Machine-Level Collaborative Manufacturing and Scheduling for Heterogeneous Plants

Haitao Yuan, *Senior Member, IEEE*, Qinglong Hu, *Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Guanghong Gong, Jia Zhang, *Senior Member, IEEE*, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Current Industrial Internet supports the sharing of information on heterogeneous resources and elements in a process of industrial production. It enables intelligent production processes and supports cost-effective scheduling. However, collaborative manufacturing and scheduling planning for enterprises with multiple plants cause several major challenges because of a large number of decision variables and constraints of manufacturing abilities of plants, resources of production, *etc*. Existing methods cannot comprehensively optimize the cost of multiple products in different plants, and fail to consider machine-level optimization of tasks of manufacturing. We propose a comprehensive machine-level architecture for enterprises with multiple plants. Based on this architecture, we formulate a limited nonlinear integer optimization problem to decrease the total cost of transportation, production, and sales. In it, several real-life complicated nonlinear constraints are jointly considered, and they include constraints of storage space, replacement times, pairing production, substitution, and order fulfillment rates. To solve this optimization problem, we design a hybrid meta-heuristic optimization algorithm named genetic simulated annealing-based particle swarm optimizer with auto-encoders (GSPAE). Extensive experiments with real-life data show that GSPAE decreases the total cost by 25% than other state-of-the-art methods.**

***Index Terms*—Autoencoder (AE), collaborative manufacturing and scheduling (CMS), cost optimization, genetic algorithm (GA), particle swarm optimization (PSO), simulated annealing (SA).**

I. Introduction

INDUSTRIAL Internet provides a network platform to connect plants, warehouses, equipment, enterprises, customers, and products [1]. It provides different elements and shares resource information of a whole industrial production process [2], and realizes networked, digital, intelligent, and automated industrial production processes [3]. Accordingly, a collaborative manufacturing and scheduling (CMS) system composed of multiple heterogeneous plants is operated in a unified manner, which enables the complete sharing of manufacturing resources and capabilities [4].

To reduce the total cost of CMS systems, multiple plants need to be organized as a whole. Thus, a system needs to develop more efficient and cost-effective plans involving manufacturing, transportation, and sales [5]. Manufacturing planning (MP) aims to effectively schedule all production activities in the system during a planning period, and it is highly important to supply capacities, production efficiency, and production cost of CMS systems [6]. Transportation planning (TP) enables all kinds of raw materials, semi-finished products, finished ones, and other resources to be transmitted in all plants according to MP, thus avoiding the increase of transportation cost caused by redundant transportation [7]. Sales planning (SP) aims to design the optimal delivery of finished products to customers based on the current storage situation of each plant in the system, thereby minimizing the cost of sales of finished products to customers [8].

However, CMS problems suffer from several big challenges [9]. First, many real-life production resources and manufacturing capacities are limited. Second, there are a huge number of decision variables. Therefore, MP problems in the current industrial Internet become highly challenging and complicated. Several existing methods are designed to solve them [10], [11], [12], [13]. For example, Marinho de Brito et al. [10] proposed an optimization method integrating the p-median and mixed-integer linear programs. It determines deploying decisions of productive resources in specific locations of generic spare part supply chains. Hao et al. [11] designed an optimization algorithm based on deep reinforcement learning, thus minimizing acquisition latency in an intelligent and fiber-driven 6G fabric computing network. Rahman et al. [12] presented a hybrid algorithm based on genetic algorithm (GA) and particle swarm optimization (PSO) to simultaneously optimize the acceptance of orders and provide real-time operations of a make-to-order flow shop production system. Han and Yang [13] adopted a dueling double Q network with a prioritized replay, which allows an agent to continually interact with a scheduling environment through trial and error, thereby solving job shop scheduling problems with limited and real-time responsiveness. However,

Manuscript received 6 April 2023; revised 4 August 2023 and 2 November 2023; accepted 10 January 2024. Date of publication 16 January 2024; date of current version 25 April 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62173013 and Grant 62073005; in part by the Beijing Natural Science Foundation under Grant 4232049; and in part by the Fundamental Research Funds for the Central Universities under Grant YWF-23-03-QB-015. This article was presented in part at the 2022 IEEE International Conference on Systems, Man, and Cybernetics, Prague, Czech Republic [DOI: 10.1109/SMC53654.2022.9945305]. *(Corresponding author: Haitao Yuan.)*

Haitao Yuan, Qinglong Hu, and Guanghong Gong are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn; qlhu_ekite@buaa.edu.cn).

Jing Bi is with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

Jia Zhang is with the Department of Computer Science, Southern Methodist University, Dallas, TX 75206 USA (e-mail: jiazhang@smu.edu).

MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/JIOT.2024.3354251

they only investigate MP problems in a single plant, and their methods are unsuitable for the CMS problems.

This work proposes a comprehensive machine-level CMS framework for systems comprising multiple heterogeneous plants. A hybrid meta-heuristic optimization algorithm named genetic simulated annealing-based particle swarm optimization with auto-encoders (GSPAE) is developed to address the CMS problem. To summarize, this work makes the following contributions to the field of CMS in multiple heterogeneous plants.

1) It formulates a comprehensive and nonlinear constrained integer optimization problem for minimizing the cost of production, transportation, and sales in heterogeneous plants. Several complicated nonlinear constraints, e.g., limits of storage space, replacement times, substitution, order fulfillment rates, and pairing production, are considered. In addition, it ensures machine-level scheduling for heterogeneous machines with different manufacturing capacities while also accounting for the cost of maintenance and operations. It also takes into account the production time and cost of products in heterogeneous machines.
2) It designs a hybrid meta-heuristic optimization algorithm named GSPAE. GSPAE combines GA's genetic operations [14], the conditional acceptance mechanism of simulated annealing (SA) [15], and PSO [16]. In addition, an autoencoder (AE) [17] is integrated to capture distribution features of particles' positions for strong global optimization abilities. Extensive experiments with real-life data are given to prove its performance in terms of convergence speed and accuracy. Results reveal that GSPAE reduces the cost by more than 25% compared with other state-of-the-art algorithms.

The organization of this article is given here. Section II discusses related work. Section III introduces the framework of the system. Section IV presents the constrained cost optimization problem. Section V presents details of GSPAE. Section VI gives the performance evaluation and Section VII draws the conclusion.

II. Related Work

A. Cost Optimization in Manufacturing Planning

Recently, more and more emerging studies have been proposed to optimize the cost of planning in manufacturing systems [5], [18], [19], [20], [21]. Leng et al. [18] proposed an iterative bi-level optimization model to reduce inconsistency between the whole planning and local one in personalized manufacturing systems, thereby decreasing the cost of manufacturing. However, it does not consider manufacturing efficiency differences among heterogeneous machines. Wang et al. [5] proposed an interactive optimization algorithm to obtain the best tradeoff between order fulfillment rate and cost. A multiobjective integer program is formulated and addressed by a two-stage algorithm. However, its MP is coarse-grained, and it ignores the machine-level scheduling and the selling cost. Chen and Wang [19] formulated a mathematical problem of capacity planning and production control, and solve it with an integer programming method to jointly optimize them. However, it considers the average production cost and ignores the material cost of different products. Rubaiee et al. [20] considered an energy-aware scheduling method for a manufacturing factory where energy prices vary in a real-time manner. Several variants of GA are developed to reduce tardiness and energy cost. However, it does not consider the product cost in a manufacturing environment. Li et al. [21] provided a large-scale scheduling method to minimize the energy consumed by flexible manufacturing systems. To solve it, a method derived from dynamic programming is designed to efficiently realize complex scheduling. However, it does not consider the cost of manufacturing and transportation.

Different from these methods, we take into account the heterogeneous cost of various products across multiple plants. In particular, the price of raw materials for each product varies depending on the locations of plants. Furthermore, the manufacturing time for each product and the cost of electricity and maintenance during operations vary based on the machines used. Additionally, our method focuses on the production capacities of products at the machine level, rather than at the plant level. Besides, it jointly considers the scheduling and allocation of manufacturing tasks at the machine level, which enhances the comprehensiveness and applicability of the system.

B. Scheduling in Multiplant Enterprises

An increasing number of methods are proposed to consider scheduling in multiple-plant enterprises. Pan et al. [22] formulated a flow shop scheduling problem for distributed permutation of lot-streaming with different applications in realistic manufacturing systems. Then, jobs are optimally allocated to different distributed factories for minimizing the maximum execution time of the system. Wang and Wang [23] solved a problem of hybrid flow-shop scheduling with heterogeneous factories. A mixed linear integer program is formulated and handled by a bi-population cooperative memetic algorithm. Different from them, we focus on minimizing the total cost of multiplant enterprises. Liu et al. [24] considered a many-objective job-shop scheduling problem of jointly optimizing five objectives, including machine loss, production cost, total tardiness, advance time, and completion time. Specifically, a multipopulation and multiobjective framework based on GA is proposed. Different from it, this work considers the manufacturing cost, transportation one, and sales one. Ma et al. [4] designed a production scheduling framework following a collaborative paradigm with edge and cloud. Based on this framework, a cloud periodically predicts the completion time of production tasks, and tasks are scheduled efficiently and accurately. Different from it, we aim to reduce the cost of multiplant enterprises. Lin et al. [25] develop an intelligent framework for manufacturing factories in edge computing. The scheduling problem of the job shop is investigated and solved by a deep Q network with this framework. Different from it, we formulate a total cost minimization problem as a constrained nonlinear integer optimization one, which considers three major types of cost. Yuan et al. [26] considered

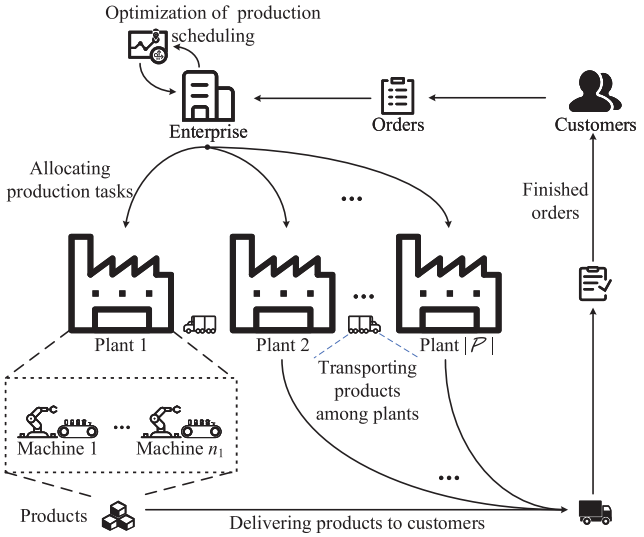


Fig. 1. Illustrative system architecture.

small-scale manufacturing problems with low-dimensional decision spaces and they consider many complex and real-life nonlinear constraints, such as limits of replacement times, storage space, substitution, and pairing productions. However, they fail to take into account the order fulfillment rates in their objective functions.

