Engineering 7 (2021) 1274–1281

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
AI Energizes Process Manufacturing—Article

# Intelligent Ironmaking Optimization Service on a Cloud Computing Platform by Digital Twin

Heng Zhou, Chunjie Yang *, Youxian Sun

*The State Key Laboratory of Industrial Control Technology & College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China*

ARTICLE INFO

*Article history:*
Received 14 July 2020
Revised 18 September 2020
Accepted 2 April 2021
Available online 24 July 2021

*Keywords:*
Cloud factory
Blast furnace
Multi-objective optimization
Distributed computation

ABSTRACT

The shortage of computation methods and storage devices has largely limited the development of multi-objective optimization in industrial processes. To improve the operational levels of the process industries, we propose a multi-objective optimization framework based on cloud services and a cloud distribution system. Real-time data from manufacturing procedures are first temporarily stored in a local database, and then transferred to the relational database in the cloud. Next, a distribution system with elastic compute power is set up for the optimization framework. Finally, a multi-objective optimization model based on deep learning and an evolutionary algorithm is proposed to optimize several conflicting goals of the blast furnace ironmaking process. With the application of this optimization service in a cloud factory, iron production was found to increase by 83.91 $t \cdot d^{-1}$, the coke ratio decreased 13.50 $kg \cdot t^{-1}$, and the silicon content decreased by an average of 0.047%.



## 1. Introduction

Ironmaking is a complicated manufacturing system that continuously provides fundamental materials for other industries. The ironmaking industry plays an important role in the raw materials and energy markets, which utilize more than 10% of the global total energy consumption. As China has become the world's most important iron and steel supplier, with half of the total global production, it is of vital importance to achieve energy conservation and emission reduction in this industry by means of technological innovations [1].

Conventional approaches that address the complexity of transform theory and reaction rules, based on the ironmaking mechanism, have shortcomings in modeling and optimizing the blast furnace [2,3]. Therefore, researchers have investigated a variety of intelligent modeling methods for the ironmaking process. To address the advantages of black box models, Chen and Gao [4] developed a novel algorithm to enhance the transparency of a soft-margin support-vector machine for the blast furnace. Zhou et al. [5] proposed a recursive learning-based bilinear subspace identification algorithm to model and control a blast furnace with nonlinear time-varying dynamics. Furthermore, Li et al. [6] used a fuzzy classifier to judge the product quality and thermal state by forecasting the development tendency of the hot metal silicon content. In order to obtain gas flow distribution and optimize the charging operation, Huang et al. [7] designed a method for three-dimensional (3D) topography measurement based on a high-temperature industrial endoscope to detect the blast furnace burden surface. Li et al. [8] then proposed an intelligent data-driven optimization scheme to determine the proper burden surface distribution. Although all these methods have partially improved the operation of the blast furnace, the whole ironmaking process requires a cloud computing architecture integrated with all kinds of services.

Cloud computing offers the on-demand services of a computer system, computing power, and data storage to customers, without the customers needing to actually possess the associated objects [9]. The availability of mass-storage devices, low-cost computers, and high-capacity networks has led to the development of cloud computing, while the widespread evolution of hardware resources virtualization, autonomic utility computing, and service-oriented architecture has also contributed to the growth of cloud services [10]. In fact, many companies have deployed their business on cloud platforms, resulting in cloud services being involved in every aspect of our work and life. For example, the Yelp advertising team depends on prediction models to analyze the likelihood of a

* Corresponding author.
*E-mail address:* cjyang999@zju.edu.cn (C. Yang).

https://doi.org/10.1016/j.eng.2021.04.022

customer interacting with an advertisement. They use Apache Spark on Amazon Elastic MapReduce to process big data and train machine learning models, which has led to an increase in revenue and advertising click-through rate [11]. In the past, the 12306.cn booking system always broke down when the data volume increased dramatically at a specific time. Alibaba Cloud effectively solved this bottleneck problem by dealing with the remaining ticket queries through cloud computing during the Spring Festival [12].

With the fast development of high-integration and large-scale process industries, traditional methods are limited in modeling and optimizing the ironmaking process. Because it is difficult to combine conflicting goals in operation, one good indicator can always decrease the effect of another indicator. In order to optimize several goals simultaneously, we use a weighted method in a genetic algorithm (GA) to transform the multi-objective problem into a single-objective problem. Furthermore, a self-adaptive scheme is implemented in the GA to improve its searching performance. Before applying the GA to a practical problem, it is necessary to learn the algorithm's physical characteristics by means of modeling methods. To address the need for high accuracy and other real-time requirements, we simplify the structure of the recurrent neural network by compressing the update and reset gates into a single gate to model the ironmaking process. Thus, we design a multi-objective optimization framework for a cloud ironmaking plant based on distributed computation, as illustrated in Fig. 1. With this service example in a cloud factory, scholars and researchers that cooperate with ironmaking plants can work on the plant from anywhere in the world. Along with the explanation above, the major contributions of this paper can be briefly summarized as follows:

(1) An improved GA is implemented in conjunction with a recurrent neural network to optimize a multi-objective problem in the ironmaking process.

(2) A cloud computing platform for a digital twin is constructed based on the Rancher and Harbor frameworks.

(3) The hybrid multi-objective model is deployed to the cloud computing platform as an optimization service.

## 2. Methodology

An ironmaking factory generates a large amount of data during the manufacturing process from programmable logic controllers, industrial sensors, and local indicators [13]. There are hundreds of ironmaking plants and thousands of blast furnaces across China, and the data amount and computing demand of these factories are much greater than the factories can afford. For a blast furnace with thousands of measuring points sampled at the minutes level, the total data amount can be as large as a terabyte every day. However, these big industrial data are insufficiently exploited and become a burden for ironmaking plants. Even though major progress has been made in both cloud services and the ironmaking process, few academic studies or industrial applications have attempted to combine the two. Therefore, we have designed a distribution-system-based cloud factory, as shown in Fig. 2, to explore the hidden information by making full use of these big industrial data. At first, the process data are temporarily stored in local servers at the data center of the ironmaking factory. After some necessary pretreatment and reformatting, the clean data is directly transferred to the relational database. With its industrial data backed up on the cloud database, the factory does not need to expand its storage devices to scale to the increasing data volumes. Then, if we want to modify the algorithm and train datasets from the cloud storage, we can extract samples from either the local server or the cloud database.

