of the KAN, novel metaheuristic approaches can be used to explore the entire search space and seek N promising solutions. Because the LiWO is designed to balance exploration and exploitation, it consistently meets applicable requirements. LiWO

was proposed by researchers N. Fang and Q. Cao [19]. The convergence analyses and statistical comparisons conducted on three benchmark test suites demonstrate that LiWO consistently achieves superior performance compared with ten existing optimizers. Sellam *et al.* proposed a modified LiWO for hyperparameter tuning to assess the plant health and environmental monitoring [31]. Thiagarajan *et al.* applied LiWO to assist in early breast cancer diagnosis, reducing the risk of overfitting through fine-tuning of the model's parameters [32]. Divya *et al.* utilized the LiWO approach to optimize hyperparameters for an accuracy-enhancing diagnostic system [33]. Amanullah *et al.* adopted LiWO to optimize the weight parameters of binarized spiking neural networks of weather prediction [34].

The inspiration for LiWO comes from the dynamic behavior of falling leaves, whose motion is governed by wind forces and translational and rotational movements. Influenced by external forces, falling leaves repeatedly adjust their posture and trajectory as they descend from considerable heights. The falling process represents the successive stages of leaf motion, whereas landing corresponds to the leaf's final state. The motion of falling leaves is characterized by oscillations, rolling and spiral trajectories, along with unpredictable transitions between these states, including random events and abrupt changes in velocity. Therefore, the optimization problem can be effectively solved by simulating the motion of the descending foliage. In this context, the solution space corresponds to the leaf motion domain, with each leaf representing an individual in the optimization process. During the optimization process, wind emerges as the primary factor driving the constant motion of leaves in space. Falling leaves in a dynamic environment are influenced by two distinct types of wind: gentle breezes and strong winds. The winds arise unpredictably, setting the leaves in motion, though they do not occur simultaneously.

However, LiWO often suffers from premature convergence due to rapid diversity loss. The RLWO is proposed to address these limitations. First, a population is generated by random

initialization, and the fitness of each leaf is calculated. Leaves constantly move, influenced either by gentle breezes or strong winds. During this motion, they are affected by external forces, with strong winds presumed to occur randomly across the spatial area. The following four operations are iteratively performed: (a) determining the global best leaf during each movement; (b) dynamically adjusting the adaptive movement factor and three thresholds; (c) breeze driven leaf; and (d) strong wind driven leaf. Leaves carried by a gentle breeze often display a combination of translational and spiral motions. However, not all leaves are influenced by the breeze. The threshold of breeze translation motion, $P1$, and the threshold of breeze spiral motion, $P2$, are introduced.

In each iteration, a leaf x_i randomly generates a random number $R1$. If $R1$ is greater than $P1$, the breeze drives the leaf. The leaf x_i exhibits translational motion, resulting in position updates driven by the effect of a mild breeze. At the same time, each variable $x_{i,k}$ of this leaf also randomly generates a random number $R2$. If $R2$ is greater than $P2$, this variable exhibits spiral motion, updating its position under the influence of a gentle breeze. On the other hand, if $R1$ is less than $P1$, this leaf is driven by strong wind. Compared with gentle breezes acting on foliage, strong winds are proposed to influence leaves by inducing changes in a distinct one-dimensional spatial position. Only a single dimension was impacted, while the others remained unaffected. A randomly selected variable influenced by strong winds was analyzed, leading the leaves to experience unidirectional displacement. In addition, the reset threshold $P3$ is introduced. Each leaf randomly generates a random number $R3$. If $R3$ is less than $P3$, a variable is chosen at random to adjust its position, ensuring the diversity of leaf placement is maintained even under the influence of strong winds. In the LiWO, the three thresholds are fixed, leading to premature convergence.

The core concept of the RLWO is to dynamically adjust the adaptive movement factor and the three thresholds for convergence, diversity, and uncertainty throughout the iterative

process. The adaptive movement factor C is dynamically adjusted to make the translation and spiral motion of the leaves more flexible. The three thresholds are also dynamically adjusted, including the breeze translation motion $P1$ threshold, the breeze spiral motion $P2$ threshold, and the reset threshold $P3$. Common mathematical functions, such as linear, exponential, quadratic, and polynomial, can be used to model dynamic adjustment. Although the quadratic and polynomial functions have more terms, they will require more algorithmic parameters. Compared to a linear function, an exponential function allows for more flexible modeling of the balanced trade-off between diversification and intensification. The control functions for the adaptive movement factor and the three thresholds are developed based on the control parameters of the exponential particle swarm optimization algorithm, providing a framework for defining the adaptive adjustment mechanism [35].

The first step in RLWO is to encode a leaf's position. We use $\mathbf{x} = [x_1, \dots, x_K]^T$ to represent the decision variable as the position of a leaf. The fitness is the objective value $F(\mathbf{x})$. The second step is to assign initial positions to the initial population randomly. The third step is to evaluate the fitness of each leaf, sort these leaves in ascending order according to their fitness, and select the leaf with the smallest fitness as the global best leaf. The fourth step is to adjust the adaptive movement factor and the three thresholds dynamically. The fifth step is that the breeze drives the leaves to update their positions. The sixth step is that strong wind drives the leaves to update their positions. The seventh step is to use the greedy method to update each leaf's position. The eighth step is to check whether the stopping condition is met. The search process is stopped once it has completed the maximum number of iterations.

The notations employed in RLWO are outlined below: Ψ represents the total number of leaves, and $t_{\max}$ is the maximum number of iterations. At the t -th iteration, $\mathbf{x}_i^t = [x_{i,1}^t, \dots, x_{i,K}^t]^T$ and $\mathbf{x}_{gb}^t = [x_{gb,1}^t, \dots, x_{gb,K}^t]^T$ represent the i -th leaf and the global best leaf positions, respectively; $C^t \in [C_{\min}, C_{\max}]$ represents the adaptive movement factor, and

$[C_{\min}, C_{\max}]$ is the range of adaptive movement factor; $P1^t \in [P1_{\min}, P1_{\max}]$ represents the threshold of breeze translation motion, and $[P1_{\min}, P1_{\max}]$ is the threshold range of breeze translation motion; $P2^t \in [P2_{\min}, P2_{\max}]$ represents the threshold of breeze spiral motion, and $[P2_{\min}, P2_{\max}]$ is the threshold range of breeze spiral motion; $P3^t \in [P3_{\min}, P3_{\max}]$ represents the reset threshold, and $[P3_{\min}, P3_{\max}]$ is the range of reset threshold.

Algorithm I: The RLWO

Step 1: Set parameters

Define the values of Ψ , $t_{\max}$, $[C_{\min}, C_{\max}]$, $[P1_{\min}, P1_{\max}]$, $[P2_{\min}, P2_{\max}]$, $[P3_{\min}, P3_{\max}]$, and set $t = 0$, where t denotes an iteration variable.

Step 2: Initialization

$$\mathbf{x}_i^0 = \mathbf{LB} + \lfloor \text{rand}[0,1] \cdot (\mathbf{UB} - \mathbf{LB}) \rfloor, \quad i = 1, \dots, \Psi. \quad (9)$$

where $\mathbf{LB}$ and $\mathbf{UB}$ represent the lower and upper limit, respectively, $\lfloor \cdot \rfloor$ is a floor function, and $\text{rand}[0,1]$ depicts a random floating number within the range $[0, 1]$.

Step 3: Selection of the global best leaf

- (a) Determine the fitness $F(\mathbf{x}_i^t)$ of the i th leaf assisted by KAN, $i = 1, \dots, \Psi$.
- (b) Determine the global best leaf $\mathbf{x}_{gb}^t$ with the smallest fitness.

Step 4: Adjust the adaptive movement factor and three thresholds

$$C^t = C_{\min} + (C_{\max} - C_{\min}) \cdot \exp\left(\ln\left(\frac{C_{\min}}{C_{\max}}\right) \cdot \frac{t}{t_{\max}}\right) \quad (10)$$

$$P1^t = P1_{\min} + (P1_{\max} - P1_{\min}) \cdot \left(1 - \exp\left(\frac{P1_{\max}}{P1_{\min}} \cdot \frac{t}{t_{\max}}\right)\right) \quad (11)$$

$$P2^t = P2_{\max} \cdot \exp\left(-\sqrt{\frac{P2_{\max}}{P2_{\min}}} \cdot \frac{t}{t_{\max}}\right) \quad (12)$$

$$P3^t = P3_{\min} + (P3_{\max} - P3_{\min}) \cdot \exp\left(1 - \frac{t_{\max}}{t}\right) \quad (13)$$

Step 5: Breeze driven leaf

(a) If $\text{rand}[0,1] > P1^t$ for $\mathbf{x}_i^t$, perform breeze translation motion.

$$\mathbf{x}_i^{\text{new}} = \mathbf{x}_i^t + \lfloor \text{rand}[0,1] \cdot (\mathbf{x}_{gb}^t - \mathbf{x}_s^t) \rfloor \quad (14)$$

where $s \neq i$ is a randomly selected leaf.