Different from them, we jointly consider the TP among different plants in a CMS system, the selling optimization among plants and customers, and the collaborative MP among multiple plants. We aim to reduce the cost of production, transportation, and sales while meeting the need for an order fulfillment rate. In addition, our method enables multiple plants to collaborate with each other more efficiently, Then, it enables plants to leverage each other's strengths and mitigate their weaknesses, and results in better coordination across multiple plants.

III. Research Motivation

Fig. 1 presents an architecture of the CMS system. The framework includes four major roles, including an enterprise, customers, a transportation system, and multiple plants. $\mathcal{P}$ is a plant set, and there are $|\mathcal{P}|$ plants. There are n_p machines in a plant p $(1 \leq p \leq |\mathcal{P}|)$. The raw materials, semi-finished products, and finished products are transmitted among plants and delivered to customers through the transportation system. A transaction is finished as follows. A customer sends an order to the enterprise, which determines a plan of manufacturing and scheduling by considering product types and quantity, and the delivery time of the order. Then, the enterprise schedules tasks of production to machines in different plants and delivers them to the transportation system. At last, the transportation system directly transmits products from plants to customers to complete orders.

Here, this work adopts "product" to generally refer to all production components, e.g., semi-finished products, raw materials, and completed products. The scheduling of production includes the production plan in different plants, the replacement plan for each product, the products' transportation plan among plants, and the delivered product number from different plants to customers. It is worth noting that different plants have heterogeneous characteristics. First, the manufacturing capacities of each plant for producing different products are different, which leads to heterogeneous time and material costs for manufacturing the same product. Second, due to differences in distances among multiple plants and those between plants and customers, each plant has different transportation cost during the delivery of products. Third, there are complex relations among some products, e.g., some products can be replaced by other ones, such as a large resistor, can be replaced by two small resistors in series, or they can be consumed by manufacturing processes of other ones. Therefore, it is challenging to properly handle such relations among products for reducing the cost of the CMS system. This work aims to obtain the production scheduling strategy that minimizes the total cost by using GSPAE.

IV. Problem Formulation

This section formulates a constrained cost optimization problem for the CMS system. Main notations and their definitions of our total cost minimization problem are summarized in Table I.

A. Optimization Variables

T denotes the period length of completing all orders. $w^j_{i,p,t}$ $(1 \leq t \leq T)$ is a binary decision variable. If product i is scheduled to machine j of plant p in time slot t, $w^j_{i,p,t} = 1$; otherwise, $w^j_{i,p,t} = 0$. $z_{i,p,t}$ is the number of product i delivered to customers in plant p in t. $s^{p'}_{i,p,t}$ is the number of product i delivered from p' to p in t. $x_{i,p,t}$ is the production batch number of product i of plant p in t. $h^{i'}_{i,p,t}$ is the number of product i' adopted to replace product i in t.

The product i of plant p in t can only be scheduled to at most one machine. Then

$$\gamma_{i,p,t} = \sum_{j=1}^{n_p} w^j_{i,p,t} \tag{1}$$

$$\gamma_{i,p,t} \leq 1 \tag{2}$$

where $\gamma_{i,p,t}$ is binary variable. If product i is produced in plant p at t, $\gamma_{i,p,t} = 1$; otherwise, $\gamma_{i,p,t} = 0$. n_p is the number of machines in plant p.

$R_{i,p,t}$ is the maximum number of replacement times for product i that can be replaced by other ones of plant p in t. Then, we have

$$\sum_{i' \in \mathcal{I}} h^{i'}_{i,p,t} \leq R_{i,p,t} \tag{3}$$

where $\mathcal{I}$ is a set of product types, and $|\mathcal{I}|$ is the number of product types.

B. Production, Sales, and Storage Models

$\Gamma_{i,p,t}$ is the number of product i in stock of plant p in t. $\Gamma_{i,p,t}$ is nonnegative obtained as

$$\Gamma_{i,p,t} = \begin{cases} E_{i,p} + \acute{\Delta}_{i,p,t} - \grave{\Delta}_{i,p,t}, & t = 0 \\ \Gamma_{i,p,t-1} + \acute{\Delta}_{i,p,t} - \grave{\Delta}_{i,p,t}, & \text{otherwise} \end{cases} \tag{4}$$

TABLE I
MAIN NOTATIONS AND DEFINITIONS OF OUR PROBLEM

Notation	Definition
T	Period length of completing all orders.
$w^j_{i,p,t}$	Binary decision variable. If product i is scheduled to machine j of plant p in time slot t, $w^j_{i,p,t}$=1; otherwise, $w^j_{i,p,t}$=0.
$z_{i,p,t}$	Number of product i delivered to customers in plant p in t.
$s^{p'}_{i,p,t}$	Number of product i delivered from p' to p in t.
$x_{i,p,t}$	Production batch number of product i of plant p in t.
$h^{i'}_{i,p,t}$	Number of product i' adopted to replace product i in t.
$\gamma_{i,p,t}$	Binary variable. If product i is produced in plant p at t, $\gamma_{i,p,t}$ =1; otherwise, $\gamma_{i,p,t}$ =0.
n_p	Number of machines in plant p.
$R_{i,p,t}$	Maximum number of replacement times for product i that can be replaced by other ones of plant p in t.
$\mathcal{I}$	Set of product types.
$\lvert\mathcal{I}\rvert$	Number of product types.
$\Gamma_{i,p,t}$	Number of product i in stock of plant p in t.
$E_{i,p}$	Number of product i in stock of plant p.
$\acute{\Delta}_{i,p,t}$	Increased numbers of product i of plant p in t.
$\grave{\Delta}_{i,p,t}$	Decreased numbers of product i of plant p in t.
$B^i_{i',p,t}$	Number of product i consumed in producing a product i' in p in t.
$\Theta_{i,p,t}$	Production limit of product i in p in t.
Φ_i	Number of i in each batch.
$U_{i,p}$	Storage space needed by i in p.
Λ_p	Maximum limit of storage space in p.
$\pi^{i'}_{i,p}$	Relation of product i' and product i.
f_i	Actual order fulfillment rate for product i.
$\hat{f}_i$	Lower limit of f_i.
$u_{i,t}$	Order fulfillment rate of i in t.
$O_{i,t}$	Quantity of i ordered by customers in t.
ε	Constant to avoid having a denominator of 0.
$m_{i,t}$	Quantity of i that is delayed in shipment in t.
M_i	Quantity of i delayed in shipment when t=0.
λ	Cost of production, transportation, and sales of all customer orders.
λ_1	Production cost of all customer orders.
λ_2	Transportation cost of all customer orders.
λ_3	Sales cost of all customer orders.
$\psi^j_{i,p,t}$	Production cost of i in machine j in p in t.
$\Psi_{i,p,t}$	Production cost of i in all machines in p in t.
$b^j_{i,p}$	Production cost per unit time of product i in j of p.
$a^j_{i,p}$	Material cost of product i in j of p.
$q^j_{i,p}$	Production time of i in j of p.
$\chi^{p'}_{i,p}$	Cost in delivering product i from plant p' to plant p.
$H_{i,p}$	Cost of sales of product i that plant p provides to customers.
$\widetilde{\lambda}$	New objective of the unconstrained problem.
$\hbar$	Decision vector.
$\overset{\infty}{\mathcal{N}}$	Large constant.
Ω	Total penalties of all constraints.
$\mathbb{N}^=$	Number of equality constraints.
$\mathbb{N}^{\neq}$	Number of inequality constraints.

where $E_{i,p}$ is the number of product i in stock of plant p. $\acute{\Delta}_{i,p,t}$ and $\grave{\Delta}_{i,p,t}$ are increased and decreased ones of product i of plant p in t, respectively, which are calculated as

$$\acute{\Delta}_{i,p,t} = \hat{x}_{i,p,t}\gamma_{i,p,t} + \sum_{p'\in\mathcal{P}} s^{p'}_{i,p,t} + \sum_{i'\in\mathcal{I}} h^{i'}_{i,p,t} \tag{5}$$

$$\grave{\Delta}_{i,p,t} = z_{i,p,t} + \sum_{p'\in\mathcal{P}} s^{p}_{i,p',t} + \sum_{i'\in\mathcal{I}} h^{i}_{i',p,t} + \sum_{i'\in\mathcal{I}} B^i_{i',p,t}\hat{x}_{i',p,t}\gamma_{i,p,t}. \tag{6}$$

In (5), for each product i, its three components include its produced number, that delivered from other plants to plant p, and that replaced by other products in plant p in t, respectively. In (6), for each product i, its four components include its number transmitted to customers, that transmitted from p to other plants, that adopted to replace other products, and that consumed in producing other products in p in t, respectively. $B^i_{i',p,t}$ is the number of product i consumed in producing a product i' in p in t. $\hat{x}_{i,p,t}$ is obtained as

$$\hat{x}_{i,p,t} = \begin{cases} 0, & \text{if } x_{i,p,t}\Phi_i \le \Theta_{i,p,t} \\ x_{i,p,t}\Phi_i, & \text{otherwise} \end{cases} \tag{7}$$

where $\Theta_{i,p,t}$ is a production limit of product i in p in t, i.e., product i cannot be produced if its number is less than or equal to $\Theta_{i,p,t}$, and Φ_i is the number of i in each batch.