The cloud ironmaking blast furnace consists of the storage, framework, and service layers. The storage layer backs up the stream data from the manufacturing system by means of a cloud relational database. The computing framework layer, which sits between the storage and service layers, contains a hybrid model based on deep learning and an evolutionary algorithm. Moreover, the computing cluster is an instance of Apache Spark and is associated with virtual machines from the local cloud service provider. With the support of a computing framework and storage capacity, the cloud factory is able to provide a multi-objective optimization service to the blast furnace ironmaking process. Finally, we use Apache Spark to deploy the multi-objective optimization framework on a virtual machine to provide cloud computing service for the cloud factory. Both the factory and academia can benefit from the cloud ironmaking plant: There is no need for the ironmaking plant to store all the manufacturing data, while academic researchers can work on practical data from other places. By using Apache Spark and cloud computing, we have successfully applied a multi-objective optimization service with clustering, modeling, and optimizing models to the blast furnace ironmaking process.

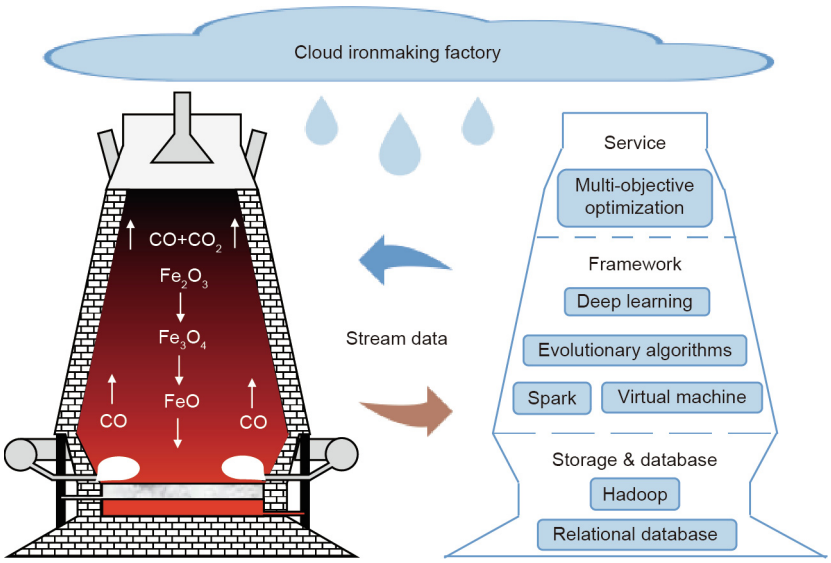


**Fig. 1.** Schematic architecture of a cloud ironmaking blast furnace.

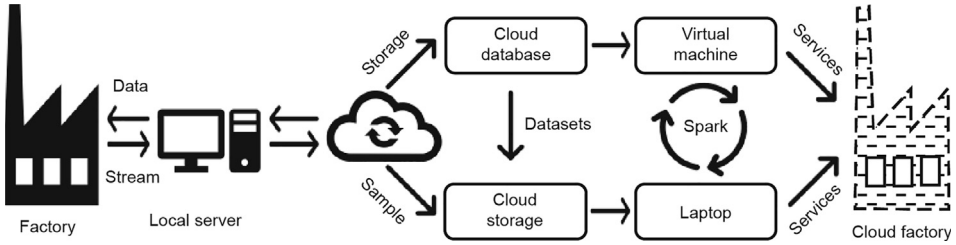


**Fig. 2.** System flowchart of a cloud factory interacting with a real-life factory.

The integration of the hybrid model combines the clustering, modeling, and optimizing procedures into an indivisible whole. Based on the preprocessing of the cluster analysis, modeling methods are used to obtain the dynamic information of the blast furnace ironmaking process. After that, optimization approaches search for real-time optimal solutions for production indices under critical constraints. Thus, the objective function of this optimization problem can be written as follows:

$$\begin{cases} \min F(x) = [w_1, \cdots, w_n] \cdot [f_1(x), \cdots, f_n(x)]^T \\ f(x) = g[\varphi(x_t, h_{t-1} | d_t)] \end{cases} \tag{1}$$

where $w$ is the weight matrix, $\varphi$ and $g$ are the activation functions, $d_t$ is the disposition gate, and $T$ is the generation. $F(x)$ is the comprehensive fitness function, while $f(x)$ is a single fitness function for each optimization object. $x_t$ is the input variable $x$ at $t$ moment, and $h_{t-1}$ is the hidden state at $t$-1 moment.

Clustering carries out the task of classifying a collection of objects into several groups, where objects belonging to the same cluster are more similar to each other than to the objects in other clusters [14,15]. It is the main task of exploratory data mining and a general approach for the statistical analysis used in machine learning. At the very beginning, we utilize a Gaussian mixture model (GMM) to group data into different clusters and select ideal datasets to realize knowledge discovery. A GMM is a distribution-based model that fits the datasets with a certain number of Gaussian functions [16,17]. The probability density function of the GMM is listed in Eq. (2).

$$\begin{aligned} p(x) &= \sum_{k=1}^{K} p(k) p(x|k) = \sum_{k=1}^{K} {}_k \cdot N(x|\mu_k, \Sigma_k) \\ &= \sum_{k=1}^{K} {}_k \frac{\exp\left[-0.5(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k)\right]}{\sqrt{(2)^K |\Sigma_k|}} \end{aligned} \tag{2}$$

where $k$ represents the $k$th cluster, $\mu$ is the mean, $\Sigma$ is the variance, $p$ is the probability density function, $N$ is the number of observations, $\pi$ is the mixture weight, and the sum of $\pi$ is 1.