Else, go to Step 6.

(b) For $k=1, \dots, K$, if $\text{rand}[0,1] < P2^t$ for the k -th dimension of $\mathbf{x}_i^t$, perform breeze spiral motion.

$$x_{i,k}^{\text{new}} = x_{i,k}^t + \lfloor C^t \cdot |x_{i,k}^t - x_{s,k}^t| \cdot \sin(\text{rand}[0,2\pi]) \cdot \text{rand}[0,2\pi] \rfloor \quad (15)$$

where $s \neq i$ is a randomly selected leaf, and $\text{rand}[0,2\pi]$ depicts a randomly generated value uniformly distributed over the range $[0, 2\pi]$.

Step 6: Strong wind driven leaf

(a) For a random k -th dimension of $\mathbf{x}_i^t$, perform strong wind translation motion.

$$x_{i,k}^{\text{new}} = x_{i,k}^t + \lfloor \text{rand}[0,1] \cdot (x_{s,k}^t - x_{i,k}^t) \rfloor \quad (16)$$

where $s \neq i$ is a randomly selected leaf. If $\text{rand}[0,1] < P3^t$ for a random k -th dimension of $\mathbf{x}_i^t$, perform reset.

$$x_{i,k}^{\text{new}} = LB_k + \lfloor \text{rand}[0,1] \cdot (UB_k - LB_k) \rfloor \quad (17)$$

Step 7: Renew the position of the i th leaf

If $x_{i,k}^{\text{new}} < LB_k$, set $x_{i,k}^{\text{new}} = LB_k$, else if $x_{i,k}^{\text{new}} > UB_k$, set $x_{i,k}^{\text{new}} = UB_k$, $i = 1, \dots, \Psi$, $k = 1, \dots, K$.

Compute $F(\mathbf{x}_i^{\text{new}})$ using KAN. If $F(\mathbf{x}_i^{\text{new}}) < F(\mathbf{x}_i^t)$, set $\mathbf{x}_i^{t+1} = \mathbf{x}_i^{\text{new}}$.

Step 8: Termination

If $t \geq t_{\max}$, terminate; else, update $t = t + 1$ and return to Step 3.

When the iteration variable exceeds the maximum allowed iterations $t_{\max}$, the RLWO is terminated, and all leaves are ranked according to their fitness. Subsequently, a subset of candidate solutions is constructed by selecting the most N promising alternatives.

In the OO framework, N represents the candidate subset size and plays a critical role in balancing computational efficiency with the likelihood of identifying the true optimal solutions. Selecting an appropriate value for N is crucial, as it involves balancing processing speed and the probability of capturing the true best outcomes. The goal is to ensure that the candidate subset has a significant overlap with the actual optimum, achieving high alignment probabilities. A smaller N dramatically reduces resource consumption because the AOCBA only evaluates a limited number of potential solutions. However, it increases the risk of missing the true optimal solutions due to errors or noise in the KAN model. Conversely, a larger N enhances computational demands, requiring the AOCBA to run more replications. This not only boosts the likelihood that the candidate subset contains the actual optimum but also serves as a safety measure against inaccuracies stemming from KAN estimates.

The computational complexity of RLWO is predominantly determined by two phases: population initialization and the leaf-position update phase. During population initialization, the computational complexity is $O(\Psi)$, where Ψ represents the number of leaves. The complexity solely attributed to fitness evaluation by KAN has not been considered in this analysis. In the leaf-position update phase, the computational complexity depends on the maximum allowed iterations ($t_{\max}$), the population size (Ψ), and the dimension of the BSCOP (K). Consequently, the computational complexity for this phase can be expressed as $O(t_{\max} \cdot \Psi \cdot K)$. As a result, the overall computational complexity, integrating both the population initialization and the leaf-position update phase, is calculated as $O((1+t_{\max} \cdot K) \cdot \Psi)$.

2.4 AOCBA

The OCBA was originally proposed by C.H. Chen in the mid-1990s [36]. The OCBA strategically allocates computing budgets to reduce estimation variance for performance metrics, making it a crucial tool for simulation experiments. However, the OCBA overlooks the influence of the simulation budget, which is essential in tight budget allocation. A near-optimal solution is determined in local exploitation using the AOCBA from the N promising solutions. The primary focus of the AOCBA is to allocate more computational resources to a select number of potentially critical solutions while minimizing efforts on less significant ones.

More simulation budgets are allocated to crucial solutions whose means are closer to that of the best solution. Let CB represent the allowable simulation budget, and L_i depict the replications provided to the i th solution. The initial replications provided to the N promising solutions are L_0 . A specific given computing budget, Δ , increases at each iteration. The purpose of the AOCBA is to assign CB to L_i so as to asymptotically maximize the probability of correct selection while satisfying $\sum_{i=1}^N L_i = CB$. The allowable simulation budget CB is computed by $CB = \frac{N \times L_e}{\nu}$, where L_e denotes the replications used in rigorous evaluation, and ν is a speedup factor [36].

Algorithm II: The AOCBA

Step 1. Let $l = 0$, $L_i^l = L_0, i = 1, \dots, N$. Calculate the allowable simulation budget

$$CB = \frac{N \times L_e}{\nu}.$$

Step 2. If $\sum_{i=1}^N L_i^l < CB$, go to Step 3; else, the process is stopped, and the near-optimal solution corresponding to the lowest objective value is selected.

Step 3. Determine the sample mean $\bar{F}_i^l$ and the sample standard deviation δ_i^l .

$$\bar{F}_i^l = \frac{1}{L_i^l} \sum_{h=1}^{L_i^l} F_h(\mathbf{x}_i) \quad (18)$$

$$\delta_i^l = \sqrt{\frac{1}{L_i^l} \sum_{h=1}^{L_i^l} (F_h(\mathbf{x}_i) - \bar{F}_i^l)^2} \quad (19)$$

where $\mathbf{x}_i$ represents the i th critical solution, $F_h(\mathbf{x}_i)$ indicates the performance of the h th replication, and $b = \arg \min_i \bar{F}_i^l$.

Step 4. Include a predetermined budget Δ in $\sum_{i=1}^N L_i^l$, and revise the new replications accordingly.

$$L_b^l = \delta_b^l \sqrt{\sum_{i=1, i \neq b}^N \left(\frac{L_i^l}{\delta_i^l}\right)^2} \quad (20)$$

$$L_j^{l+1} = \left(\sum_{i=1}^N L_i^l + \Delta\right) \cdot L_j^l / \left(L_b^l + \sum_{i=1, i \neq b}^N L_i^l\right) \quad (21)$$

$$L_b^{l+1} = \frac{\delta_b^l}{L_j^l} \cdot \sqrt{\sum_{i=1, i \neq b}^N \left(\frac{L_i^l}{\delta_i^l}\right)^2} \cdot L_j^{l+1} \quad (22)$$

$$L_i^{l+1} = \left(\frac{\delta_i^l \cdot (\bar{F}_b^l - \bar{F}_j^l)}{\delta_j^l \cdot (\bar{F}_b^l - \bar{F}_i^l)}\right)^2 \cdot L_j^{l+1} \quad (23)$$

Step 5. Perform $\max[0, L_i^{l+1} - L_i^l]$ additional replications for the i th critical solution, then update the $\hat{F}_i^{l+1}$ and $\hat{\delta}_i^{l+1}$ of additional replications by

$$\hat{F}_i^{l+1} = \frac{1}{(L_i^{l+1} - L_i^l)} \sum_{h=L_i^l+1}^{L_i^{l+1}} F_h(\mathbf{x}_i) \quad (24)$$

$$\hat{\delta}_i^{l+1} = \sqrt{\frac{1}{(L_i^{l+1} - L_i^l)} \sum_{h=L_i^l+1}^{L_i^{l+1}} (F_h(\mathbf{x}_i) - \hat{F}_i^{l+1})^2} \quad (25)$$

Step 6. Update the $\bar{F}_i^{l+1}$ and δ_i^{l+1} of total replications

$$\bar{F}_i^{l+1} = \frac{1}{L_i^{l+1}} \left(L_i^l \cdot \bar{F}_i^l + (L_i^{l+1} - L_i^l) \cdot \hat{F}_i^{l+1} \right) \quad (26)$$

$$\delta_i^{l+1} = \sqrt{\frac{1}{(L_i^{l+1} - 1)} \cdot \left(L_i^l (\bar{F}_i^l)^2 + (L_i^l - 1) (\delta_i^l)^2 + (L_i^{l+1} - L_i^l) (\hat{F}_i^{l+1})^2 + (L_i^{l+1} - L_i^l - 1) (\hat{\delta}_i^{l+1})^2 - L_i^{l+1} (\hat{F}_i^{l+1})^2 \right)} \quad (27)$$

Update $l = l + 1$, then go to Step 2.