For plant p, storage space needed by its all produced products cannot be larger than its maximum limit Λ_p. Thus

$$\sum_{i\in\mathcal{I}} U_{i,p}\Gamma_{i,p,t} \le \Lambda_p \tag{8}$$

where $U_{i,p}$ denotes the storage space needed by i in p.

C. Product Relation Limits

In addition, the number of product i' produced in p has to be matched with that of its related product i. For example, a charger needs to be produced for a phone. The relation of i' and i is $\pi^{i'}_{i,p}$. Thus

$$\hat{x}_{i',p,t} = \pi^{i'}_{i,p}\hat{x}_{i,p,t}. \tag{9}$$

In addition, the number of i consumed in p in t cannot be smaller than that of other products used to replace it. Thus

$$z_{i,p,t} + \sum_{i'\in\mathcal{I}} B^i_{i',p,t}\hat{x}_{i',p,t}\gamma_{i,p,t} \ge \sum_{i'\in\mathcal{I}} h^{i'}_{i,p,t}. \tag{10}$$

D. Order Fulfillment Rate

The order fulfillment rate is an important metric in the intelligent manufacturing system. An order can be completed only when the order fulfillment rate exceeds the given minimum limit. f_i denotes the actual order fulfillment rate for product i, and $\hat{f}_i$ denotes its lower limit, i.e.,

$$f_i \ge \hat{f}_i. \tag{11}$$

f_i is calculated as

$$f_i = \frac{\sum_{t=1}^{T} u_{i,t}}{T}. \tag{12}$$

In (12), $u_{i,t}$ is an order fulfillment rate of i in t, and it is calculated as

$$u_{i,t} = \max\left\{\frac{\left(\sum_{p\in\mathcal{P}} z_{i,p,t}\right) - m_{i,t-1} + \varepsilon}{O_{i,t} + \varepsilon}, 0\right\}. \tag{13}$$

In (13), $O_{i,t}$ is the quantity of i ordered by customers in t. ε denotes a constant to avoid having a denominator of 0. $m_{i,t}$

is the quantity of i that is delayed in shipment in t, which is given as

$$m_{i,t} = \begin{cases} M_i + O_{i,t} - \sum_{p\in\mathcal{P}} z_{i,p,t}, & t = 0 \\ m_{i,t-1} + O_{i,t} - \sum_{p\in\mathcal{P}} z_{i,p,t}, & t \geq 0 \end{cases} \tag{14}$$

where M_i denotes the quantity of i that is delayed in shipment when $t = 0$.

E. Cost Model

λ is the cost of production, transportation, and sales of all customer orders. In addition, λ_1, λ_2, and λ_3 are production, transportation, and sales cost, respectively. Thus

$$\lambda = \lambda_1 + \lambda_2 + \lambda_3. \tag{15}$$

λ_1 is calculated as

$$\lambda_1 = \sum_{i\in\mathcal{I}} \sum_{p\in\mathcal{P}} \sum_{t=1}^{T} \left(\hat{x}_{i,p,t} \Psi_{i,p,t}\right) \tag{16}$$

$$\Psi_{i,p,t} = \sum_{j=1}^{n_p} \left(\psi^j_{i,p,t} w^j_{i,p,t}\right) \tag{17}$$

$$\psi^j_{i,p,t} = \left(a^j_{i,p} + q^j_{i,p} b^j_{i,p}\right) w^j_{i,p,t} \tag{18}$$

where $\psi^j_{i,p,t}$ and $\Psi_{i,p,t}$ are production cost of i in machine j, and that of i in all machines in p in t. $b^j_{i,p}$, $a^j_{i,p}$, and $q^j_{i,p}$ are production cost per unit time, material cost, the production time of i in j of p.

λ_2 is obtained as

$$\lambda_2 = \sum_{i\in\mathcal{I}} \sum_{p'\in\mathcal{P}} \sum_{p\in\{\mathcal{P}\setminus p'\}} \chi^{p'}_{i,p} \sum_{t=1}^{T} s^{p'}_{i,p,t} \tag{19}$$

where $\chi^{p'}_{i,p}$ is the cost in delivering product i from plant p' to plant p.

λ_3 is obtained as

$$\lambda_3 = \sum_{i\in\mathcal{I}} \sum_{p\in\mathcal{P}} H_{i,p} \sum_{t=1}^{T} z_{i,p,t} \tag{20}$$

where $H_{i,p}$ denotes the cost of sales of product i that plant p provides to customers.

F. Cost Minimization Problem

Our objective is to minimize the total cost (λ), i.e.,

$$\underset{z_{i,p,t}, w^j_{i,p,t}, x_{i,p,t}, s^{p'}_{i,p,t}, h^{i'}_{i,p,t}}{\textbf{Min}} \quad \{\lambda\} \tag{21}$$

subject to (1), (3), (8)–(11), and (22)–(26)

$$z_{i,p,t} \in N^+ \tag{22}$$

$$s^{p'}_{i,p,t} \in N^+ \tag{23}$$

$$x_{i,p,t} \in N^+ \tag{24}$$

$$h^{i'}_{i,p,t} \in N^+ \tag{25}$$

$$w^j_{i,p,t} \in \{0, 1\}. \tag{26}$$

This work adopts the penalty function approach [27] to obtain an unconstrained problem. Each constraint is transformed into a nonnegative penalty added to λ. Therefore, if the total penalty is 0, each constraint is met strictly; otherwise, it is not. Let $\widetilde{\lambda}$ denote a new objective of the unconstrained problem, which is the fitness for each solution. Each inequality constraint is first transformed into its standard form $\overset{=}{h}_{\Upsilon_1}(\hbar) \geq 0$, and each equality one is transformed into its standard form $\overset{\neq}{h}_{\Upsilon_2}(\hbar)=0$. Then, we have

$$\underset{\hbar}{\textbf{Min}} \left\{\widetilde{\lambda} = \overset{\infty}{\mathcal{N}}\Omega + \lambda\right\} \tag{27}$$

$$\Omega = \sum_{\Upsilon_1=1}^{\mathbb{N}^{\neq}} \left(\max\{0, -\overset{=}{h}_{\Upsilon_1}(\hbar)\}\right)^2 + \sum_{\Upsilon_2=1}^{\mathbb{N}^{=}} \left|\overset{\neq}{h}_{\Upsilon_2}(\hbar)\right|^2$$

$$\overset{=}{h}_{\Upsilon_1}(\hbar) \geq 0$$

$$\overset{\neq}{h}_{\Upsilon_2}(\hbar) = 0$$

where $\hbar$ is a decision vector, $\overset{\infty}{\mathcal{N}}$ is a large constant, Ω is the total penalties of all constraints, $\mathbb{N}^{=}$ and $\mathbb{N}^{\neq}$ are numbers of equality and inequality constraints.

Here, $s^{p'}_{i,p,t}$, $w^j_{i,p,t}$, $h^{i'}_{i,p,t}$, $x_{i,p,t}$ and $z_{i,p,t}$ are discrete integer variables. Besides, f_i, $u_{i,t}$, $m_{i,t}$, λ_1, λ_2 and λ_3 are nonlinear with respect to $s^{p'}_{i,p,t}$, $w^j_{i,p,t}$, $h^{i'}_{i,p,t}$, $x_{i,p,t}$ and $z_{i,p,t}$. It is clear that (10) is nonlinear with respect to $z_{i,p,t}$, $x_{i,p,t}$, and $h^{i'}_{i,p,t}$. Constraint (11) is nonlinear in terms of $z_{i,p,t}$. In addition, (21) is nonlinear in terms of decision variables. Consequently, it is a typical nonlinear and constrained integer optimization problem, which is NP-hard [28], and no polynomial-time algorithms are available [29].