When original data from the ironmaking process have been pretreated by GMM clustering, it is necessary to reconstruct the ironmaking process using intelligent approaches. Among a variety of deep learning methods, recurrent neural networks (RNNs) exhibit an excellent performance in learning the physical characteristics of time-series data [18,19]. However, the structure of the traditional gated recurrent-unit recurrent neural network (GRU-RNN) is somewhat complicated and cannot fulfill the high real-time requirements of the process industries [20]. Therefore, a novel RNN is proposed for the ironmaking process by simplifying the update and reset gates of the GRU-RNN into a single disposition gate [21]. The mathematical definitions of the disposition gated recurrent unit (dGRU) are illustrated in Eq. (3), and a visualization of the internal architecture is provided in Fig. 3.

$$\begin{cases} h_t = (1 - d_t) \odot h_{t-1} + d_t \odot h_t \\ h_t = \tanh\left[W x_t + U(d_t \odot h_{t-1})\right] \\ d_t = \sigma(W_d x_t + U_d h_{t-1}) \end{cases} \tag{3}$$

where the activation state $h_t$ is a linear interpolation between the candidate activation state $h_t$ and the previous activation state $h_{t-1}$. $\sigma$ is the activation function. $U$ and $W$ are the weights of input variable and activation threshold. $W_d$ and $U_d$ represent the weight and threshold of disposition gate. The $d_t$ gate makes compromises between the historical memory $h_{t-1}$ and the candidate information $x_t$. The information flowing in and out of the unit in the dGRU is manipulated by only one gate, resulting in an increase in computing efficiency.

The back propagation of the dGRU transforms errors backward in time and space to update its parameters. Therefore, the back propagation error at time $t-1$ can be written as Eq. (4).

$$\begin{aligned} \delta_{t-1} &= \Delta\delta_{t-1}^{l+1} + \delta_{t-1}^{l} = \Delta\delta_{t-1}^{l+1} + \frac{\partial E}{\partial h_{t-1}} \\ &= \Delta\delta_{t-1}^{l+1} + \frac{\partial E}{\partial \mathrm{net}_{d,t}} \frac{\partial \mathrm{net}_{d,t}}{\partial h_{t-1}} + \frac{\partial E}{\partial \mathrm{net}_{\tilde{h},t}} \frac{\partial \mathrm{net}_{\tilde{h},t}}{\partial h_{t-1}} + \frac{\partial E}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \\ &= \Delta\delta_{t-1}^{l+1} + \delta_{d,t} U_d + \left(\delta_{\tilde{h},t} U - \delta_t\right) \odot d_t + \delta_t \end{aligned} \tag{4}$$

where $E$ is the output estimation, and net is the dGRU network. $\delta$ is the output error, $l$ represents the step, $d$ is the disposition gate, and $\tilde{h}$ represents the candidate state. The internal errors $\delta_{d,t}$ and $\delta_{h,t}$ can be transformed into the expressions in Eq. (5).

$$\begin{aligned} \delta_{d,t} &= \frac{\partial E}{\partial \mathrm{net}_{d,t}} = \frac{\partial E}{\partial h_t} \frac{\partial h_t}{\partial d_t} \frac{\partial d_t}{\partial \mathrm{net}_{d,t}} \\ &= \delta_t \odot \left(\tilde{h}_t - h_{t-1}\right) \odot d_t \odot (1 - d_t) \\ \delta_{\tilde{h},t} &= \frac{\partial E}{\partial \mathrm{net}_{\tilde{h},t}} = \frac{\partial E}{\partial h_t} \frac{\partial h_t}{\partial \tilde{h}_t} \frac{\partial \tilde{h}_t}{\partial \mathrm{net}_{\tilde{h},t}} \\ &= \delta \odot d_t \odot \left(1 - \tilde{h}_t^2\right) \end{aligned} \tag{5}$$

Thus, the gradients to update the weights and thresholds in $d_t$ and $h_t$ have the prototypes shown in Eq. (6). With this learning

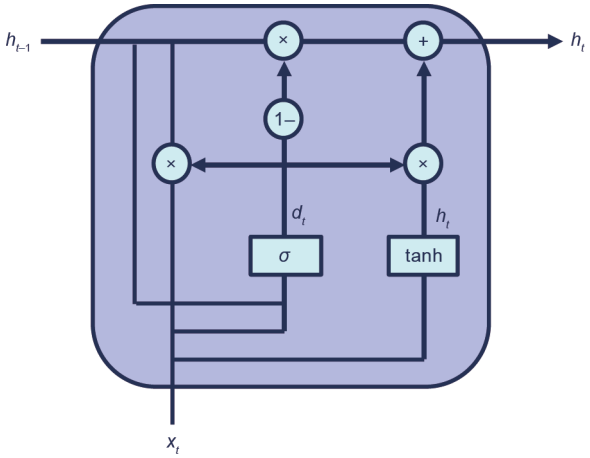


**Fig. 3.** Internal structure of the dGRU.

schema, the dGRU is able to optimize its internal structure during the iterations.

$$\begin{aligned}\Delta U_d &= \frac{\partial E}{\partial U_d} = \frac{\partial E}{\partial \mathrm{net}_{d,t}} \frac{\partial \mathrm{net}_{d,t}}{\partial U_d} = \delta_{d,t} h_{t-1} \\ \Delta W_d &= \frac{\partial E}{\partial W_d} = \frac{\partial E}{\partial \mathrm{net}_{d,t}} \frac{\partial \mathrm{net}_{d,t}}{\partial W_d} = \delta_{d,t} x_t \\ \Delta U &= \frac{\partial E}{\partial U} = \frac{\partial E}{\partial \mathrm{net}_{h,t}} \frac{\partial \mathrm{net}_{h,t}}{\partial U} = \delta_{h,t} (d_t \odot h_{t-1}) \\ \Delta W &= \frac{\partial E}{\partial W} = \frac{\partial E}{\partial \mathrm{net}_{h,t}} \frac{\partial \mathrm{net}_{h,t}}{\partial W} = \delta_{h,t} x_t\end{aligned} \tag{6}$$

where $\Delta W_d$ and $\Delta W$ are the weights of input variables in $d_t$ gate and candidate activation state, $\Delta U_d$ and $\Delta U$ are the weights of activation threshold in $d_t$ gate and candidate activation state.

Among the evolutionary algorithms, the GA is a global optimum approach to solve combined optimization problems [22]. The major elements of a GA are encoding/decoding types, fitness function, genetic operators, and control parameters; more specifically, a GA can perform the following tasks:

- Encoding the solutions of optimization issues;
- Creating a population that includes $N(t)$ encoded solutions at the $t$th generation;
- Establishing a fitness function that evaluates the optimality of solutions;
- Using genetic operators to generate offspring to produce a new population;
- Setting control parameters.