AOCBA exhibits a computational complexity of $O(N \cdot \frac{CB}{\Delta})$. The total number of replications performed by AOCBA is constrained by the available simulation budget (CB), which specifies the allocated computational resources. This complexity arises from the iterative allocation mechanism, in which simulation replications are distributed across N alternatives over multiple cycles. In each cycle, the algorithm evaluates the alternatives and updates the allocation strategy, requiring computational effort proportional to N . Since the cumulative number of replications does not exceed the simulation budget (CB), the overall computational complexity is $O(N \cdot \frac{CB}{\Delta})$.

2.5 LWOO

Figure 2 provides a flowchart outlining the LWOO approach.

Algorithm III: The LWOO

Step 1: Configure the values of R , Ψ , $t_{\max}$, $[C_{\min}, C_{\max}]$, $[P1_{\min}, P1_{\max}]$, $[P2_{\min}, P2_{\max}]$,

$[P3_{\min}, P3_{\max}]$, N , L_e , L_0 , and Δ .

Step 2: R $\mathbf{x}$'s are randomly selected from the entire search space to estimate $F_e(\mathbf{x})$, after which the KAN is trained offline using the sampled solutions.

Step 3: Randomly produce Ψ $\mathbf{x}$'s and apply RLWO to these leaves assisted by KAN. Once RLWO has terminated, the Ψ $\mathbf{x}$'s are sorted in ascending order of performance, and the former N $\mathbf{x}$'s are selected to form the candidate subset.

Step 4: AOCBA is employed on the candidate subset of N promising solutions, and the

solution $\mathbf{x}^*$ with the best performance is chosen as the near-optimal solution.

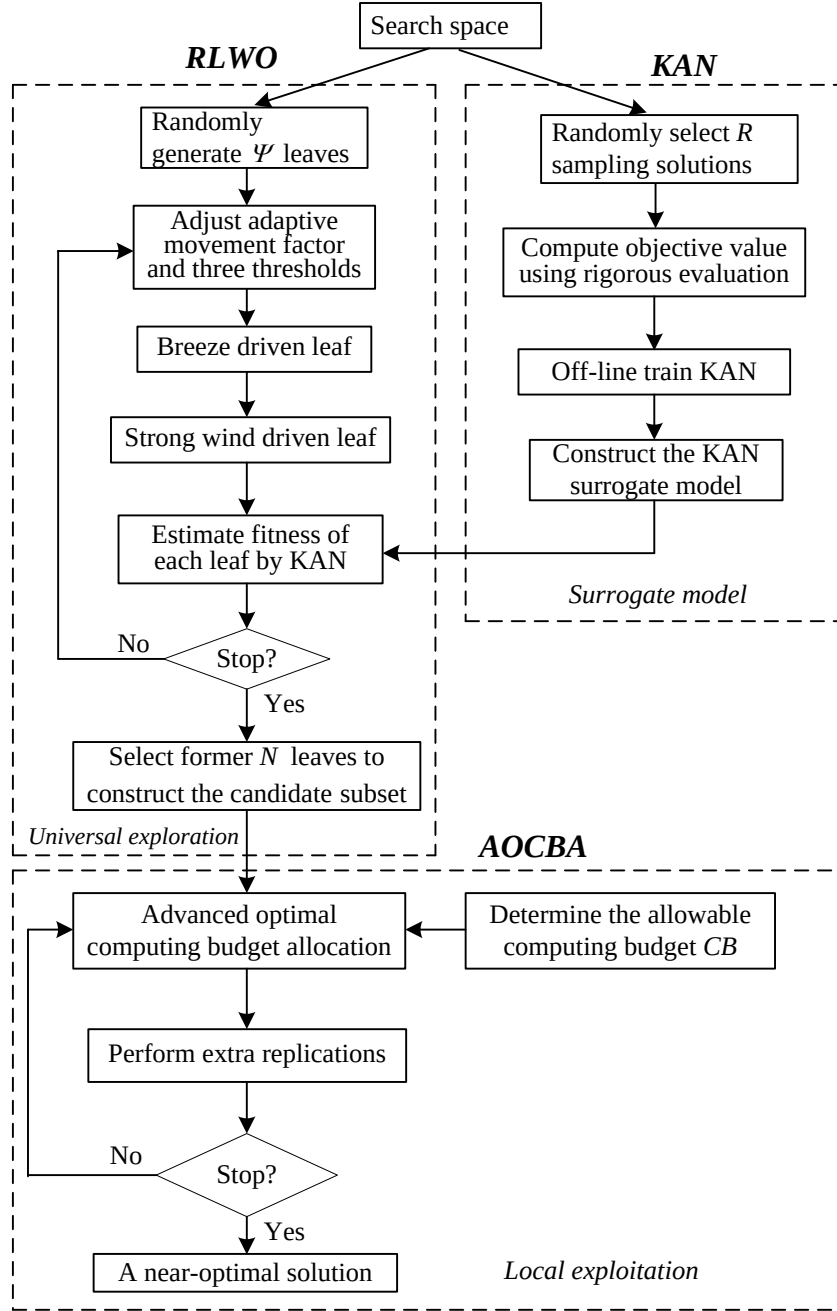


Fig. 2 Flow chart of the LWOO approach.

Because LWOO is built upon RLWO and AOCBA, the resulting computational complexity is $O\left((1+t_{\max} \cdot K) \cdot \Psi + N \cdot \frac{CB}{\Delta}\right)$.

3. Production Planning of Reconfigurable Manufacturing Systems

3.1 Reconfigurable Manufacturing System

In a highly competitive manufacturing industry, companies often experience demand fluctuations driven by seasonal trends, economic shifts, and changing consumer preferences. RMS attempts to respond effectively and quickly to such competitive market demands through the modular, scalable design of its machine components and system levels [37]. RMS can quickly relocate material-handling equipment to respond to market changes. RMS is critical for addressing dynamic production volumes because it enables manufacturing to respond to product changes cost-effectively and rapidly. RMS provides greater modularity and reconfigurability, enabling production to maintain functionality and capacity in line with product demand. An example of an RMS consists of three workstations, each containing multiple parallel, identical machines, as shown in Figure 3 [37]. The RMS configuration accounts for two primary factors: the number of parallel machines and the sequence of tasks processed at each workstation. The RMS configuration is mainly classified into two categories based on the number of products produced: single-product flow line and multi-product flow line. Only a single product is processed in a single-product flow line, while a multi-product flow line produces various products. Multi-product flow lines are becoming increasingly popular, especially in the automotive industry, enabling the production of multiple products on the same line.

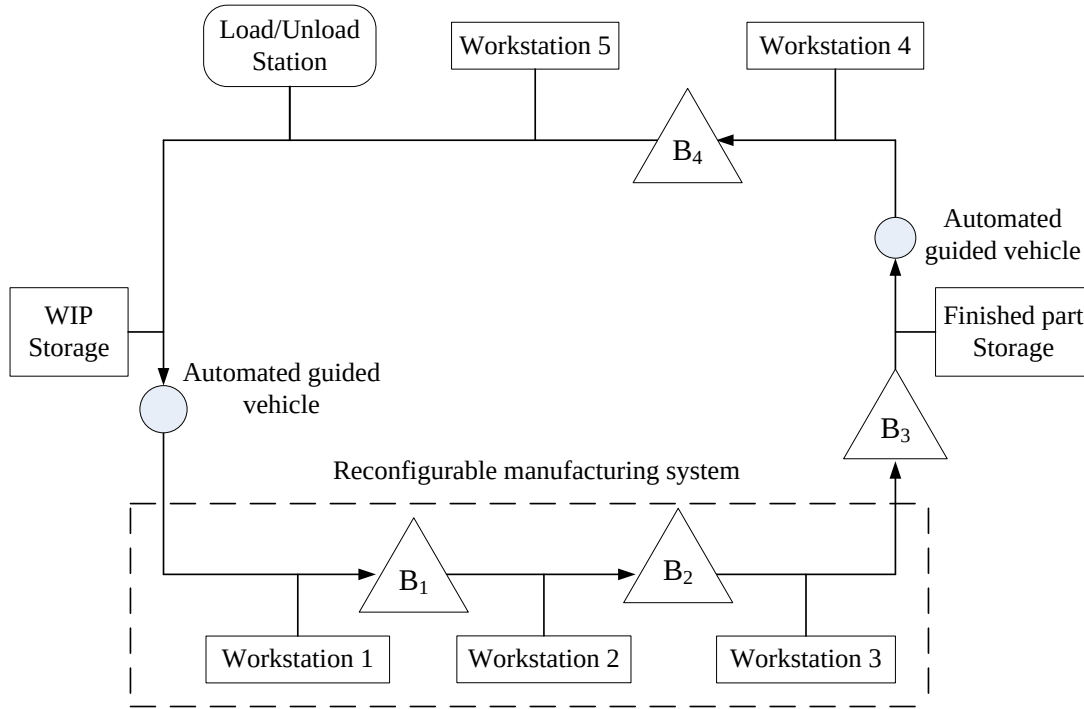


Fig. 3 Reconfigurable manufacturing system.