V. Genetic Simulated Annealing-Based Particle Swarm Optimization With Auto-Encoders

Currently, there are many mathematical optimization algorithms, e.g., stochastic gradient descent to solve constrained problems. However, they can only be applied when objective functions follow certain mathematical characteristics. For instance, first-order derivatives are often essential. Besides, their obtained solutions are unsatisfying for complicated optimization problems given limited time. To handle such challenges, typical meta-heuristic algorithms have many merits, e.g., strong robustness, fast convergence [30], [31]. Thus, they are widely applied by an increasing number of emerging methods to address the disadvantages and address different real-life complex problems. However, PSO has a fast convergence speed, yet it only yields local optima when solving complex problems [32]. Its search process oscillates if each particle's locally best and the globally best positions differ greatly. Besides, GA is widely used because it can yield more diverse individuals, thereby providing high-search efficiency and accuracy [33]. SA's solution quality is high because of its conditional acceptance rule, which conditionally chooses worse solutions with the hope of finally converging to the optimal ones. However, its convergence process is relatively slow because its temperature cooling rate must

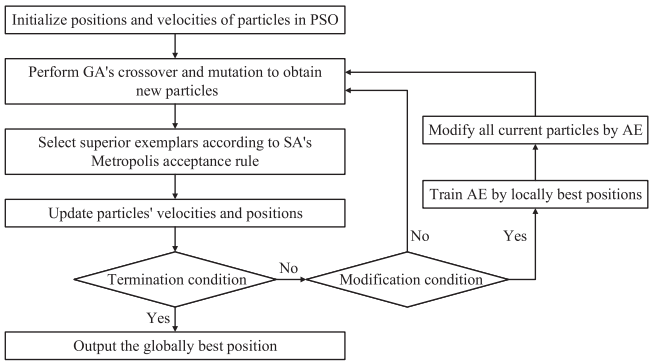


Fig. 2. Flowchart of GSPAE.

be small to guarantee high-search accuracy. Furthermore, these algorithms perform poorly in solving high-dimensional problems.

Consequently, we combine the genetic operations of GA, the metropolis acceptance rule of SA, AE, and PSO, and design a hybrid optimization algorithm called GSPAE. GSPAE has three major steps in each iteration: 1) updating superior exemplars for each particle; 2) updating the position of each particle according to its superior exemplar, and guiding the search process; and 3) adopting an AE to modify particles according to features of locally best positions of particles in the current population.

Specifically, the main advantages of GSPAE are given as follows. *First*, high-quality and diversified superior exemplars guide PSO's particles, thereby avoiding problems of premature convergence and trapping into local optima. *Second*, the Metropolis acceptance rule is adopted to conditionally select worse particles as superior exemplars, which guide the evolution of particles and improve the global search ability. *Third*, the evolution information of high-quality particles propagates genetic information back to GA that adopts crossover and mutation operations to produce improved particles, thereby enhancing the diversity of PSO's particles and achieving global exploitation. *Fourth*, AE captures features of high-quality particles and modifies others in the current population according to them for improving the overall quality of particles, thereby accelerating GSPAE's optimization process. *Fifth*, in our experiment, AE greatly improves GSPAE's ability to handle high-dimensional optimization problems. Finally, GSPAE yields a near-optimal solution. A detailed process of GSPAE is shown in Fig. 2. Main notations and their definitions of our proposed GSPAE are summarized in Table II.

$|\mathbb{X}|$ is the particle number. D is the position dimension of each particle. In a position, its first $|\mathcal{I}||\mathcal{P}|T$ elements store $w^{j}_{i,p,t}$. The next $|\mathcal{I}||\mathcal{P}|T$ elements store $z_{i,p,t}$. The next $|\mathcal{I}||\mathcal{P}|T(|\mathcal{P}|-1)$ elements store $s^{p'}_{i,p,t}$. The next $|\mathcal{I}||\mathcal{P}|T$ elements store $x_{i,p,t}$. The last $|\mathcal{I}||\mathcal{P}|T(|\mathcal{I}|-1)$ elements store $h^{i'}_{i,p,t}$. Thus, $D{=}|\mathcal{I}||\mathcal{P}|T(|\mathcal{P}|+|\mathcal{I}|+1)$.

ℓ_ϱ is a locally best position of particle ϱ ($1{\le}\varrho{\le}|\mathbb{X}|$), and g is a globally best position in the whole population. A superior exemplar (e_ϱ) is designed for particle ϱ. $\ell_{\varrho,d}$ is element d ($1{\le}d{\le}D$) of ℓ_ϱ, and g_d is element d of g. $e_{\varrho,d}$ is element d of e_ϱ obtained as

$$e_{\varrho,d}=\frac{c_1{\cdot}r_1{\cdot}\ell_{\varrho,d}+c_2{\cdot}r_2{\cdot}g_d}{c_1{\cdot}r_1+c_2{\cdot}r_2} \tag{28}$$

where c_1 (c_2) is cognitive (social) acceleration constant, and r_1, $r_2\in(0,1)$.

In each iteration of GSPAE, there are five major components to obtain superior exemplars, position update for each particle, and the modification of particles by the AE. The details are given as follows.

TABLE II
MAIN NOTATIONS AND DEFINITIONS OF GSPAE

Notation	Definition
$\lvert\mathbb{X}\rvert$	Particle number.
$\mathbb{X}$	Particle swarm.
D	Position dimension of each particle.
ℓ_ϱ	Locally best position of particle ϱ.
g	Globally best position in the whole population.
e_ϱ	Superior exemplar designed for particle ϱ.
c_1	Cognitive acceleration constant.
c_2	Social acceleration constant.
r_1	Random number in (0,1).
r_2	Random number in (0,1).
ξ	Randomly selected particle.
o_ϱ	Yielded offspring.
$\widetilde{\lambda}(\ell_\varrho)$	Fitness value of particle ϱ.
$\hat{b}_d$	Upper limit of each entry d.
$\check{b}_d$	Lower limit of each entry d.
ζ	Specified mutation probability.
T_ς	Current temperature in each iteration ς.
$\varkappa$	Random number in (0,1).
$\hat{\varsigma}$	Number of iterations.
ω_ς	Inertia weight in each iteration ς.
$\hat{\omega}$	Maximum value of ω_ς.
$\check{\omega}$	Minimum value of ω_ς.
ω_ς	Inertia weight in each iteration ς.
$h_\varrho^{\varsigma+1}$	Position of particle ϱ in iteration $\varsigma{+}1$.
$v_\varrho^{\varsigma+1}$	Velocity of particle ϱ in iteration $\varsigma{+}1$.
c	Parameter of acceleration.
$F(\cdot)$	Encoding process.
$G(\cdot)$	Decoding process.
$\widetilde{\ell}_\varrho$	Output of a perfect AE.
$\widetilde{h}$	Output of AE.
$\widetilde{\varsigma}$	Number of iterations for retraining AE.
T_0	Starting temperature.
ϖ	Cooling rate of temperature.
ϱ	Epoch number in training AE.
$\mathcal{T}$	Training set of particles.
$\lvert\mathcal{T}\rvert$	Number of samples in the training set $\mathcal{T}$.

A. Crossover Operation

We conduct the crossover operation on each dimension d of particle ϱ. A particle ξ ($\xi\in\{1,2,\ldots,|\mathbb{X}|\}$) is first selected randomly. (29) is used to yield the offspring $o_\varrho=(o_{\varrho,1}, o_{\varrho,2}, \cdots, o_{\varrho,D})$, i.e.,

$$o_{\varrho,d}=\begin{cases} r_d{\cdot}\ell_{\varrho,d}+(1-r_d){\cdot}g_d, & \widetilde{\lambda}(\ell_\varrho)<\widetilde{\lambda}(\ell_\xi) \\ \ell_{\xi,d}, & \text{otherwise} \end{cases} \tag{29}$$

where $r_d\in[0,1]$, and $\widetilde{\lambda}(\ell_\varrho)$ is the fitness value of particle ϱ, which is calculated by (27).

Different from the random selection of two particles in GA's crossover, the proposed crossover operation adopts historical

search results (the globally best position and locally best one) of each particle.

B. Mutation Operation

To yield high-quality superior exemplars, we use (30) to conduct the mutation operation on $o_{\varrho,d}$. $o_{\varrho,d}$ is changed as

$$o_{\varrho,d} = \mathbf{rand}\left(\check{b}_d, \hat{b}_d\right), \text{ if } r_d < \zeta \tag{30}$$

where $\hat{b}_d$ and $\check{b}_d$ are upper and lower limits of each entry d, and ζ is the specified mutation probability.

C. SA-Based Selection Component

After crossover and mutation components, this work compares o_ϱ and e_ϱ, and selects one of them according to SA's Metropolis acceptance rule as a new superior exemplar for particle ϱ in this iteration. In detail, if $\widetilde{\lambda}(o_\varrho) < \widetilde{\lambda}(e_\varrho)$, e_ϱ is updated to o_ϱ. If $\widetilde{\lambda}(o_\varrho) \geq \widetilde{\lambda}(e_\varrho)$ and $\exp^{[-(\widetilde{\lambda}(o_\varrho)-\widetilde{\lambda}(e_\varrho))/T_\varsigma]} > \varkappa$, e_ϱ is updated by o_ϱ. Here, T_ς is the current temperature in each iteration ς, and $\varkappa$ is a random number in (0, 1) in different iterations. If $\widetilde{\lambda}(o_\varrho) \geq \widetilde{\lambda}(e_\varrho)$ and $\exp^{[-(\widetilde{\lambda}(o_\varrho)-\widetilde{\lambda}(e_\varrho))/T_\varsigma]} \leq \varkappa$, e_ϱ remains unchanged, i.e.,

$$e_\varrho = \begin{cases} o_\varrho, & \widetilde{\lambda}(o_\varrho) < \widetilde{\lambda}(e_\varrho) \\ o_\varrho, & \widetilde{\lambda}(o_\varrho) \geq \widetilde{\lambda}(e_\varrho) \text{ and } \exp^{\left(-\frac{\widetilde{\lambda}(o_\varrho)-\widetilde{\lambda}(e_\varrho)}{T_\varsigma}\right)} > \varkappa \\ e_\varrho, & \widetilde{\lambda}(o_\varrho) \geq \widetilde{\lambda}(e_\varrho) \text{ and } \exp^{\left(-\frac{\widetilde{\lambda}(o_\varrho)-\widetilde{\lambda}(e_\varrho)}{T_\varsigma}\right)} \leq \varkappa. \end{cases} \tag{31}$$

D. Position Update Component

$\hat{\varsigma}$ is the iteration number. ω_ς is inertia weight in each iteration ς $(1 \leq \varsigma \leq \hat{\varsigma})$. Then

$$\omega_\varsigma = \hat{\omega} - \frac{\varsigma\left(\hat{\omega} - \check{\omega}\right)}{\hat{\varsigma}} \tag{32}$$

where $\hat{\omega}$ ($\check{\omega}$) is a maximum (minimum) value of ω_ς.