In recent years, most investigations on GAs have focused on probability distribution, genetic operators, and chromosome encoding [23,24]. Population size receives little attention, even though it dramatically influences the computing efficiency. Generally speaking, the population size is directly proportional to the solution accuracy and inversely proportional to the computational efficiency. To address this issue, Koumousis and Katsaras [25] proposed a saw-tooth GA whose population size increased and decreased periodically. To maintain accuracy and efficiency at the same time, we designed a self-adaptive population genetic algorithm (SAPGA) whose population size changes along with the solutions of fitness function [26]. The ideal fitness distribution in each generation is supposed to be a normal distribution, which is always difficult to implement due to the randomness of genetic operators. The degree of deviation from a real distribution to a normal one is defined as the skewness ($S_k$) in Eq. (7).

$$S_k = \frac{\bar{f} - f_e}{\sigma} \tag{7}$$

where $f_e$ is the media, $\bar{f}$ is the average, and $\sigma$ is the standard deviation. The relationship between the skewed distribution and population size is visualized in Fig. 4, with an explanation in Eq. (8).

$$\begin{cases} N(t+1) = N(t) + k, \ S_k(t) < 0 \\ N(t+1) = N(t) - k, \ S_k(t) > 0 \end{cases} \tag{8}$$

where $S_k$ is the fitness skewness, $N$ is the population size, and $k$ is an integer of the population size transformation interval. As for the SAPGA, the rising tendency of $S_k$ represents an increase in superior solutions, leading to a demand for new candidates to increase the genetic diversity. The decline of $S_k$ indicates that there is a majority of inferior solutions, which need to be eliminated in order to keep the SAPGA in good searching performance. To summarize, when the skewness of the fitness distribution turns from negative to positive, it signals that the proportion of superior solutions increase, or the inferior individuals decrease, and the fitness of the candidates needs to be adjusted back to a normal distribution.

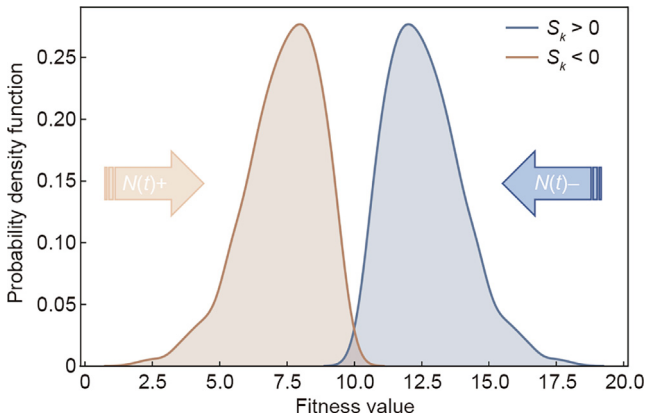


**Fig. 4.** Correspondence between fitness skewness and population size. $N(t)$+ and $N(t)$− represent the increase and decrease processes of population during the evolution.

To compare the performance of the SAPGA and a standard genetic algorithm (SGA), a theoretical analysis for the fitness of solutions based on the symbols in Table 1 has been made according to pattern theory.

In the latter stage of the evolutionary process, the increasing of the mean fitness causes decreasing in the population size. $T$ is the generation at which the population of the SAPGA becomes less than that of the SGA. Therefore, we state a lemma defined as follows.

Lemma : $N(t)_{\mathrm{SAPGA}} = N(t)_{\mathrm{SGA}} - k$, where $t > T$.

Due to the influence of mutation and crossover, the expected number of offspring produced by candidate $i$ under pattern $h$ is given in Eq. (9) and the definitions of symbols are shown in Table 1.

$$n_i^h(t+1) \geq \frac{f_i}{\bar{f}} \left[ 1 - \frac{l(h)}{L-1} p_c \right] (1 - n p_m) \tag{9}$$

In the $t$th generation, the fitness at order $n$ under pattern $h$ can be expressed as Eq. (10).

$$M_h^n(t) = \frac{1}{N_h(t)} \sum_{i \in h} (f_i)^n \tag{10}$$

The expected mean fitness of pattern $h$ in the $(t+1)$th generation consists of two parts. One is the original solution inherited from the parent population without destruction; the other is the new solution produced by recombination operations.

The mean fitness of the solutions under pattern $h$ in the $(t+1)$th generation is represented by Eq. (11).

**Table 1**
List of symbols in the article.

| Symbol | Content |
|---|---|
| $h$ | Pattern |
| $f_i$ | Fitness value for candidate $i$ |
| $\bar{f}$ | Mean fitness value |
| $n_i^h(t+1)$ | Expected quantity of offspring |
| $N_h(t)$ | Quantity of solutions |
| $l(h)$ | Defined length of pattern |
| $L$ | Encoding length of candidate |
| $p_c$ | Crossover probability |
| $p_m$ | Mutation probability |
| $n$ | Order of pattern |
| $k$ | A positive integer |
| $t$ | $t$th generation |

$$M_h^q(t+1)_{\mathrm{SAPGA}} = \frac{\sum_{i\in h} n_i^h(t+1) f_i + \sum_{j\in h} n_j f_j}{\sum_{i\in h} n_i^h(t+1) + \sum_{j\in h} n_j} = \frac{\sum_{i\in h} (f_i)^{q+1}\left[1-\frac{l(h)}{L-1}p_c\right](1-np_m)/\bar{f} + \sum_{j\in h} n_j f_j}{\sum_{i\in h} (f_i)^{q}\left[1-\frac{l(h)}{L-1}p_c\right](1-np_m)/\bar{f} + \sum_{j\in h} n_j} = \frac{M_h^{q+1}(t)[N_h(t)-k]\alpha + \sum_{j\in h} n_j f_j}{M_h^{q}(t)[N_h(t)-k]\alpha + \sum_{j\in h} n_j} \tag{11}$$

where $q$ is the selected candidates to be measured by pattern theory; $\alpha = \left[1-\frac{l(h)}{L-1}p_c\right](1-np_m)/(S_k\sigma + f_e)$

As a comparison, the corresponding prototype for the mean fitness of the solutions under pattern $h$ in the SGA is given in Eq. (12) [27].