Despite its advantages in managing demand fluctuations and scalable capacity, RMS design involves NP-hard problems that can be addressed using simulation and optimization techniques. The optimal design of an RMS requires addressing three issues: configuration management, system components, and process planning. Configuration management manages configuration items in a system, which strongly influences functionality and productivity. Most research has focused on machine allocation at workstations. System components determine the number of units required to achieve the desired production capacity. This is a key issue in scalability and capacity planning, and most research has focused on minimizing the number of components in the system. Finally, process planning involves how tasks are assigned to workstations and the overall system balance, which will impact how efficiently the system can be reconfigured to handle fluctuating volumes. The research aims to optimize task allocation across workstations. However, most studies ignore some important factors, such as buffer capacity or system uncertainty and variability. None of the reviewed studies combine task allocation with configuration management in a scalable multi-product flow line

to address fluctuations in production volumes. Accordingly, this work simultaneously maximizes throughput and minimizes total buffer capacity while determining the optimal system configuration and task allocation to achieve scalable capacity under fluctuating production volumes.

3.2 Mathematical Formulation

Consider an RMS, assume there are S workstations and T tasks, and available number of machines is M . Let $M_{\min}$ and $M_{\max}$ depict minimum and maximum number of machines per workstation, respectively, $B_{\min}$ and $B_{\max}$ indicate safety buffer capacity and maximum buffer capacity, respectively, W_s and x_s denote number of machines and tasks for workstation s , respectively, and B_s denote buffer capacity for buffer s .

The production planning of RMS can be formulated as a BSCOP.

$$\max f_1 = E[TH(\mathbf{w}, \mathbf{x}, \mathbf{b})] \quad (28)$$

$$\min f_2 = \sum_{s=1}^{S-1} B_s \quad (29)$$

$$\text{subject to } \sum_{s=1}^S W_s \leq M, \quad (30)$$

$$\sum_{s=1}^S x_s = T, \quad (31)$$

$$M_{\min} \leq W_s \leq M_{\max}, \quad s = 1, \dots, S, \quad (32)$$

$$1 \leq x_s \leq T, \quad s = 1, \dots, S, \quad (33)$$

$$B_{\min} \leq B_s \leq B_{\max}, \quad s = 1, \dots, S-1. \quad (34)$$

where $\mathbf{w} = [W_1, \dots, W_S]^T$ is the machine vector, $\mathbf{x} = [x_1, \dots, x_S]^T$ is the task vector, $\mathbf{b} = [B_1, \dots, B_{S-1}]^T$ is the buffer capacity vector, and $E[TH(\mathbf{w}, \mathbf{x}, \mathbf{b})]$ depicts the expected throughput, which is the total number of products processed over the simulation time period.

Hence, the size of the search space is $(M_{\max} - M_{\min} + 1)^S \cdot T^S \cdot (B_{\max} - B_{\min} + 1)^{S-1}$. This BSCOP aims to determine the optimal $(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$ solution that maximizes expected throughput while simultaneously minimizing the total buffer capacity. Constraint (30) enforces that machine allocation at each workstation remains within the available number of machines. Constraint (31) indicates that the total number of allocated tasks to workstations equals the number of given tasks. Constraint (32) states that the machines allocated to each workstation cannot exceed the maximum number, and that a minimum number must be allocated to ensure production. Constraint (33) represents that at least one task should be assigned to each workstation. Constraint (34) describes that each interstation buffer should not exceed a specific safety capacity or a maximum buffer size. Given the customers' demands, the target is to find the optimal system configuration $\mathbf{w}^*$, task allocation $\mathbf{x}^*$, and buffer capacity $\mathbf{b}^*$, to maximize the expected throughput $E[TH(\mathbf{w}, \mathbf{x}, \mathbf{b})]$ and minimize the total buffer capacity $\sum_{s=1}^{S-1} B_s$. The sample mean is used to approximate $E[TH(\mathbf{w}, \mathbf{x}, \mathbf{b})]$.

$$E[TH(\mathbf{w}, \mathbf{x}, \mathbf{b})] = \frac{1}{L} \sum_{\ell=1}^L TH_{\ell}(\mathbf{w}, \mathbf{x}, \mathbf{b}) \quad (35)$$

where L is the number of replications, and $TH_{\ell}(\mathbf{w}, \mathbf{x}, \mathbf{b})$ depicts the throughput of the ℓ th replication. Since both constraints are soft, the BSCOP can be converted into an unconstrained one.

$$\min F(\mathbf{w}, \mathbf{x}, \mathbf{b}) = -\alpha \cdot \left(\frac{1}{L} \sum_{\ell=1}^L TH_{\ell}(\mathbf{w}, \mathbf{x}, \mathbf{b}) \right) + (1-\alpha) \cdot \sum_{s=1}^{S-1} B_s + \frac{\eta}{2} \cdot \left[\left(\max \left\{ 0, \sum_{s=1}^S W_s - M \right\} \right)^2 + \left(\sum_{s=1}^S x_s - T \right)^2 \right] \quad (36)$$

where α is a weighting parameter, η is a penalty parameter, $\left(\max \left\{ 0, \sum_{s=1}^S W_s - M \right\} \right)^2$ is the quadratic penalty function for inequality constraint, $\left(\sum_{s=1}^S x_s - T \right)^2$ is the quadratic penalty

function for equality constraint, and $F(\mathbf{w}, \mathbf{x}, \mathbf{b})$ denotes a penalized function.

Figure 4 shows the relationships between input-output in the production planning of RMS, where $(\mathbf{w}, \mathbf{x}, \mathbf{b})$ indicates the input variable, L depicts the number of replications, and $F(\mathbf{w}, \mathbf{x}, \mathbf{b})$ is the penalized function.

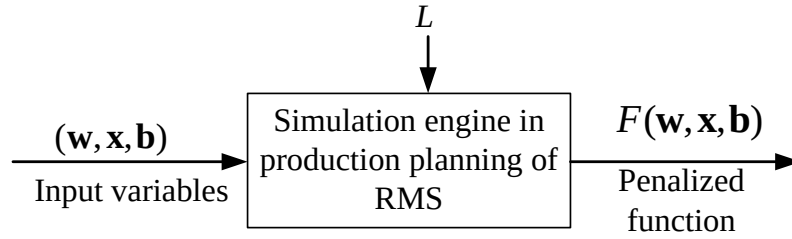


Fig. 4 Input and output relation in production planning of RMS.

4. Simulation Results

4.1 Practical Example

The featured RMS serves as a practical example from an automotive manufacturer in Taiwan, focusing specifically on a 4-cylinder crankshaft production line. This line produces two distinct product families and comprises 21 workstations. Its processes include unpacking, mass centering, milling, drilling, turning, deburring, heat treatment, grinding, mass balancing, washing, and quality control, all facilitated by 20 inter-station buffers. Workstations 14, 15, and 16, which handle milling, drilling oil holes, and turning bearings and taps, form the bottleneck of the production line and are pivotal to its performance. Unlike other machines on the line, these three workstations have a flexible configuration that allows for additions, removals, or relocations during future system revisions. This adaptability enables the system to scale up or down in response to fluctuations in production volumes or overall capacity requirements. Adjustments to the process plan and system configuration are made based on current scenarios, affecting factors such as the number of operating machines, task allocation across workstations, and buffer capacity. As demand fluctuates over time, workstations 14, 15, and 16 undergo continuous reconfiguration to meet evolving needs. These changes influence

not only the number of machines used, but also the nature of the tasks performed at these workstations. Furthermore, such reconfigurations influence the buffer capacities required to maintain workflow efficiency throughout the line.