$\hbar_\varrho^{\varsigma+1}$ and $v_\varrho^{\varsigma+1}$ denote position and velocity of particle ϱ in iteration $\varsigma+1$. Then, $v_{\varrho,d}^{\varsigma+1}$ and $\hbar_{\varrho,d}^{\varsigma+1}$ are updated and guided by e_ϱ, i.e.,

$$v_{\varrho,d}^{\varsigma+1} = \omega_\varsigma \cdot v_{\varrho,d}^{\varsigma} + c \cdot r_d \cdot \left(e_{\varrho,d}^{\varsigma} - \hbar_{\varrho,d}^{\varsigma}\right) \tag{33}$$

$$\hbar_{\varrho,d}^{\varsigma+1} = \hbar_{\varrho,d}^{\varsigma} + v_{\varrho,d}^{\varsigma} \tag{34}$$

where c is a parameter of acceleration.

E. Particle Modification Component

In some specific iterations, after positions are updated, AE is introduced to modify current particles to accelerate the optimization process. Fig. 3 shows the structure of AE, which is a symmetrical neural network, including an encoder and a decoder [34]. During the training phase, the encoder transforms the input data into its low-dimensional one. Then, the neural network learns the most informative and key features. The encoding process is denoted by $F(\cdot)$. The decoding process is denoted by $G(\cdot)$, which reconstructs the high-dimensional output data. In this work, the locally best positions in the population are used as training samples for AE. Let $\widetilde{\ell}_\varrho$ denote the output of a perfect AE. In the ideal case, the output of the perfect AE accurately restores the original input, i.e.,

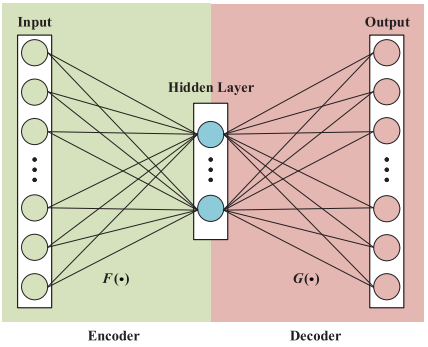


Fig. 3. Structure of AE.

$$\widetilde{\ell}_\varrho = G\left(F(\ell_\varrho)\right) \approx \ell_\varrho. \tag{35}$$

$F(\cdot)$ and $G(\cdot)$ of AE mean the feature extraction process and the feature decoding one, respectively. Thus, after a particle $(\vec{\hbar})$ enters AE as the input data, it is modified by the trained $F(\cdot)$ and $G(\cdot)$. Let $\widetilde{\vec{\hbar}}$ denote the output of AE, which means a reconstructed position with features of training samples (locally best positions), i.e.,

$$\widetilde{\vec{\hbar}} = G\left(F(\vec{\hbar})\right). \tag{36}$$

The training process of our AE module is given as follows. The backpropagation algorithm is used to train the AE module with the optimizer of Adam. In addition, in GSPAE, the AE module is trained again for every $\widetilde{\varsigma}$ iterations, and all particles in the current data set are removed. Then, in the next $\widetilde{\varsigma}$ iterations, new and high-quality particles are collected again as the training data set of our new AE. The details of the integration of AE into GSPAE are as follows. First, after every $\widetilde{\varsigma}$ iterations, the locally best positions of the current population are adopted as training samples to train a new AE, which allows GSPAE to capture features of high-quality particles. Then, particles are reconstructed by the AE after their positions are updated in this iteration. Finally, the features of high-quality particles are added to each particle, thereby improving the quality of the entire population.

The details of GSPAE are shown in Algorithm 1. Line 1 randomly sets velocities and positions of particles. Line 2 obtains a fitness value $(\widetilde{\lambda})$ of a particle. g and ℓ_ϱ are updated in line 3. Line 4 initializes key parameters of SA, GA and PSO, e.g., the starting temperature (T_0), the cooling rate (ϖ), c, c_1, c_2, $\check{\omega}$, $\hat{\omega}$, $\widetilde{\varsigma}$, $\hat{\varsigma}$ and $|\mathbb{X}|$. Line 5 set superior exemplars with (28). Line 7 shows the *while* loop stops if $\varsigma > \hat{\varsigma}$. Line 8 performs (29) on each element d of particle ϱ to yield o_ϱ. Line 9 performs (30) on o_ϱ following the mutation probability of ζ. Line 10 reduces temperature T_ς by ϖ. Line 11 performs the selection operation to select o_ϱ. Lines 12 and 13 update $v_\varrho^{\varsigma+1}$ and $\hbar_\varrho^{\varsigma+1}$ with (33) and (34), respectively. **If** $\varsigma \% \widetilde{\varsigma}$ is 0, line 15 trains the AE with the obtained locally best particles. Line 16 rebuilds each particle $(\vec{\hbar})$ with the AE. Line 18 changes ℓ_ϱ and g. Line 19 decreases ω_ς linearly from $\hat{\omega}$ to $\check{\omega}$. Line 22 returns g.

Algorithm 1 GSPAE

1: Randomly initialize velocities and positions of particles
2: Obtain the fitness value of each particle
3: Update g and ℓ_ϱ
4: Initialize $T_0, \zeta, \varpi, c_1, c_2, c, \check{\omega}, \hat{\omega}, \widetilde{\varsigma}, \hat{\varsigma}, |\mathbb{X}|$ and parameters of AE
5: Initialize superior exemplars with (28)
6: $\varsigma \leftarrow 1$
7: **while** $\varsigma \leq \hat{\varsigma}$ **do**
8: Perform (29) to obtain o_ϱ
9: Perform (30) on o_ϱ
10: $T_\varsigma \leftarrow T_{\varsigma-1}\varpi$
11: Perform the selection operation with (31) to update e_ϱ
12: Update $v_\varrho^{\varsigma+1}$ with (33)
13: Update $\hbar_\varrho^{\varsigma+1}$ with (34)
14: **if** $\varsigma \% \widetilde{\varsigma} == 0$ **then**
15: Train an AE by ℓ_ϱ
16: Reconstruct all particles ($\hbar_\varrho^{\varsigma+1}$) with the trained AE
17: **end if**
18: Update g and ℓ_ϱ
19: Update ω_ς with (32)
20: $\varsigma \leftarrow \varsigma + 1$
21: **end while**
22: **return** g

The complexity of Algorithm 1 is discussed here. Its overhead is largely brought by the *while* loop. As shown in lines 8–20, the complexity of crossover, mutation, SA-based selection, and position update in an iteration is $\mathcal{O}(|\mathbb{X}|D)$. The training complexity of AE is $\mathcal{O}(D|\mathcal{T}|\varrho)$ where ϱ is the epoch number in training AE, and $|\mathcal{T}|$ is the number of samples in the training set $\mathcal{T}$. In addition, $D{=}|\mathcal{I}||\mathcal{P}|T(|\mathcal{P}|+|\mathcal{I}|+1)$, and the complexity in an iteration is $\mathcal{O}((|\mathbb{X}|+|\mathcal{T}|\varrho)|\mathcal{I}||\mathcal{P}|T(|\mathcal{P}|+|\mathcal{I}|+1))$. In addition, the number of types of products is much bigger than the plant number, i.e., $|\mathcal{I}|{>}|\mathcal{P}|$. Above all, the complexity of Algorithm 1 is $\mathcal{O}((|\mathbb{X}|+|\mathcal{T}|\varrho)|\mathcal{I}|^2|\mathcal{P}|T)$.

It is worth noting that GSPAE tackles the problem of dimensional curse in the CMS. However, AE inevitably increases the computational overhead, thereby increasing solution time. Therefore, when GSPAE is applied to real-life CMS systems, high-performance servers are needed to show their superior performance. Unlike long-time training and instant prediction modes of neural network approaches, GSPAE is an intelligent optimization algorithm that requires iterative computation for decision-making and problem-solving. Additionally, during the iterative process, GSPAE also involves the training of AEs, which requires a significant amount of time. For instance, yielding a 30-day production plan in the CMS with GSPAE may take several tens of minutes. In manufacturing industries, it is acceptable to use a few hours to yield a monthly production plan. However, GSPAE determines intelligent production planning strategies within 1 h based on current order information and enterprise-specific conditions, thereby making it feasible and adaptable to the real-time needs of the manufacturing sectors.

In addition, it is worth noting that GSPAE lacks sufficient capability to handle disruptions in the production processes, such as newly added or canceled orders, machine failures, and uncertainties, caused by weather factors. When encountering such situations, the only option is to replan, i.e., GSPAE regenerates a decision to resolve the issues. GSPAE has excellent solving speeds in manufacturing industries and takes only a few tens of minutes to generate a machine-level plan, which proves that GSPAE effectively overcomes the aforementioned problems of disruptions.