$$M_h^q(t+1)_{\mathrm{SGA}} = \frac{M_h^{q+1}(t)N_h(t)\beta + \sum_{j\in h} n_j f_j}{M_h^{q}(t)N_h(t)\beta + \sum_{j\in h} n_j} \tag{12}$$

where $\beta = (1-p_c)(1-p_m)^n$

From Eqs. (11) and (12), it is obvious that the SAPGA has a higher mean fitness than the SGA under pattern $h$. Therefore, the self-adaptive population schema promotes the SAPGA to have more accurate solutions and a faster convergence speed.

$$M_h^q(t+1)_{\mathrm{SAPGA}} \geq M_h^q(t+1)_{\mathrm{SGA}} \tag{13}$$

## 3. Experiments and results

The data in this section were originally collected from one blast furnace with a working space of 2650 $m^3$. When algorithms needed to be verified before going online, we obtained datasets from the local server by sampling from the Oracle database. As data quality is one of the most important issues in identifying data sources, the GMM method is used to group the datasets into different categories and eliminate the one that does not satisfy the high data-quality requirements. The Calinski–Harabasz index [28], a clustering evaluation function shown in Eq. (14), is taken into account to determine the optimal number of cluster centers. A higher Calinski–Harabasz index represents a better clustering performance, which is caused by less covariance inside the cluster and greater covariance between clusters. Therefore, it is easy to determine that the best cluster number in Fig. 5 is 4. Moreover, there are three working patterns in the ironmaking process: the ascended, descended, and stationary operation trends.

Therefore, the cluster with the worst performance is eliminated to adjust the iron production conditions.

$$s(m) = \frac{\mathrm{tr}(B_m)}{\mathrm{tr}(W_m)}\frac{k-m}{m-1} \tag{14}$$

where $k$ is the number of samples, $m$ is the number of clusters, $B_m$ is the covariance matrix between different clusters, $W_m$ is the covariance inside the cluster, and tr is the trace of the matrix.

The samples taken from the cloud database are divided into four groups. We take the silicon content and the top pressure as the representation of the clusters in Fig. 6. The yellow cluster has a wide distribution in top pressure, which is abnormal in blast furnace operation. To make the top gas pressure recovery turbine work efficiently, the blast furnace top pressure must remain stable and fluctuate within a narrow range. Therefore, the cluster represented by a widely distributed top pressure is deleted from the datasets to increase the data quality.

After the elimination of the unsatisfied cluster, the dGRU is verified by the remaining 2000 samples, 20% of which are divided into a test dataset. The input arguments of the deep neural network are the permeability index, $CO_2$, CO, bosh gas index, theoretical combustion temperature, top temperature northeast, top temperature southwest, top temperature northwest, and top temperature southeast. In Fig. 7, it is shown that the dGRU-RNN has an excellent performance in following the changing trend of the output arguments, including the silicon content, coke ratio, and iron yield. The comparison between the practical data and the predicted data indicates that the dGRU units have high accuracy and fast convergence. As listed in Table 2, a root mean square error (RMSE) of 0.025 in the silicon content means that the dGRU-RNN has an extraordinary ability to learn the physical dynamics of the ironmaking process.

It seems that the fluctuation of the coke ratio is more balanced than the silicon content and iron yield. To obtain a more intuitive understanding of the simulation results, the predicted coke ratio with its corresponding error are displayed in Fig. 8. It can be seen that the dGRU-RNN can track the change trend of the coke ratio with minor errors most of the time. However, several major errors at a high coke ratio indicate that the deep neural network model can have low accuracy when encountering a sudden change. Therefore, the dGRU has great potential in dealing with a steady industry process such as blast furnace ironmaking.

Many variants of the SGA exist, including the adaptive genetic algorithm (AGA) and the genetic algorithm with simulated annealing mutation probability (SAMGA) [29,30]. Four numerical test functions (Eqs. (15)–(18)), characterized by single object optimization, are conducted to compare the performance of the SAPGA and SGA, as well as other GA variants. From Fig. 9, we can find Eq. (15) with a minimum at (0, 0), Eq. (16) with a minimum at (3, 0.5), Eq. (17) with a minimum at ($\pi$, $\pi$), and Eq. (18) with four minimums at (3, 2), (−2.805, 3.131), (−3.779, −3.283) and (3.584, −1.848), respectively.