Figure 5 illustrates the practical example of an RMS setup featuring two machines per workstation, with provision for adding up to four additional machines per workstation. For ease of understanding, workstations 1, 2, and 3 correspond to workstations 14, 15, and 16, respectively. Table 1 shows the parameters used in RMS. Available number of machines in RMS is $M = 18$. This RMS has a machining process that takes 16 minutes and is divided into $T=36$ tasks. Hence, the size of the search space is $5^3 \cdot 36^3 \cdot 16^2 = 1.49 \times 10^9$. The process is subject to real disruptions, with machine availability at 90% and an average repair time of 5 minutes. The machining process satisfies a demand following a Poisson distribution with a demand rate $\lambda = 550$ products/day in an 11-hour working day (50 products/hour for 11 working hours per day). Additionally, the buffers require 5 seconds of material-handling time for loading/unloading. The SimEvents library [38] was employed to model the production system for the simulation engine. The LWOO approach was implemented in MATLAB R2024a on Windows 11. Interested readers may refer to Ref. [39] for the source code of the LWOO approach.

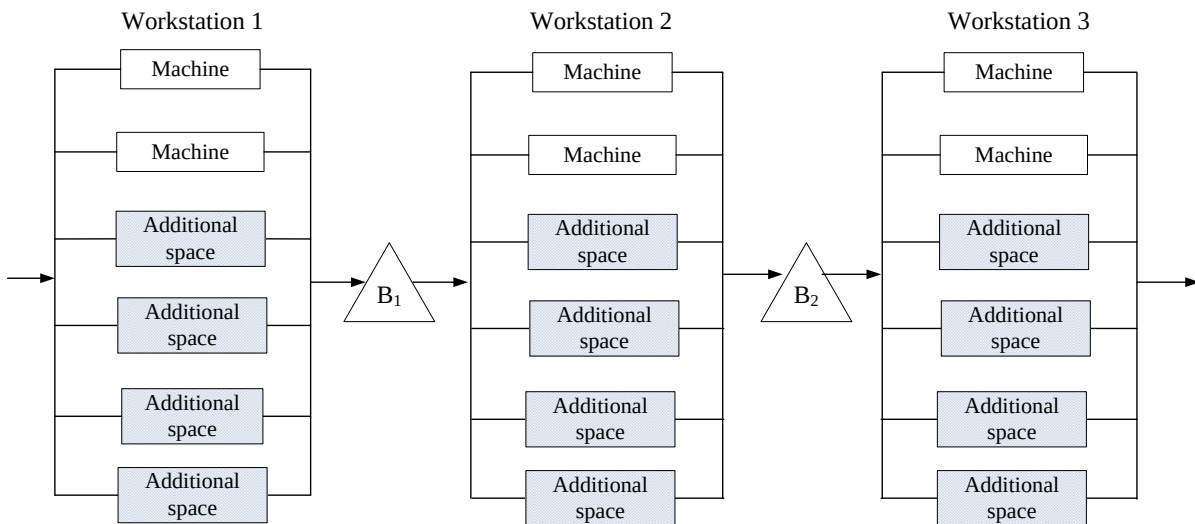


Fig. 5 Practical example of RMS.

Table 1. Parameters used in RMS.

Parameter	S	T	M	$M_{\max}$	$M_{\min}$	$B_{\max}$	$B_{\min}$
Value	3	36	18	6	2	20	5

The KAN was pre-trained off-line using $R=9604$ samples. Figure 6 shows a solution representation of the practical example. Since the population size is 1.49×10^9 , using a sample size calculator, $R=9604$ was derived under a 95% confidence level, 5% margin of error, and a population proportion of 0.5 [40]. A [8,17,1] KAN structure was employed, with 8 input, 17 hidden, and 1 output nodes. The following hyperparameters are used in the KAN: grid size (g) was 10, and the order of the polynomial (k) for the cubic B-splines was 3.

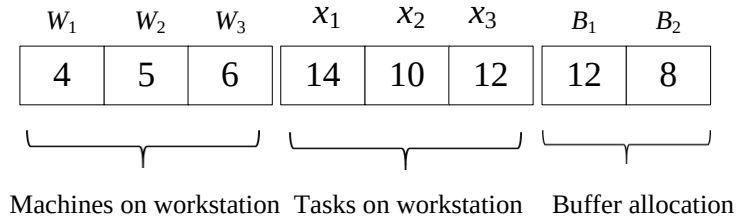


Fig. 6 Alternative representation for practical example.

The weighting parameter α was 0.6, and the penalty parameter η was 10^4 . The following parameters were used in RLWO: $\Psi=100$, $t_{\max}=500$, $C_{\min}=0.001$, $C_{\max}=0.8$, $P1_{\min}=0.1$, $P1_{\max}=0.3$, $P2_{\min}=0.05$, $P2_{\max}=0.2$, $P3_{\min}=0.05$, and $P3_{\max}=0.2$. Various hand-tuning experiments showed that these parameters ensure a balanced trade-off between throughput and the entire buffer capacity. Figure 7 displays the curves of the adaptive movement factor and three thresholds over 500 iterations. The adjustment function of factor C , threshold $P1$, and threshold $P2$, is an exponentially decreasing function. Large settings of C , $P1$, and $P2$ tend to emphasize diversification at early iterations, while small settings of C , $P1$, and $P2$ increase the local search tendency. The adjustment function for threshold $P3$ is an exponentially increasing function. The value of $P3$ is increased exponentially with the number

of iterations to progressively strengthen the intensification process. The LWOO was implemented with four values of N , which were $N=10, 20, 30$, and 40 , to examine the influence of N . The parameters utilized in AOCBA were $L_0=20$, $\Delta=10$, and $L_e=10^3$. The speedup factors for $N=10, 20, 30$, and 40 are $\nu=3.3, 6.1, 8.4$, and 10.7 [36], respectively. Therefore, the allowable simulation budgets regarding $N=10, 20, 30$, and 40 were $CB=3030, 3279, 3571$, and 3738 , respectively.

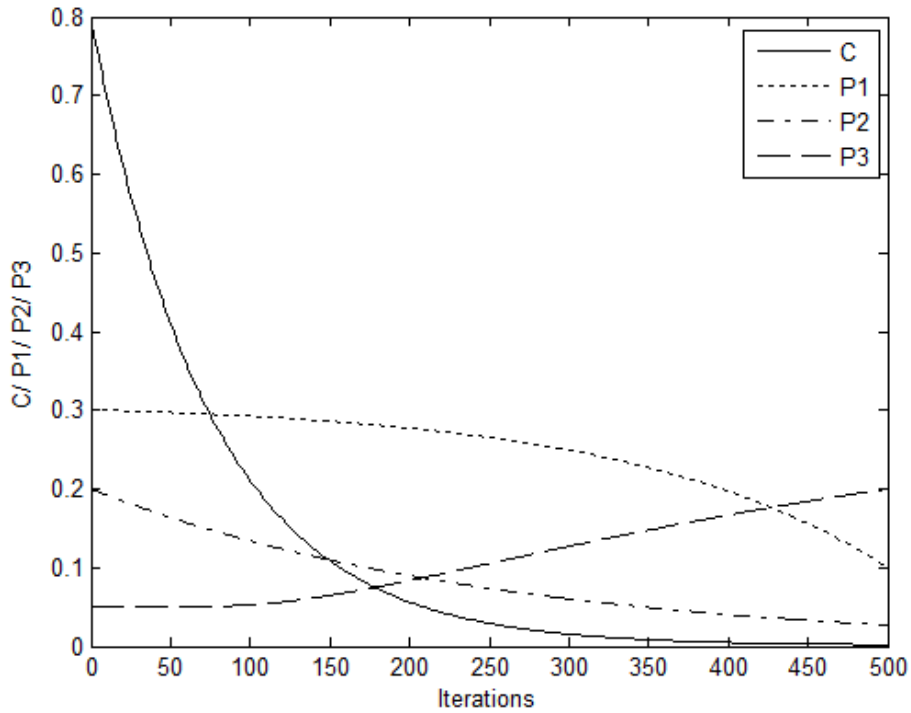


Fig. 7 Variations of $C, P1, P2$, and $P3$ over 500 iterations.