VI. Performance Evaluation

This section evaluates the performance of GSPAE. We consider different manufacturing tasks. Here, $|\mathcal{I}| \in [1, 15]$, $T \in [1, 30]$, $|\mathcal{P}| \in [2, 6]$ and each plant has different numbers of heterogeneous machines. GSPAE is coded with Python in Pycharm 2020 and it runs in a server with a 32-GB memory and an Intel Core i7-13700FQ CPU with 4.90 GHz. This work adopts Pytorch as the framework of our AE module, which is a symmetric and fully connected neural network. The AE module includes five layers, i.e., an input layer, three hidden ones, and an output one. In addition, the number of neurons in each symmetric layer is the same. The neuron numbers in the second and fourth layers are 3/4 of that in the input layer. The neuron number in the third layer is 1/2 of that in the input layer. Besides, this work selects ReLu as the activation function. Furthermore, this work adopts a GPU of NVIDIA RTX 4090 24GB to train our AE module in our proposed GSPAE. The training data set includes high-quality particles (the globally best and the locally best particles) collected in the evolutionary process.

A. Parameter Setting

Table III shows 14 real-world MP cases used to evaluate the proposed GSPAE. Here, the planning horizon means the planning period. Besides, cases 9-12 need to consider the pairing relationships among products. The dimension means the number of decision variables of MP tasks. Figs. 4 and 5 show the effect of ζ and c on GSPAE for cases 5 and 7, respectively. It is observed that GSPAE's cost is smaller for each case if $\zeta \in [0.02, 0.06]$. GSPAE's cost is the lowest when $c = 1.5$. Furthermore, this work performs multiple experiments to show the impact of other key parameters of GSPAE for case 9, as shown in Table IV. It is worth noting that results for other cases are similar and we take case 9 for example. When the population size $|\mathbb{X}|$ is 50, a close-to-optimal solution is already yielded. Further increase of $|\mathbb{X}|$ does not yield significant cost reduction, but it prolongs the computation time. The cognitive acceleration parameter c_1 and the social acceleration one c_2 represent the influence of locally best positions and globally best ones, respectively. We vary c_1 by setting c_2 to 0.5 to find its best setting. Table IV demonstrates that the lowest cost is yielded when $c_1 = 0.5$. $\widetilde{t}$ determines the update frequency of AE. If $\widetilde{t}$ is too small, the AE is trained frequently, thus significantly increasing the training time and potentially leading to insufficient samples for capturing features of high-quality particles. On the other

TABLE III
14 REAL-WORLD MP CASES

Case	Product type number	Plant number	Product line number	Planning horizon	Dimension
Case 1	1	2	2	01/12-03/12 (2 days)	28
Case 2	1	3	4	01/12-03/12 (2 days)	60
Case 3	2	2	2	01/12-03/12 (2 days)	64
Case 4	2	3	4	01/12-03/12 (2 days)	132
Case 5	3	2	2	01/12-03/12 (2 days)	108
Case 6	3	3	4	01/12-03/12 (2 days)	216
Case 7	4	2	2	01/12-03/12 (2 days)	160
Case 8	4	3	4	01/12-03/12 (2 days)	312
Case 9	9	3	6	02/01-16/01 (7 days)	3780
Case 10	9	5	6	16/01-30/01 (14 days)	13860
Case 11	15	3	6	01/05-15/05 (14 days)	16380
Case 12	10	3	6	01/03-30/03 (30 days)	18900
Case 13	10	6	6	01/09-15/09 (14 days)	20160
Case 14	15	5	10	01/12-15/12 (14 days)	33600

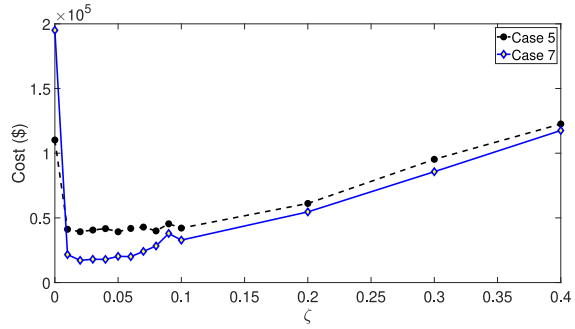


Fig. 4. Effect of ζ on GSPAE.

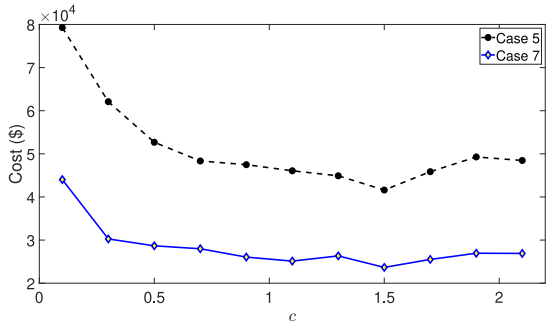


Fig. 5. Effect of acceleration parameter (c) on GSPAE.

hand, if $\widetilde{t}$ is too large, the AE's accelerating effect on evolution diminishes. Therefore, $\widetilde{t} = 100$, which indicates that AE transforms the population once every 100 iterations. As shown in Table IV, it is obvious that GSPAE already converges when $\hat{t} = 1000$. δ determines the cooling rate of temperature. When the temperature is decreased too rapidly, GSPAE is prone to trapping into local optima. On the other hand, a small temperature cooling rate leads to a slower evolution speed. The above results demonstrate that the best solution is yielded when $\delta = 0.95$. Finally, the main parameters of GSPAE are presented in Table V. For AE, the neuron number in the hidden layer is half of that in the input layer, the epoch number is 500, and the learning rate is 0.001.

TABLE IV
IMPACT OF MAIN PARAMETERS OF GSPAE

Parameter	Value	Total cost ($)	Computation time (mins.)
$\lvert\mathbb{X}\rvert$	30	15,742	2.6439
	40	15,277	3.4979
	50	**15,239**	**4.3715**
	60	15,235	5.6740
c_1	0.7	19,624	4.4000
	0.6	15,595	4.3716
	0.5	**13,486**	**4.3741**
	0.4	15,005	4.2656
	0.3	15,041	4.3531
$\widetilde{t}$	50	12,498	4.9135
	100	**10,064**	**4.5823**
	150	14,869	4.6050
	200	15,305	4.3766
$\hat{t}$	800	14,765	3.4129
	900	14,712	3.8348
	1000	**14,568**	**4.2599**
	1100	14,568	4.6847
δ	0.99	26,605	5.1502
	0.95	**15,154**	**5.1212**
	0.9	15,324	5.6185
	0.8	19,425	4.9226

B. Comparison of Different Algorithms for 14 Cases

We prove GSPAE's performance by comparing it with its state-of-the-art benchmark algorithms, including surrogate-assisted AE-embedded evolutionary optimization (SAEO) [36], self-adaptive bat algorithm with genetic operations (SBAGOs) [37], SAPSO [38], PSO, and GA.

1) *SAEO:* It integrates AEs and surrogate models into evolutionary algorithms, enabling it to effectively tackle high-dimensional expensive problems.

2) *SBAGO:* It incorporates the mutation operation from GA into the adaptive bat optimization algorithm, thus significantly enhancing its diversity and search efficiency.

TABLE V
SETTING OF MAIN PARAMETERS OF GSPAE

$\lvert\mathbb{X}\rvert$	$c_1(c_2)$	c	ζ	η_1	δ	$\hat{t}$	$\tilde{t}$	$\hat{\omega}$	$\check{\omega}$
50	0.5	1.5	0.05	10^8	0.95	1000	100	0.95	0.4

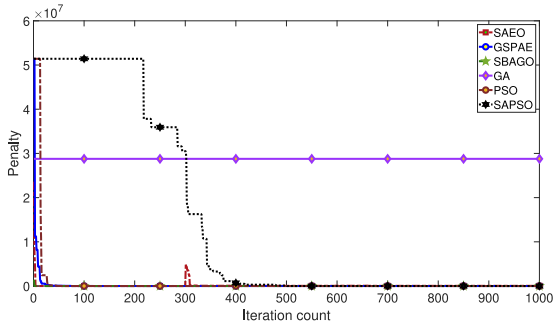


Fig. 6. Penalty of six algorithms for case 12.

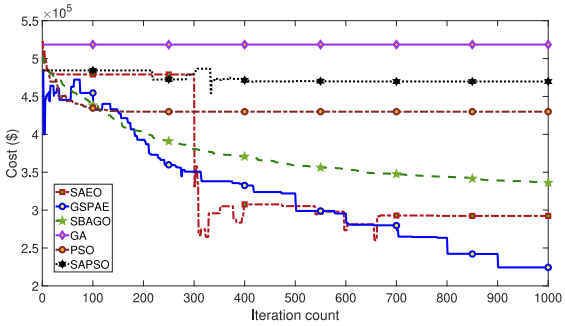


Fig. 7. Cost of six algorithms for case 12.

3) *SAPSO:* It combines the merits of SA's Metropolis acceptance rule and PSO, and it owns fast convergence and global search ability.
4) *PSO:* Its convergence speed is very fast due to its simple implementation. However, its global search accuracy is unsatisfying in many cases.
5) *GA:* It has a strong global search ability due to its crossover, mutation, and selection operations. However, its convergence speed is often slow.

Figs. 6 and 7 show evolutionary processes of penalty and cost of six algorithms for case 12. A solution with a penalty of 0 means that it satisfies all the constraints, and it is a valid solution. Fig. 6 indicates that the penalties of GSPAE, PSO, SAEO, and SBAGO drop rapidly, while that of GA fails to reduce because it is unable to optimize 18 900 decision variables for case 12. This shows that GSPAE quickly finds valid solutions for case 12. In addition, Fig. 7 shows that the cost of GSPAE and SBAGO constantly decreases during 1000 iterations. In addition, SAEO has a sharp decrease in cost after entering its second stage, followed by fluctuations within a certain range, which is consistent with the situation described in [36]. For GSPAE, in the first 400 iterations, it has strong global search capability and escapes from local optima. After 400 iterations, AE enables GSPAE to obtain better solutions through high-quality reconstruction every 100 iterations, thereby greatly improving the accuracy of GSPAE.