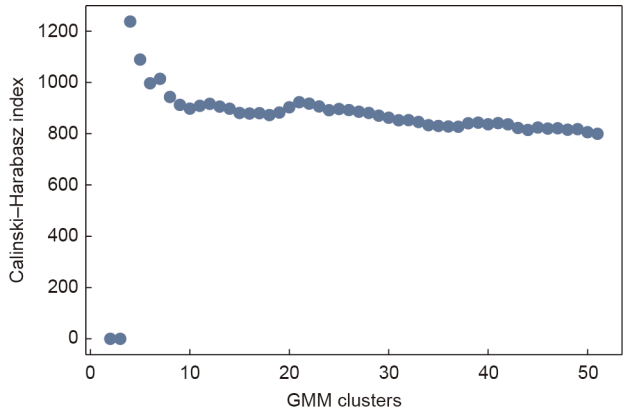


**Fig. 5.** Correlation between the evaluation index and cluster numbers.

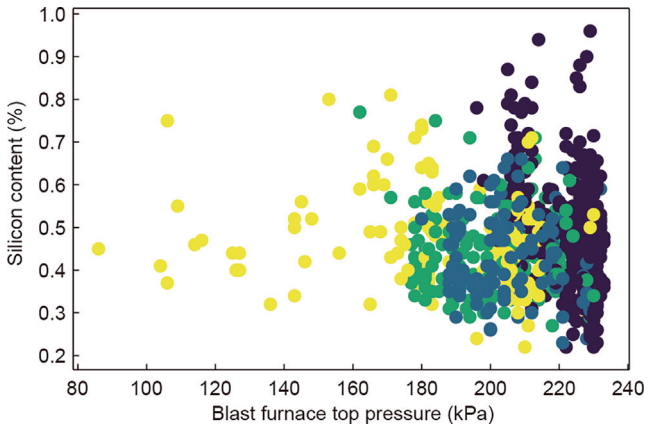


**Fig. 6.** Representation of the four-component GMM in the presence of clusters.

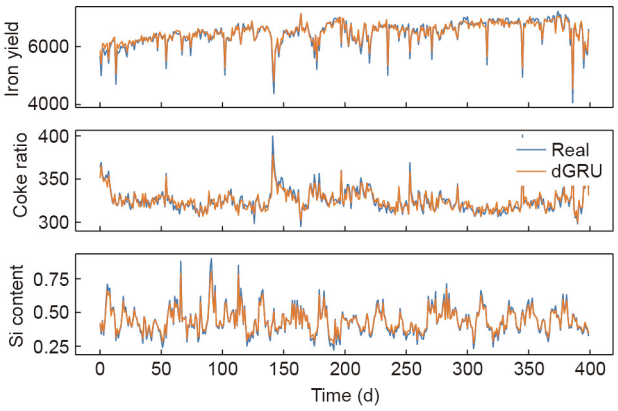


**Fig. 7.** Prediction results of the dGRU-RNN for multiple ironmaking production indices.

**Table 2**
A comparison of predicted and practical data.

| Index | Error | | | |
|---|---|---|---|---|
| | RMSE | MSE | MAE | SD |
| Iron yield | 123.5 | 15251 | 94.91 | 360.02 |
| Coke ratio | 4.60 | 21.13 | 3.75 | 12.40 |
| Si content | 0.025 | 0.0006 | 0.018 | 0.096 |

MSE: mean square error; MAE: mean absolute error; SD: standard deviation.

$$f = 0.26(x^2 + y^2) - 0.48xy \tag{15}$$

$$f = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \tag{16}$$

$$f = -\cos(x)\cos(y)\exp[-(x - \pi)^2 - (y - \pi)^2] + 1 \tag{17}$$

$$f = (x^2 + y - 11)^2 + (x + y^2 - 7)^2 \tag{18}$$

The optimizing processes of Eq. (18) by four GA variants are shown in Fig. 10, in which the SAPGA has the most outstanding performance in terms of both searching accuracy and convergence speed. The SAPGA can converge to the optimum within five iterations, which is far less than other GAs. Under some conditions, the SGA—as well as other GAs—can be stuck in a local extreme. However, the SAPGA was seldom stuck in local optimums during the test, due to its self-adaptive scheme. The combined fitness skewness and population size schema lead to a typical phenomenon in which the number of solutions will first increase and then decrease during the evolution process, which contributes to avoiding local extreme and improving computation efficiency.

To obtain a more intuitive visualization of the optimization effect, the optimized solutions of Eq. (18) at a certain generation are presented in Fig. 11 by transforming the 3D graph to the two-dimensional plane. The results indicate that the SGA has the widest population distribution around optimal points at the final stage. However, the SAPGA has highly concentrated solutions and can find all four optimal points, displaying a much better performance than other GA variants. Thus, the SAPGA is demonstrated to be more powerful than the other three typical GA variants in optimizing the single-objective optimization problem.

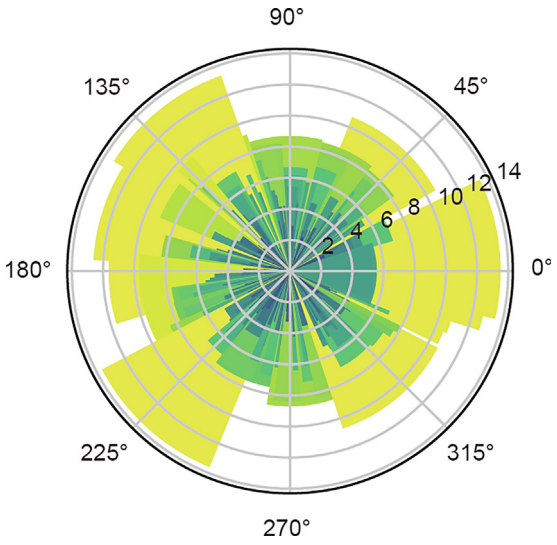


**Fig. 8.** Coke ratio error bar chart on the polar axis. Widths represent the predicted coke ratio, and the radii are their corresponding errors.

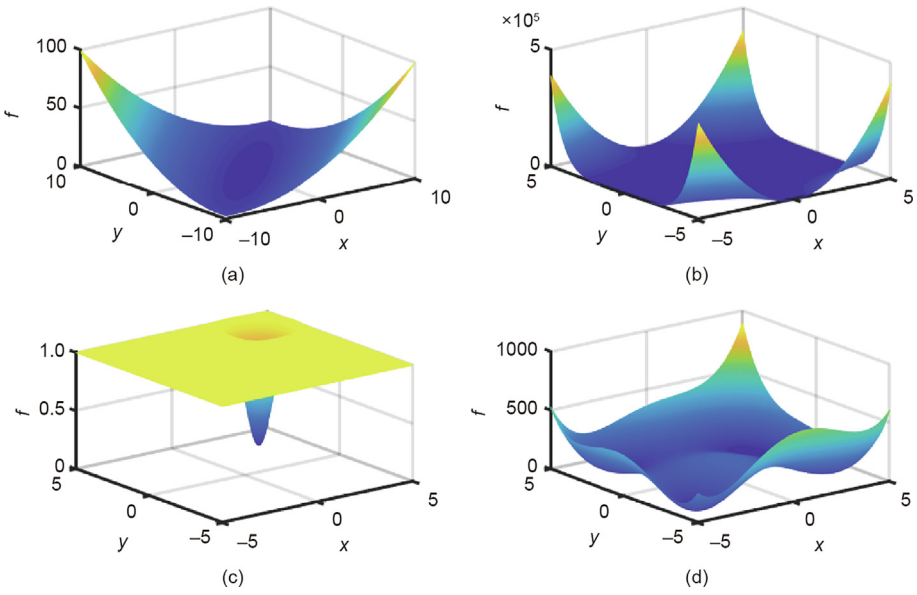


**Fig. 9.** 3D graph of the test functions for GA variants. $f$ is the fitness of GA in each evolution.

After the validation of the SAPGA, it is merged in conjunction with the dGRU to optimize the actual ironmaking process. As for the blast furnace, the output and input sequences can vary significantly due to different producing conditions. The iron quality silicon content and the energy consumption coke ratio are determined by the operation level of the field engineers, while the iron yield can change dynamically according to current and future market situations. Conflicting goals always exist among material suppliers, iron producers, and steel customers. Therefore, it is of vital importance for an ironmaking plant to depend on a multi-objective optimization algorithm to balance these contradictory objectives. Before uploading the model to a cloud factory system, we must verify the multi-objective optimization algorithm by means of three necessary procedures. First, a hybrid framework based on the conventional GRU-RNN and the SGA is applied to optimize the process indexes in the blast furnace. After that, the improved algorithms replace the traditional ones and optimize the ironmaking process in the time and space scale. Finally, we deploy the verified dGRU-SAPGA to a distribution computing system Spark in Standalone mode.