Table 2 shows the near-optimal solution $(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$, $E[TH(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)]$, $B_1 + B_2$, $F(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$, and CPU times of four cases for a typical run. For case $N=40$, workstations 1, 2, and 3 are allocated 3, 6, and 6 machines, respectively. The number of tasks performed in workstations 1, 2, and 3 is 6, 16, and 14, respectively. Two buffer capacities are 11 and 8, and the expected throughput is 50.20 (Jobs Per Hour). The CPU time for the four cases was less than 2 minutes, demonstrating that LWOO is suitable for real-time applications.

Table 2. Near-optimal solution $(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$, $E[TH(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)]$, $B_1 + B_2$, $F(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$, and CPU times of four cases for a typical run.

N	$\mathbf{w}^*$	$\mathbf{x}^*$	$\mathbf{b}^*$	$E[TH(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)]$ (Jobs Per Hour)	$B_1 + B_2$	$F(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$	CPU times (sec.)
10	$[2, 6, 5]^T$	$[4, 19, 13]^T$	$[7, 5]^T$	44.14	12	-21.68	98.83
20	$[6, 5, 5]^T$	$[16, 10, 10]^T$	$[14, 11]^T$	53.32	25	-21.99	104.15
30	$[2, 6, 6]^T$	$[4, 17, 15]^T$	$[9, 6]^T$	46.78	15	-22.07	111.47
40	$[3, 6, 6]^T$	$[6, 16, 14]^T$	$[11, 8]^T$	50.20	19	-22.52	117.23

A critical balance exists between the number of buffers and the resulting throughput. Since buffer space can be costly, it is preferable to reduce the total number of buffers utilized. However, throughput is directly influenced by the quantity of allocated buffers. Increasing the number of buffers typically increases overall throughput. All solutions for the four cases in Table 2 lie on the Pareto front. While case $N=20$ achieves higher throughput than case $N=40$, it requires more buffers. Due to the chosen weighting parameter, the penalized value for the case with $N = 40$ is lower than that for the case with $N = 20$.

To further examine the effectiveness of each phase, we conduct an ablation study investigating its performance involving two different solution strategies. (i) In the surrogate model, KAN was substituted with a coarse evaluation (CE) using $L_c = 10$, and the fitness of each leaf was evaluated accordingly. (ii) In local exploitation, AOCBA was substituted with a rigorous evaluation (EE) using $L_e = 10^3$, and all leaves were evaluated rigorously upon the termination of RLWO. By combining these two solution strategies, three algorithm variants are obtained: (CE+RLWO+EE), (CE+RLWO+AOCBA), and (KAN+RLWO+EE). They were compared against the LWO (KAN+RLWO+AOCBA) for case $N=40$. Table 3 summarizes the results of the ablation experiment for a typical run. The reported p -values were computed using a multiple-comparison test. While substituting KAN with CE yields a marginal performance improvement, it requires approximately 2.8 times the computational effort of

LWOO. Replacing AOCBA with a rigorous evaluation maintains the performance but requires approximately 25 percent additional time. AOCBA provides only a slight improvement in solution quality compared to rigorous evaluation, while KAN sacrifices a small amount of quality to achieve a faster search process. These findings demonstrate that trading off a small amount of solution quality can yield substantial savings in computational resources.

Table 3. Results of the ablation study and corresponding relative changes for a typical run.

Algorithm	Performance	Performance change (%)	p -value	CPU time (min.)	CPU time change (%)	p -value
KAN+RLWO+AOCBA	-22.52	0	-	1.95	0	-
CE+RLWO+EE	-22.86	-1.51	0.002	5.38	175.9	<0.001
CE+RLWO+AOCBA	-22.86	-1.51	0.002	4.82	147.2	<0.001
KAN+RLWO+EE	-22.52	0	0.42	2.44	25.1	0.004

4.2 Statistical Result Comparison

The performance of LWOO is compared with four heuristic approaches: the original LiWO, the genetic algorithm (GA), the artificial immune system (AIS), and ant colony optimization (ACO). The latter three heuristic approaches are often categorized as discrete optimization algorithms. The population size was set to 100 for all approaches. The original LiWO [19] uses constant thresholds with $P1=0.1$, $P2=0.1$, and $P3=0.3$. In GA [41], roulette wheel selection was adopted, with crossover and mutation rates set to 0.8 and 0.03, respectively. In AIS [42], clonal selection was used, with crossover and mutation rates set to 0.7 and 0.05, respectively. In ACO [43], the initial pheromone level was set to 1, with a pheromone evaporation rate of 0.4 and a deposition constant of 0.2; the influence coefficients for pheromone and proximity were set to 0.9 and 0.7, respectively. The objective values were estimated using coarse evaluation with $L_c=10$ for four heuristic approaches.

To account for the stochastic nature, the performance of each method is measured over 30 independent runs. The optimization process for the four heuristic approaches terminates after 6 minutes of CPU time. Figure 8 depicts the curves of best-so-far performance along with

error bars for five approaches evaluated over 30 independent runs. The “*” point at (1.96, -22.50) indicates the average CPU time and corresponding average performance achieved by LWOO. Table 4 shows that the average performances of LiWO, GA, AIS, and ACO are 7.07%, 13.24%, 11.73%, and 8.98% higher than that of LWOO, respectively, over 30 independent runs.

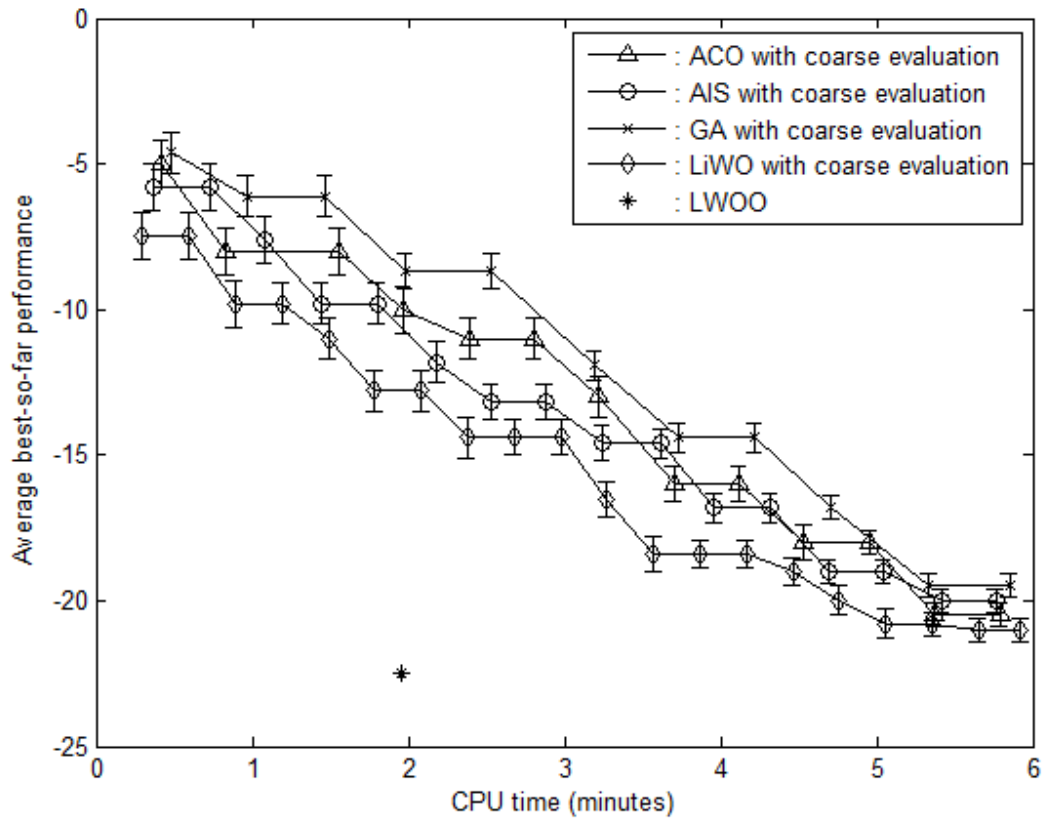


Fig. 8 Average best-so-far performance along with error bars for five methods over 30 independent runs.

Table 4. Comparison of average performance over 30 independent runs.

