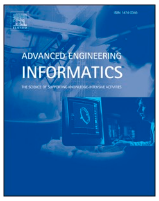
Contents lists available at ScienceDirect

Advanced Engineering Informatics

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



Building surrogate models for engineering problems by integrating limited simulation data and monotonic engineering knowledge

Jia Hao [a], Wenbin Ye [a], Liangyue Jia [a], Guoxin Wang [a], Janet Allen [b]

[a] *Beijing Institute of Technology, No. 5 South Street, Zhongguancun, Haidian District, Beijing, China*
[b] *The University of Oklahoma, 60 Parrington Oval, Norman, OK, USA*

ARTICLE INFO

Keywords:
Surrogate model
Limited simulation data
Engineering knowledge
Evolutionary neural network
Neuro Evolution of augmenting topologies (NEAT)

ABSTRACT

The use of surrogate models to replace expensive computations with computer simulations has been widely studied in engineering problems. However, often only limited simulation data is available when designing complex products due to the cost of obtaining this kind of data. This presents a challenge for building surrogate models because the information contained in the limited simulation data is incomplete. Therefore, a method for building surrogate models by integrating limited simulation **da**ta and engineering **know**ledge with **e**volutionary neural networks (eDaKnow) is presented. In eDaKnow, a neural network uses an evolutionary algorithm to integrate the simulation data and the monotonic engineering knowledge to learn its weights and structure synchronously. This method involves converting both limited simulation data and engineering knowledge into the respective fitness functions. Compared with the previous work of others, we propose a method to train the surrogate model by combining data and knowledge through evolutionary neural network. We take knowledge as fitness function to train the model, and use a network structure self-learning method, which means that there is no need to adjust the network structure manually. The empirical results show that: (1) eDaKnow can be used to integrate limited simulation data and monotonic knowledge into a neural network, (2) the prediction accuracy of the newly constructed surrogate model is increased significantly, and (3) the proposed eDaKnow outperforms other methods on relatively complex benchmark functions and engineering problems.

1. Introduction

In many engineering problems, it is difficult to define the explicit relationship between design variables and performance metrics [1]. Therefore, computer simulations are used to evaluate performance metrics when different design variables are used [2]. For example, to find an optimal airfoil shape, an engineer simulates the air flow around an airfoil for different configurations of design variables (length, curvature, material, etc.). However, using computer simulation is usually time-consuming and expensive, and this is often a limiting factor when using computer simulations in conceptual design where proposed design solutions require rapid evaluation and iteration.

One way to address this issue is to use surrogate models (also known as metamodels) [3] to replace computer simulations. Surrogate models offer the advantage of low computational complexity [4–6]. When a dataset is available with most of the information regarding the mapping between design variables and performance metrics, methods such as Kriging [7–10], radial basis functions [11], Gaussian processes [12] and others can be adopted easily because numerous commercial and open-source packages are available. However, often only limited simulation data are available, as collecting data can be extremely expensive, and thus, the information about the mapping relationship between design variables and performance metrics is inadequate. This makes building an appropriate surrogate model difficult. The key problem is that the limited simulation data provides limited information, so incorporating additional information would be helpful [13]. Fortunately, designers often have their own background knowledge and expertise, which is useful information for building a surrogate model. Utilizing and integrating this engineering knowledge provides a way to address the difficulties caused by limited simulation data. Therefore, we propose a method to integrate monotonic engineering knowledge and limited simulation data by using an evolutionary neural network (ENN) for constructing surrogate models. We transform engineering knowledge and limited simulation data into fitness functions and combine them in an iterative process using an evolutionary neural network (ENN) based on NSGA-II[14].

E-mail addresses: Haojia632@bit.edu.cn (J. Hao), 2631730625@qq.com (W. Ye), 719834513@qq.com (L. Jia), wangguoxin@bit.edu.cn (G. Wang), janet.allen@ou.edu (J. Allen).

https://doi.org/10.1016/j.aei.2021.101342
Received 26 October 2020; Received in revised form 8 May 2021; Accepted 16 May 2021
Available online 8 July 2021

Table 1
Critical evaluation of the existing literature.