To take the randomness of GAs into consideration, each test is repeated 100 times. As shown in Fig. 12, the Spark Standalone system has a huge advantage in terms of computing efficiency. The median running time of the dGRU-SAPGA at the distribution system is 0.04 s, which is far less than the 0.10 s at the local server. During local tests, the modified hybrid framework dGRU-SAPGA runs slightly faster than the GRU-SGA. However, there are more outliers of the dGRU-SAPGA than of the GRU-SGA, which means that the latter algorithm is more stable than the former one.

We have deployed the verified hybrid framework dGRU-SAPGA to the cloud factory for more than two months. As illustrated in Fig. 13, the iron production has increased by 1.29%, the coke ratio has decreased by 3.60%, and the silicon content has decreased by an average of 10.61%.

## 4. Conclusions

A highly integrated and large-scale ironmaking process manufacturing factory requires a timely response and an elastic computing system to deal with a variety of working conditions. Conventional methods are limited in the ironmaking process; therefore, a hybrid-model-based distribution computation method was proposed to optimize the conflicting objectives of the blast furnace ironmaking process. In the local mode, the dGRU-SAPGA exhibits a competitive performance in modeling and optimizing the ironmaking process. Based on three-step verification, the hybrid optimization framework in the Spark distribution system was applied to the #2 blast furnace of Guangxi Liuzhou Iron and Steel Group Co., Ltd. After two months of application, the multiple production indicators of the blast furnace were significantly improved. It should be noted, however, that the multi-objective optimization service alone is insufficient for the ironmaking process. The distribution computation-based cloud factory is in need of many other services, including intelligent detection, data fusion, fault diagnosis, and advanced control.

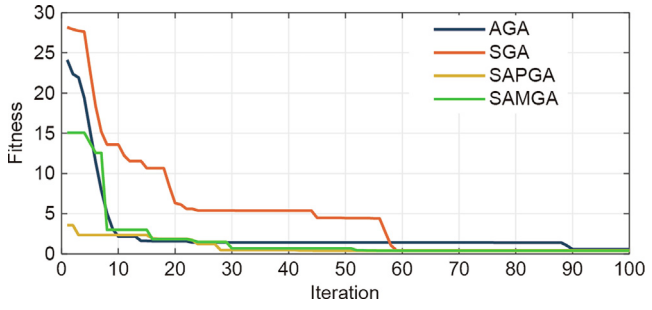


**Fig. 10.** Fitness iteration of GAs on test function.

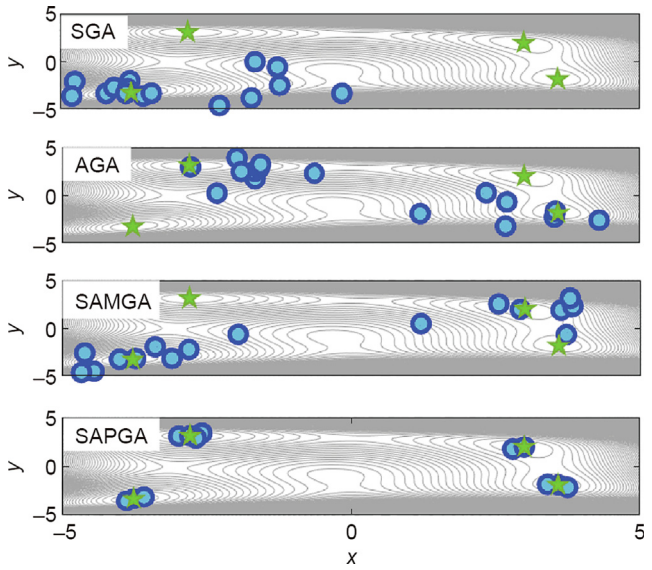


**Fig. 11.** Solution distributions of GAs for test Eq. (18) terminated at the 20th generation.

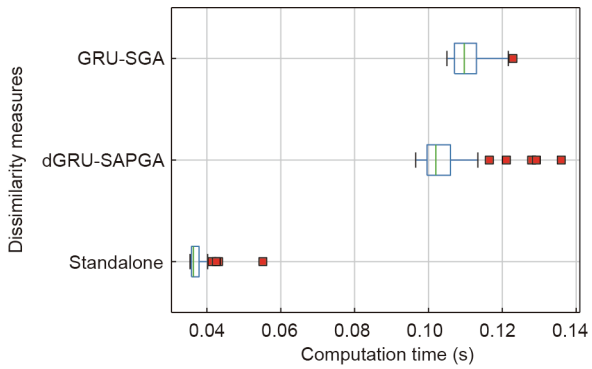


**Fig. 12.** The computation efficiency of multi-objective optimization by different methods.

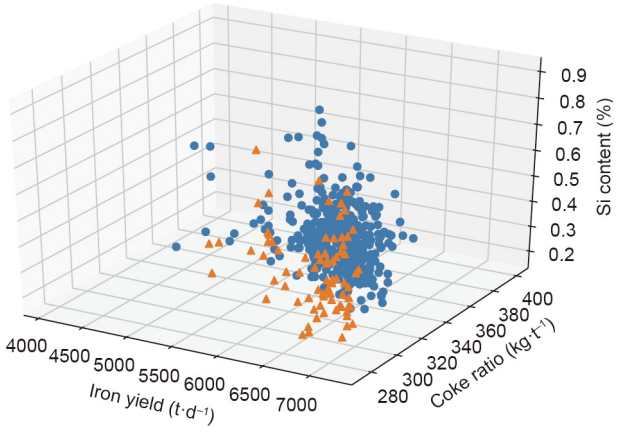


**Fig. 13.** Effect of multi-objective optimization service in the blast furnace cloud factory.

## Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (61933015). The authors would like to thank Guangxi Liuzhou Iron and Steel Group Co., Ltd.: This article has no chance of being successful without its support and cooperation. The author Heng Zhou would also like to thank his lab-mates for their dedication and sacrifice.