Approaches	AP [†]	$\frac{AP - *^{\S}}{*} \times 100\%$
LWOO	-22.50	0
LiWO with coarse evaluation	-20.91	7.07%
GA with coarse evaluation	-19.52	13.24%
AIS with coarse evaluation	-19.86	11.73%
ACO with the coarse evaluation	-20.48	8.98%

[†]AP: average performance

[§]*: average performance of LWOO

Moreover, the quality of the solutions was evaluated by analyzing their rankings. Because it

is difficult to evaluate and rank all solutions, a representative subset Ω is constructed as a substitute for the entire search space. First, $|\Omega|=16641$ solutions were randomly selected to form the representative subset from the search space. Since the population size is 1.49×10^9 , a sample size of approximately 16641 is required to estimate the population proportion at a 99% confidence level, with a margin of error of 1% and a population proportion of 0.5 [40]. A coarse evaluation was used to evaluate the performance of a solution. An analytical procedure was adopted to evaluate rank percentages and describe the ranking of near-optimal solutions in a representative subset. The percentile rank of a near-optimal solution within Ω is calculated as $\frac{rk}{|\Omega|} \times 100\%$, where rk indicates its rank. Table 5 presents descriptive statistics for five approaches, analyzed over 30 independent runs to provide a robust comparison of their performances. LWOO yielded a standard error of the mean (S.E.M.) of 0.0237. This small value reveals that the LWOO approach consistently produces near-optimal solutions across multiple trials, demonstrating high reliability and robustness.

Table 5. Descriptive statistics using five approaches over 30 independent runs.

Approach	Minimum	Maximum	Mean	S. D.	S.E.M.	Average percentile rank
LWOO	-22.91	-22.12	-22.50	0.13	0.0237	0.012%
LiWO with coarse evaluation	-21.49	-20.25	-20.91	0.19	0.0346	0.686%
GA with coarse evaluation	-20.68	-18.16	-19.52	0.38	0.0694	1.183%
AIS with coarse evaluation	-20.75	-18.89	-19.86	0.29	0.0529	0.964%
ACO with coarse evaluation	-21.19	-19.71	-20.48	0.24	0.0438	0.815%

Furthermore, the Wilcoxon rank-sum test is carried out at the 5% significance level to verify the significance of the experimental results [44]. The p -values and h -values are used as metrics. A p -value represents the probability of observing results as extreme as those measured, assuming the null hypothesis is true. An h -value of 1 corresponds to rejection of

the null hypothesis, whereas an h -value of 0 corresponds to failure to reject it. A p -value < 0.05 together with $h = 1$ indicates a statistically significant difference between the two approaches. Table 6 presents the results of the Wilcoxon rank-sum test comparing LWOO with four heuristic approaches. LWOO shows a statistically significant difference from the four heuristic approaches, indicating greater reliability and robustness.

Table 6. Results of the Wilcoxon rank-sum test.

Metric	LWOO vs. LiWO	LWOO vs. GA	LWOO vs. AIS	LWOO vs. ACO
p -value	0.012	5.54×10^{-4}	7.85×10^{-3}	2.69×10^{-3}
h -value	1	1	1	1

5. Conclusion

The LWOO method, a sim-heuristic approach that integrates LiWO with OO, was developed for efficiently solving the BSCOP. It consists of three phases: surrogate model, universal exploration, and local exploitation. The KAN surrogate model provides fast solution evaluation, while LWOO uses RLWO for universal exploration to improve diversity and AOCBA for local exploitation to strengthen intensification. The LWOO approach was applied for RMS production planning, which is formulated as a BSCOP. The goal is to find the optimal system configuration and task allocation to the workstation, as well as buffer allocation, to maximize expected throughput and minimize RMS's total buffer capacity. An RMS consisting of 3 workstations with 2 inter-station buffers, focusing on a 4-cylinder crankshaft production line, serves as a practical example. The performance of LWOO was evaluated against four heuristic approaches, LiWO, GA, AIS, and ACO, using a coarse evaluation. The average best-so-far performances of LiWO, GA, AIS, and ACO are 7.07%, 13.24%, 11.73%, and 8.98% higher than that of LWOO, respectively. An analytical procedure was applied to the rank percentages to assess solution quality. The S.E.M. of LWOO was 0.0237, suggesting that the LWOO method consistently yields near-optimal solutions across

multiple trials, thereby demonstrating high reliability and robustness.

The proposed LWOO is not limited to RMS applications. They could be extrapolated to untested directions for future work, such as reverse supply chain networks, facility location, reconfiguration in healthcare facilities, and logistics networks. Future research could explore extending the OO framework to address functionally constrained stochastic optimization problems, including insurance coverage decisions, crop production scheduling, retirement portfolio optimization, and stochastic multi-commodity network flow problems.

CRedit authorship contribution statement:

S.C. Horng: Writing – original draft, writing – review & editing, designed and performed the experiments, validation, software, and investigation.

Data Availability:

All data generated or analyzed during this study are included in the article.

Declarations of competing interests:

The authors declared no potential conflicts of interest.

Funding:

This research work is partly supported by the National Science and Technology Council in Taiwan, R.O.C., under Grant NSTC114-2221-E-324-011.

References

- [1] J. Ragnitz, B. Hofmeister, C. Jonscher, C. Hübler, R. Rolfes, “A stochastic multi-objective optimisation approach for damage localisation via model updating with uncertain input parameters”, *Engineering Structures*, Vol. 330, id. 119860, May 2025.
- [2] Y. Cao, Y.F. Mu, H.J. Jia, X.D. Yu, K. Hou, H.Y. Wang, “A multi-objective stochastic optimization approach for planning a multi-energy microgrid considering unscheduled islanded operation”, *IEEE Transactions on Sustainable Energy*, Vol. 15, No. 2, pp. 1300-1314, Apr. 2024.

- [3] K. Khalili-Damghani, M. Tavana, P. Ghasemi, “A stochastic bi-objective simulation-optimization model for cascade disaster location-allocation-distribution problems”, *Annals of Operations Research*, Vol. 309, No. 1, pp. 103-141, Feb. 2022.
- [4] X.J. Wang, F.G. Zhang, M.L. Yao, “Dynamic decomposition and hyper-distance based many-objective evolutionary algorithm”, *Complex & Intelligent Systems*, Vol. 11, No. 1, id. 85, Jan. 2025.
- [5] B.G. Thengvall, S.N. Hall, M.P. Deskevich, “Measuring the effectiveness and efficiency of simulation optimization metaheuristic algorithms”, *Journal of Heuristics*, Vol. 31, No. 1, id. 12, Mar. 2025.
- [6] Z.H. Li, W. Wu, Q.F. Zhang, X.Y. Cai, “Exploring high-dimensional optimization by sparse and low-rank evolution strategy”, *Swarm and Evolutionary Computation*, Vol. 92, id. 101828, Feb. 2025.
- [7] U. Asif, M.F. Javed, D.M. Alsekait, F. Aslam, D.S.A. Elminaam, “Data-driven evolutionary programming for evaluating the mechanical properties of concrete containing plastic waste”, *Case Studies in Construction Materials*, Vol. 21, id. e03763, Dec. 2024.
- [8] T. Stamadianos, A. Taxidou, M. Marinaki, Y. Marinakis, “Swarm intelligence and nature inspired algorithms for solving vehicle routing problems: a survey”, *Operational Research*, Vol. 24, No. 3, id. 47, Sep. 2024.
- [9] S. Si-Ma, H.M. Liu, H.X. Zhan, Z.F. Liu, G. Guo, C. Yu, P.C. Hu, “Efficient maximum iterations for swarm intelligence algorithms: a comparative study”, *Artificial Intelligence Review*, Vol. 58, No. 3, id. 87, Jan. 2025.
- [10] S. Szenasi, G. Légrádi, “Machine learning aided metaheuristics: A comprehensive review of hybrid local search methods”, *Expert Systems with Applications*, Vol. 258, id. 125192, Dec. 2024.