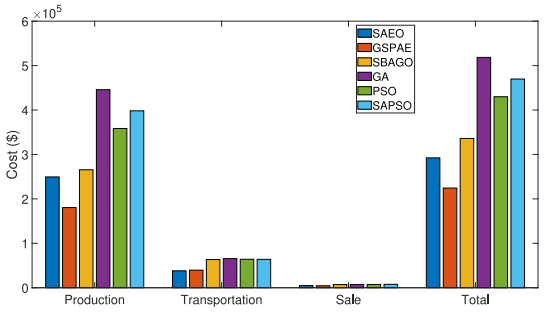


Fig. 8. Distribution of cost of six algorithms for case 12.

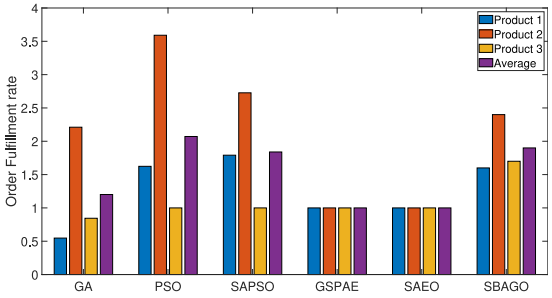


Fig. 9. Order fulfillment rates of six algorithms for case 6.

Finally, GSPAE yields a valid solution with the lowest cost. Figs. 6 and 7 illustrate accuracy and global search capabilities of GSPAE.

The cost of the system includes the manufacturing cost, transportation one, and sales one, as shown in (15). Fig. 8 shows the cost distribution of final solutions for case 12 with all six algorithms. GSPAE yields the lowest cost among the six algorithms, and it reduces the cost by more than 20% compared with other algorithms. The order fulfillment rate is an important indicator in intelligent manufacturing. A better solution means that the order fulfillment rate is closer to 1. When it is 1, the solution is called the optimal one, i.e., all orders have been finished, and there are no redundant products. Fig. 9 shows order fulfillment rates of six algorithms for case 6. It is clear that the order fulfillment rates of the three products for both GSPAE and SAEO are almost close to 1. In addition, the order fulfillment rates of SBAGP, GA, PSO, and SAPSO are unevenly distributed, and they are larger or lower than 1. Thus, Fig. 9 demonstrates that GSPAE obtains higher quality and valid solutions than its peers.

We adopt the above-mentioned six algorithms to solve 14 real-world MP cases, and each algorithm has 1000 iterations at most. Tables VI and VII show the cost and penalty, and average order fulfillment rates of 14 MP cases optimized by six algorithms, respectively. Obviously, GSPAE always obtains valid solutions with a penalty of 0. Its cost is always the lowest, and its average order fulfillment rate is the closest to 1 among all algorithms that yield valid solutions.

In cases 11, 13, and 14, although the cost of final solutions obtained by SAEO or SAPSO is less than that of GSPAE, their penalty values are very large, which indicates that their solutions are invalid. It is worth noting that in cases where all six algorithms obtain valid solutions, e.g., cases 1, 2, 9,

TABLE VI
COST AND PENALTY OF 14 MP CASES OPTIMIZED BY SIX ALGORITHMS

Case	Cost or penalty of six algorithms on each case											
	GSPAE		SAPSO		PSO		GA		SAEO		SBAGO	
	Cost	Penalty	Cost	Penalty	Cost	Penalty	Cost	Penalty	Cost	Penalty	Cost	Penalty
1	**5766**	0	7770	0	56568	0	14049	0	6011	0	6899	0
2	**6739**	0	24631	0	38883	0	48568	0	7200	0	7684	0
3	**14710**	0	118500	50000	64861	10000	128360	199900	16773	0	16820	0
4	**15581**	0	210720	50	155975	0	210120	0	16390	0	18239	0
5	**42401**	0	229628	345.9	302286	100640	301692	9935.9	45236	0	76352	0
6	**30545**	0	502020	406214	319845	7644	460600	6728	30578	0	66790	0
7	**17726**	0	209746	3.2×10^{7}	202713	3.4×10^{7}	264003	1.0×10^{8}	20348	0	39283	0
8	**39597**	0	329540	2.9×10^{8}	313724	5.0×10^{7}	344954	4.2×10^{8}	42249	0	62385	0
9	**9402**	0	44769	0	46248	0	80971	0	14599.3	0	28277.6	0
10	**121700**	0	284520	0	291276	0	397740	0	156324	0	225280.8	0
11	294390	0	**241713**	5.7×10^{8}	294350	4.6×10^{8}	425640	3.2×10^{9}	290697.2	34146.3	329536.9	0
12	**224380**	0	469700	0	429890	2.9×10^{5}	518440	2.8×10^{7}	257209.6	14634.2	419728.5	0
13	270330	0	544440	0	520570	0	583890	1.6×10^{7}	**133725.3**	2.0×10^{6}	369041.5	19692.8
14	684500	0	**528412**	2.0×10^{9}	556848	1.9×10^{9}	835510	6.0×10^{9}	75531.6	5.1×10^{6}	761152.4	2.1×10^{7}

TABLE VII
AVERAGE ORDER FULFILLMENT RATES OF 14 MP CASES OPTIMIZED BY SIX ALGORITHMS

Case	**Average order fulfillment rates**					
	GSPAE	SAPSO	PSO	GA	SAEO	SBAGO
Case 1	**1.001**	3.485	3.485	4.518	1.043	1.197
Case 2	**1.002**	3.787	5.451	2.861	1.070	1.142
Case 3	**1.000**	4.767	7.177	3.949	1.140	1.143
Case 4	**1.001**	4.030	4.074	5.295	1.053	1.171
Case 5	**1.001**	1.839	2.072	1.201	1.071	1.801
Case 6	**1.006**	7.793	4.602	7.194	1.006	2.198
Case 7	**1.001**	14.788	3.361	6.211	1.150	2.218
Case 8	**1.460**	7.344	5.992	8.305	1.558	2.301
Case 9	**1.128**	1.390	1.371	1.380	1.148	1.268
Case 10	**1.232**	1.588	1.495	1.646	1.321	1.534
Case 11	2.124	2.696	2.773	2.822	**1.258**	2.324
Case 12	**1.277**	1.611	1.603	1.593	1.280	2.370
Case 13	1.557	2.369	2.190	2.524	**1.277**	2.633
Case 14	**3.611**	4.379	4.359	4.343	0.088	4.116

and 10, GSPAE reduces at least 5% of the cost of the system given the same MP problem, e.g., case 1. Additionally, as the complexities of the manufacturing problems increase, the cost differences between GSPAE and other algorithms become more significant, with the maximum cost savings reaching 35% in the most significant case, e.g., case 9. Especially, when the problem is high-dimensional, e.g., cases 9–14, GSPAE also obtains superior solutions than other algorithms. Thus, GSPAE has higher accuracy and a stronger ability to solve MP problems in the proposed intelligent manufacturing system.

TABLE VIII
TIME FOR SIX ALGORITHMS TO FIND BETTER SOLUTIONS THAN PSO

Cases	Time (min.)					
	GSPAE	SAPSO	GA	PSO	SAEO	SBAGO
Case 11	**0.54**	9.79+	15.60+	9.77	0.66	1.60
Case 12	**10.32**	16.96+	24.11+	18.24	35.77	10.42
Case 13	**3.23**	18.56+	32.19+	18.05	6.01	10.82
Case 14	**0.66**	20.22+	35.22+	20.30	1.21	2.22

C. Computational Time

All the experiments are conducted on the same server, and they are coded in Python. This work adopts PSO as a benchmark method, and the fitness value of its final solution was set as a reference calculated by (27). Table VIII shows the time required for six algorithms to find better solutions than PSO. The column of PSO indicates the convergence time of PSO. In some cases, GA and SAPSO fail to find better solutions during 1,000 iterations, and we mark their final time at iteration 1,000 as their lower limits. Table VIII indicates that GSPAE always obtains solutions better than PSO with the least time. For SAEO, since it often finds better solutions than PSO only after entering the second stage, it requires a significant amount of time for training AEs and surrogate models. In summary, GSPAE not only exhibits high accuracy and search ability but also demonstrates less convergence time.

VII. CONCLUSION

Emerging industrial Internet realizes low-cost and high-efficiency scheduling for industrial production processes. Current manufacturing and scheduling planning in industrial Internet has many problems because it only has limited manufacturing capacities and resources of each plant and

numerous optimization variables. Existing methods cannot well optimize collaboration and transportation among different plants. Thus, this leads to resource waste, message blockage, inefficiency, redundant production, *etc*. To achieve it, we propose machine-level scheduling for multiple different plants in a system of the industrial Internet. A comprehensive machine-level architecture is proposed for enterprises with multiple plants. Then, a limited nonlinear optimization problem of the integer is formulated to reduce the total cost of transportation, production, and sales. We also consider several real-life constraints, e.g., limits of storage space, replacement times, pairing production, substitution, and order fulfillment rates. To solve it, a hybrid meta-heuristic optimization algorithm named genetic simulated annealing-based particle swarm optimization with auto-encoders (GSPAE) is designed. GSPAE integrates genetic operations and a conditional acceptance rule into the particle swarm optimizer. Extensive experiments with real-life data show that its cost is lower than other typical algorithms.

REFERENCES

[1] Y. Liu, C. Chi, Y. Zhang, and T. Tang, "Identification and resolution for Industrial Internet: Architecture and key technology," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 16780–16794, Sep. 2022.