## Compliance with ethics guidelines

Heng Zhou, Chunjie Yang, and Youxian Sun declare that they have no conflict of interest or financial conflicts to disclose.

## References

[1] Pan Y, Yang C, An R, Sun Y. Robust principal component pursuit for fault detection in a blast furnace process. Ind Eng Chem Res 2018;57(1):283–91.
[2] Zhou P, Song H, Wang H, Chai T. Datadriven nonlinear subspace modeling for prediction and control of molten iron quality indices in blast furnace ironmaking. IEEE Trans Control Syst Technol 2017;25(5):1761–74.
[3] Ge Z, Song Z, Ding SX, Huang B. Data mining and analytics in the process industry: the role of machine learning. IEEE Access 2017;5:20590–616.
[4] Chen S, Gao C. Linear priors mined and integrated for transparency of blast furnace black-box svm model. IEEE Trans Industr Inform 2020;16(6):3862–70.
[5] Zhou P, Zhang S, Dai P. Recursive learning-based bilinear subspace identification for online modeling and predictive control of a complicated industrial process. IEEE Access 2020;8:62531–41.
[6] Li J, Hua C, Yang Y, Guan X. Fuzzy classifier design for development tendency of hot metal silicon content in blast furnace. IEEE Trans Industr Inform 2018;14 (3):1115–23.
[7] Huang J, Chen Z, Jiang Z, Gui W. 3D topography measurement and completion method of blast furnace burden surface using high-temperature industrial endoscope. IEEE Sens J 2020;20(12):6478–91.
[8] Li Y, Zhang S, Zhang J, Yin Y, Xiao W, Zhang Z. Data-driven multiobjective optimization for burden surface in blast furnace with feedback compensation. IEEE Trans Industr Inform 2020;16(4):2233–44.
[9] Chen J, Hu K, Wang Qi, Sun Y, Shi Z, He S. Narrowband Internet of Things: implementations and applications. IEEE Internet Things J 2017;4(6):2309–14.
[10] Linthicum DS. Connecting fog and cloud computing. IEEE Cloud Comput 2017;4(2):18–20.
[11] Iqbal MA, Aleem M, Ibrahim M, Anwar S, Islam MA. Amazon cloud computing platform EC2 and VANET simulations. Int J Ad Hoc Ubiquitous Comput 2019;30(3):127–36.
[12] Zhang G, Ravishankar MN. Exploring vendor capabilities in the cloud environment: a case study of Alibaba Cloud Computing. Inf Manage 2019;56 (3):343–55.
[13] He W, Qian F, Lam J, Chen G, Han QL, Kurths J. Quasi-synchronization of heterogeneous dynamic networks via distributed impulsive control: error estimation, optimization and design. Automatica 2015;62:249–62.
[14] Zhu X, Zhu Y, Zheng W. Spectral rotation for deep one-step clustering. Pattern Recognit 2020;105:107175.
[15] Zhao C, Sun Y. Step-wise sequential phase partition (SSPP) algorithm based statistical modeling and online process monitoring. Chemom Intell Lab Syst 2013;125:109–20.
[16] Liu JW, Ren ZP, Lu RK, Luo XL. GMM discriminant analysis with noisy label for each class. Neural Comput Appl 2021;33(4):1171–91.
[17] Sabetsarvestani Z, Renna F, Kiraly F, Rodrigues M. Source separation with side information based on Gaussian mixture models with application in art investigation. IEEE Trans Signal Process 2020;68:558–72.
[18] Li Z, Yang C, Liu W, Zhou H, Li Y. Research on hot metal Si-content prediction based on LSTM-RNNl. CIESC J 2018;69:992–7.
[19] Che Z, Purushotham S, Cho K, Sontag D, Liu Y. Recurrent neural networks for multivariate time series with missing values. Sci Rep 2018;8(1):6085.
[20] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. Comput Sci 2014.
[21] Zhou H, Zhang H, Yang C. Hybrid-model-based intelligent optimization of ironmaking process. IEEE Trans Ind Electron 2020;67(3):2469–79.
[22] Whitley D. A genetic algorithm tutorial. Stat Comput 1994;4(2):65–85.
[23] Bozkurt E, Perez MAS, Hovius R, Browning NJ, Rothlisberger U. A genetic algorithm based design and experimental characterization of a highly thermostable metalloprotein. J Am Chem Soc 2018;140(13):4517–21.
[24] Yang C, Zhou H, Li Z. A multi-objective optimization model based on long short-term memory and non-dominated sorting genetic algorithm II. In: Proceedings of 2017 Chinese Automation Congress (CAC); 2017 Oct 20–22; Jinan, China. New York: IEEE; 2017. p. 1635–40.
[25] Koumousis VK, Katsaras CP. A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Trans Evol Comput 2006;10(1):19–28.
[26] Zhou H, Li Y, Yang C, Sun Y. Mixed-framework-based energy optimization of chemi-mechanical pulping. IEEE Trans Industr Inform 2020;16(9):5895–904.
[27] Srinivas M, Patnaik LM. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans Syst Man Cybern 1994;24(4):656–67.
[28] Łukasik S, Kowalski PA, Charytanowicz M, Kulczycki P. Clustering using flower pollination algorithm and Calinski–Harabasz index. In: Proceedings of 2016 IEEE Congress on Evolutionary Computation (CEC); 2016 Jul 24–29; Vancouver, BC, Canada. New York: IEEE; 2016. p. 2724–8.
[29] Adler D. Genetic algorithms and simulated annealing: a marriage proposal. In: Proceedings of IEEE International Conference on Neural Networks; San Francisco, CA, USA. New York: IEEE; 1993. p. 1104–9.
[30] Yi JH, Deb S, Dong J, Alavi AH, Wang GG. an improved NSGA-III algorithm with adaptive mutation operator for big data optimization problems. Future Gener Comput Syst 2018;88:571–85.