- [11] Y.P. Wang, B. Xin, B. Liu, Q. Wang, “Automated design of collaboration-based hybrid metaheuristics”, IEEE Transactions on Cybernetics, Vol. 54, No. 12, pp. 7877-7890, Dec. 2024.
- [12] C.A. Rodríguez-Espinosa, E.M. González-Neira, G.M. Zambrano-Rey, “A simheuristic approach using the NSGA-II to solve a bi-objective stochastic flexible job shop problem,” Journal of Simulation, Vol. 18, No. 4, pp. 646-670, Jul. 2024.
- [13] M. Peyman, X.A. Martin, J. Panadero, A.A. Juan, “A discrete-event simheuristic for enhancing urban mobility,” Simulation Modelling Practice and Theory, Vol. 140, id. 103084, Apr. 2025.
- [14] Y.C. Ho, Q.C. Zhao, Q.S. Jia, *Ordinal Optimization: Soft Optimization for Hard Problems*, Springer-Verlag, New York, 2007.
- [15] S.C. Horng, S.S. Lin, “Incorporate seagull optimization into ordinal optimization for solving the constrained binary simulation optimization problems”, Journal of Supercomputing, Vol. 79, No. 5, pp. 5730-5758, Mar. 2023.
- [16] S.C. Horng, S.S. Lin, “Accelerated driving-training-based optimization for solving constrained bi-objective stochastic optimization problems”, Mathematics, Vol. 12, No. 12, id. 1863, Jun. 2024.
- [17] S.C. Horng, S.S. Lin, “Advanced golden jackal optimization for solving the constrained integer stochastic optimization problems”, Mathematics and Computers in Simulation, Vol. 217, pp. 188-201, Mar. 2024.
- [18] S.C. Horng, S.S. Lin, “Adapted skill optimization algorithm to solve constrained discrete stochastic optimization problems”, Mathematics and Computers in Simulation, Vol. 238, pp. 280-295, Dec. 2025.
- [19] N. Fang, Q. Cao, “Leaf in wind optimization: A new metaheuristic algorithm for solving optimization problems”, IEEE Access, Vol. 12, pp. 56291-56308, Apr. 2024.

- [20] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T.Y. Hou, M. Tegmark, “KAN: Kolmogorov–Arnold Networks”, in Proc. International Conference on Learning Representations (ICLR), 2025.
- [21] B. Chaymae, E. Abdellah, “A reconfigurable manufacturing systems: Enhancing efficiency via product family optimization”, *Journal of Manufacturing and Materials Processing*, Vol. 9, No. 2, id. 39, Feb. 2025.
- [22] P. Rezaee, S.K. Moghaddam, “Assessment of design approaches for reconfigurable manufacturing systems based on forecasted demand data”, *Computers & Industrial Engineering*, Vol. 201, id. 110878, Mar. 2025.
- [23] J.S. Park, J.K. Byun, “Multi-objective optimization of slipring-less winding excited synchronous machine using weighted sum method”, *Journal of the Korean Magnetics Society*, Vol. 32, No. 1, pp. 32-38, Feb. 2022.
- [24] H.T. Liao, W.H. Yuan, R.X. Gao, X.J. Yuan, “An efficient penalty function method for scenario-based uncertainty quantification problems”, *Journal of Vibration and Control*, Vol. 31, No. 3-4, pp. 499-515, Feb. 2025.
- [25] D. Fehrle, C. Heiberger, J. Huber, “Polynomial chaos expansion: Efficient evaluation and estimation of computational models”, *Computational Economics*, Vol. 65, No. 2, pp. 1083-1146, Feb. 2025.
- [26] I. Perfilieva, N. Madrid, M. Ojeda-Aciego, P. Artiemjew, A. Niemczynowicz, “A critical analysis of the theoretical framework of the extreme learning machine”, *Neurocomputing*, Vol. 621, id. 129298, Mar. 2025.
- [27] A. Dalal, M. Bagherimehrab, B.C. Sanders, “Quantum-assisted support vector regression,” *Quantum Information Processing*, Vol. 24, No. 3, id. 82, Mar. 2025.
- [28] J.B. Zhang, X.J. Cai, Z.H. Cui, J.J. Chen, “A physics-informed neural network surrogate model and many-objective optimization algorithm for coupled multi-energy systems in

- smart grids”, *Expert Systems with Applications*, Vol. 298, id. 129760, Mar. 2026.
- [29] L. Chu, J.J. Shi, E.S. de Cursi, “A kriging surrogate model for ball grid array electronic packaging with stochastic material and geometrical parameters”, *Journal of Electronic Packaging*, Vol. 147, No. 1, id. 011003, Mar. 2025.
- [30] S. Somvanshi, S. A. Javed, M. M. Islam, D. Pandit, S. Das, “A survey on Kolmogorov-Arnold network”, *ACM Computing Surveys*, Vol. 58, No. 2, id. 55, Sep. 2025.
- [31] V. Sellam, N. Kannan, S.S. Pandi, I. Manju, “Enhancing sustainable agriculture using attention convolutional bidirectional Gated recurrent based modified leaf in wind algorithm: Integrating AI and IoT for efficient farming”, *Sustainable Computing: Informatics and Systems*, Vol. 47, id. 101160, Sep. 2025.
- [32] A. Thiyagarajan, C. Murukesh, S.M. Joans, “Enhancing diagnostic precision and mammogram classification in breast cancer detection utilizing an optimized explicit feature interaction aware graph neural network”, *Biomedical Signal Processing and Control*, Vol. 112, id. 108444, Feb. 2026.
- [33] N.J. Divya, N.S. Kumar, R.K. Devi, “An optimized progressive attention-based bidirectional encoder enclosed transformer network-based cardiovascular disease detection”, *AI, Computer Science and Robotics Technology*, Vol.4, No. 1, pp. 1-32, Jun. 2025.
- [34] M. Amanullah, K. Ananthajothi, D. Divya, “Enhanced smart weather prediction through advanced atmospheric analysis and forecasting techniques using binarized spiking neural networks”, *Knowledge and Information Systems*, Vol. 67, pp. 3343-3372, Jan. 2025.
- [35] K. Kassoul, N. Zufferey, N. Cheikhrouhou, S.B. Belhaouari, “Exponential particle swarm optimization for global optimization”, *IEEE Access*, Vol. 10, pp. 78320-78344, 2022.
- [36] C.H. Chen, L.H. Lee, *Stochastic Simulation Optimization: An Optimal Computing*

Budget Allocation, World Scientific, New Jersey, 2010.

- [37] M. Mastrangelo, T.A.M. Tolio, “A hybrid method combining analytical and simulation models for performance evaluation of reconfigurable manufacturing systems”, *Journal of Manufacturing Systems*, Vol. 76, pp. 259-280, Oct. 2024.
- [38] Y. Zhang, C.G. Cassandras, W. Li, P.J. Mosterman, “A discrete-event and hybrid traffic simulation model based on SimEvents for intelligent transportation system analysis in Mcity,” *Discrete Event Dynamic Systems-Theory and Applications*, Vol. 29, No. 3, pp. 265-295, Sep. 2019.
- [39] S.C. Horng, “Sim-heuristic algorithm for solving bi-objective stochastic optimization problems”, *Online Appendix*, <https://www.cyut.edu.tw/~schong/eng/technical.htm>, Jun. 2026.
- [40] T.P. Ryan, *Sample Size Determination and Power*, John Wiley and Sons, New Jersey, 2013.
- [41] S.B. Dalkılıç, A. Özgür, H. Erdem, “Balancing exploration and exploitation in genetic algorithm optimization: a novel selection operator”, *Evolutionary Intelligence*, Vol. 18, No. 2, id. 40, Apr. 2025.
- [42] J. Molina, L.F. Valdez, J.M. Gutiérrez, “An artificial immune system algorithm for classification tasks. An electronic nose case study”, *Engineering Applications of Artificial Intelligence*, Vol. 133, id. 108457, Jul. 2024.
- [43] Z.Z. Shen, J. Wu, “Multiobjective ant colony system algorithm for component-level construction schedule optimization”, *Journal of Construction Engineering and Management*, Vol. 151, No. 3, id. 04025002, Mar. 2025.
- [44] J. Derrac, S. García, D. Molina, F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms”, *Swarm and Evolutionary Computation*, Vol. 1, No. 1, pp. 3-18,

2011.

List of Figures

		On page	First reference page
Fig. 1	Framework of a KAN	9	8
Fig. 2	Flow chart of the LWOO approach.	20	18
Fig. 3	Reconfigurable manufacturing system	21	21
Fig. 4	Input and output relation in production planning of RMS.	24	24
Fig. 5	Practical example of RMS.	25	25
Fig. 6	Alternative representation for Practical example	26	26
Fig. 7	Variations of C , $P1$, $P2$, and $P3$ over 500 iterations.	27	26
Fig. 8	Average best-so-far performance along with error bars for five methods over 30 independent runs...	30	29

List of Tables

		On page	First reference page
Table 1	Parameters used in RMS.	26	25
Table 2	Near-optimal solution $(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$, $E[TH(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)]$, $F(\mathbf{w}^*, \mathbf{x}^*, \mathbf{b}^*)$, and CPU times for four cases for a typical run	28	27
Table 3	Ablation experiment results and relative performance changes for a typical run	29	28
Table 4	Comparison of average performance over 30 independent runs....	30	30
Table 5	Descriptive statistics using five approaches over 30 independent runs.. . . .	31	31
Table 6	Results of the Wilcoxon rank-sum test.	32	31