[2] M. Aazam, K. Harras, and S. Zeadally, "Fog computing for 5G tactile Industrial Internet of Things: QoE-aware resource allocation model," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3085–3092, May 2019.

[3] H. Tang, D. Li, J. Wan, M. Imran, and M. Shoaib, "A reconfigurable method for intelligent manufacturing based on industrial cloud and edge intelligence," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4248–4259, May 2020.

[4] J. Ma, H. Zhou, C. Liu, M. E. Z. Jiang, and Q. Wang, "Study on edge-cloud collaborative production scheduling based on enterprises with multi-factory," *IEEE Access*, vol. 8, pp. 30069–30080, 2020.

[5] Z. Wang et al., "Multiobjective optimization-aided decision-making system for large-scale manufacturing planning," *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 8326–8339, Aug. 2022.

[6] B. Wally et al., "Leveraging iterative plan refinement for reactive smart manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 1, pp. 230–243, Jan. 2021.

[7] S. M. Zahraee, N. Shiwakoti, and P. Stasinopoulos, "Simulation scenario analysis of operational day to day storage system of biomass supply chain for a power plant case study based on logistic cost and transportation emissions," in *Proc. Int. Conf. Adv. Electr., Comput., Commun. Sustain. Technol. (ICAECT)*, Bhilai, India, Apr. 2021, pp. 1–8.

[8] A. Alfaro, F. Torres, and C. Ibañez, "Sales and operation planning model to improve inventory management in peruvian SMEs," in *Proc. 8th Int. Conf. Ind. Technol. Manage. (ICITM)*, Cambridge, U.K., May 2019, pp. 65–68.

[9] X. Wu, Q. Yuan, and L. Wang, "Multiobjective differential evolution algorithm for solving robotic cell scheduling problem with batch-processing machines," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 757–775, Apr. 2021.

[10] F. Marinho de Brito, G. da Cruz, E. Frazzon, J. Basto, and S. Alcalá, "Design approach for additive manufacturing in spare part supply chains," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 757–765, Feb. 2021.

[11] Y. Hao, L. Hu, and M. Chen, "Joint sensing adaptation and model placement in 6G fabric computing," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 7, pp. 2013–2024, Jul. 2023.

[12] H. Rahman, M. Janardhanan, and I. Nielsen, "Real-time order acceptance and scheduling problems in a flow shop environment using hybrid GA-PSO algorithm," *IEEE Access*, vol. 7, pp. 112742–112755, 2019.

[13] B. Han and J. Yang, "Research on adaptive job shop scheduling problems based on dueling double DQN," *IEEE Access*, vol. 8, pp. 186474–186495, 2020.

[14] Y. Gao, J. Wang, S. Gao, and Y. Cheng, "An integrated robust design and robust control strategy using the genetic algorithm," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8378–8386, Dec. 2021.

[15] Z. Jiao, J. Zhang, P. Yao, L. Wan, and L. Ni, "Service deployment of C4ISR based on genetic simulated annealing algorithm," *IEEE Access*, vol. 8, pp. 65498–65512, 2020.

[16] D. Wu, N. Jiang, W. Du, K. Tang, and X. Cao, "Particle swarm optimization with moving particles on scale-free networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 497–506, Jan.–Mar. 2020.

[17] M. Tajmirriahi, R. Kafieh, Z. Amini, and H. Rabbani, "A lightweight mimic convolutional auto-encoder for denoising retinal optical coherence tomography images," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–8, 2021.

[18] J. Leng et al., "ManuChain: Combining permissioned blockchain with a holistic optimization model as bi-level intelligence for smart manufacturing," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 1, pp. 182–192, Jan. 2020.

[19] W. Chen and Z. Wang, "Integrated capacity planning and production control of an assembly manufacturing system," *IEEE Trans. Eng. Manag.*, vol. 68, no. 3, pp. 868–880, Jun. 2021.

[20] S. Rubaiee, S. Cinar, and M. Yildirim, "An energy-aware multiobjective optimization framework to minimize total tardiness and energy cost on a single-machine nonpreemptive scheduling," *IEEE Trans. Eng. Manag.*, vol. 66, no. 4, pp. 699–714, Nov. 2019.

[21] X. Li, K. Xing, M. Zhou, X. Wang, and Y. Wu, "Modified dynamic programming algorithm for optimization of total energy consumption in flexible manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 691–705, Apr. 2019.

[22] Y. Pan, K. Gao, Z. Li, and N. Wu, "Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems," *IEEE Trans. Autom. Sci. Eng.*, vol. 20, no. 1, pp. 361–371, Jan. 2023.

[23] J. Wang and L. Wang, "A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 5, no. 6, pp. 947–961, Dec. 2021.

[24] S. Liu, Z. Chen, Z. Zhan, S. Jeon, S. Kwong, and J. Zhang, "Many-objective job-shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach," *IEEE Trans. Cybern.*, vol. 53, no. 3, pp. 1460–1474, Mar. 2023.

[25] C. Lin, D. Deng, Y. Chih, and H. Chiu, "Smart manufacturing scheduling with edge computing using multiclass deep Q network," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4276–4284, Jul. 2019.

[26] H. Yuan, Q. Hu, and J. Bi, "Cost-minimized and multi-plant scheduling in distributed industrial systems," in *Proc. Int. Conf. Syst., Man, Cybern.*, Prague, Czech Republic, 2022, pp. 1–6.

[27] P. Srivastava and J. Cortes, "Nesterov acceleration for equality-constrained convex optimization via continuously differentiable penalty functions," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 415–420, Apr. 2021.

[28] T. Vu, D. Nguyen, D. Hoang, E. Dutkiewicz, and T. Nguyen, "Optimal energy efficiency with delay constraints for multi-layer cooperative fog computing networks," *IEEE Trans. Commun.*, vol. 69, no. 6, pp. 3911–3929, Jun. 2021.

[29] Y. Gao, S. Zhang, and J. Zhou, "A hybrid algorithm for multi-objective scientific workflow scheduling in IaaS cloud," *IEEE Access*, vol. 7, pp. 125783–125795, 2019.

[30] H. Yuan, J. Bi, and M. Zhou, "Temporal task scheduling of multiple delay-constrained applications in green hybrid cloud," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1558–1570, Sep./Oct. 2021.

[31] H. Yuan, J. Bi, and M. Zhou, "Profit-sensitive spatial scheduling of multi-application tasks in distributed green clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 3, pp. 1097–1106, Jul. 2020.

[32] B. Tran, B. Xue, and M. Zhang, "Variable-length particle swarm optimization for feature selection on high-dimensional classification," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 473–487, Jun. 2019.

[33] D. Huanca and L. Pareja, "Chu and Beasley genetic algorithm to solve the transmission network expansion planning problem considering active power losses," *IEEE Latin America Trans.*, vol. 19, no. 11, pp. 1967–1975, Nov. 2021.

[34] S. Harford, F. Karim, and H. Darabi, "Generating adversarial samples on multivariate time series using variational autoencoders," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 9, pp. 1523–1538, Sep. 2021.

[35] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[36] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-assisted autoencoder-embedded evolutionary optimization algorithm to solve high-dimensional expensive problems," *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 676–689, Aug. 2022.
[37] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. V. Poor, "Self-adaptive bat algorithm with genetic operations," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.
[38] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1941–1954, Oct.–Dec. 2022.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA, in 2020.

He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC. He serves as an Associate Editor for *Expert Systems with Applications*.

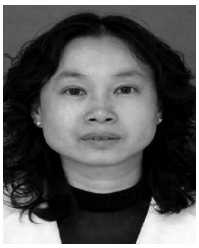
Qinglong Hu (Student Member, IEEE) received the B.S. degree in automation from Shandong University, Jinan, China, in 2021. He is currently pursuing the master's degree with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China.

His research interests include evolutionary computation, multiobjective optimization, high-dimensional optimization, deep learning, machine learning, cloud computing, and cloud manufacturing.

Jing Bi (Senior Member, IEEE) received the B.S. and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively.

From 2013 to 2015, she was a Postdoctoral Researcher with the Department of Automation, Tsinghua University, Beijing, China. From 2011 to 2013, she was a Research Scientist with the Beijing Research Institute of Electronic Engineering Technology, Beijing. From 2009 to 2010, she was a Research Assistant and participated in research on cloud computing with IBM Research, Beijing. From 2018 to 2019, she was a Visiting Research Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is currently a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing. She has over 150 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud and edge computing, large-scale data analytics, machine learning, industrial Internet, and performance optimization.

Dr. Bi was the recipient of the IBM Fellowship Award, the Best Paper Award in the 17th IEEE International Conference on Networking, Sensing, and Control, and the First-Prize Progress Award of Chinese Institute of Simulation Science and Technology. She is currently an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS: SYSTEMS.



Guanghong Gong received the Ph.D. degree in guidance, navigation and control from Beihang University, Beijing, China, in 1997.

She is currently a Full Professor with the School of Automation Science and Electrical Engineering, Beihang University. She has over 100 publications in international conference/journal papers. Her research interests include cloud manufacturing, distributed systems, cloud simulation, big data, machine learning, and deep learning.

Prof. Gong is a Council Member of the China Simulation Federation.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA.

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, the Professor with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science.

MengChu Zhou (Fellow, IEEE) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He was with New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. He has more than 1000 publications, including 12 books, more than 700 journal papers (more than 600 in IEEE transactions), 31 patents, and 30 book chapters. His interests are in Petri nets, automation, Internet of Things, cloud/edge computing, and AI.

Dr. Zhou is a Fellow of IFAC, AAAS, CAA, and NAI.