Features Paper	**Number of Items of knowledge**			**Number of Data Points**		**Engineering Application**	
	Single	Several	Multiple	Small	Normal	Engineering	Other
Abu, Y.S., 1990[18]	√				√		√
Morris, 1993[24]	√				√		√
Towell,1994[15]	√	√			√		√
Williams, 1996[25]	√			√			√
Sill, J, 1998 [16]	√	√			√		√
Solak, 2003[26]	√			√			√
Lang, B., 2005[17]	√	√			√		√
Riihimäki, 2010 [19]	√				√		√
Daniels, 2010 [20]	√				√		√
Stephenson, 2010[27]	√			√		√	
Lenk, 2017[29]	√			√			√
Wang and Welch, 2018[30]	√	√			√		√
Hao, J., and Ye,2018 [22]	√	√		√			√
Gupta, 2018[21]	√	√	√		√	√	
Yang, 2021[23]	√	√			√	√	
This Work	√	√	√	√		√	

*Single = One Item of Knowledge; Several = Two-Four Items of knowledge; Multiple = Five or More Items of knowledge.

The rest of the paper is structured as follows: in section 2, related work is reviewed. In Section 3 the proposed method is explained in detail. In Section 4 several experiments are conducted to verify the proposed method. A discussion based on empirical results is presented in Section 5, and this work is summarized in Section 6.

2. Related work

Currently in engineering domain, there are few related researches on the fusion of knowledge and data. Some methods in the domain of machine learning provide a technical basis for the research of this article. However, some methods have been proposed in the domains of artificial intelligence and machine learning[13]. In the literature, scholars use different terms to refer to engineering knowledge, such as knowledge[15], prior knowledge[16], constraints[17], hints[18], etc. This paper uses engineering knowledge as a term. To integrate knowledge and data, neural networks and Gaussian processes have been adopted as models for building surrogates, and these form the basis for two main research streams of research for integrating knowledge and data.

This paper divides the research process of neural network fusion knowledge into three stages. The first stage is the exploratory stage. In this stage, researchers thought more about how to convert knowledge into data and integrate it into neural networks. In 1990, Abu-Mostafa proposed a method of training a neural network from a hint, which is represented by a set of data[18]. Sill further extended this work and proposed a method to incorporate monotonic information into neural networks[19]. In the second stage, researchers try to change the network structure so that the neural network can better integrate the corresponding types of knowledge. Based on the work of Sill et al., Daniels proposed a monotonic neural network structure, which can integrate monotonic knowledge faster[20]. Gupta proposed a neural network structure with shape constraints, which can integrate shape-constrained knowledge faster[21]. Towel et al. proposed a hybrid learning neural network structure that can simultaneously use different knowledge and data to train network parameters[15]. In the third stage, researchers begin to consider the uncertainties in experience and data. Researchers use a variety of methods to characterize the uncertainty in knowledge and data, such as fuzzy neural networks, Bayesian methods, and so on. Han et al. proposed a Data and Knowledge driven Fuzzy Neural Network (DK-FNN), which integrates regular knowledge into the training process of the neural network and reduces the dependence on data[22]. Liu Yang et al. proposed a Physical Information Neural Network based on Bayesian method (B-PINN). This method has high accuracy when the data noise is large[23].

Gaussian process is another way of integrating data and knowledge. The research of Gaussian process fusion knowledge can be divided into three stages as well. In the first stage, the researchers try to transform the knowledge into the first-order derivative information, and then integrate it into the Gaussian process. In 1993, Morris Studied how to integrate first-order derivative information into Gaussian process[24]. William applied Gaussian process to regression analysis and trained Gaussian process regression model with first-order derivative information[25]. In the second stage, researchers begin to integrate derivative information and data into Gaussian process as prior information. Solak combined derivative information and data together as prior information in Gaussian process model. This method significantly improves the efficiency and accuracy[26]. Subsequently, a study by Stephenson showed that the use of derivatives can reduce the demand for data in computer model analysis[27]. In the third stage, the object is no longer limited to the first-order derivative. The researchers expand the information of the first-order derivative into a monotonic relationship. The current research on Gaussian process fusion experience is derived from the third stage, including monotonic knowledge, shape knowledge (segmented monotonic), etc. Riihimäki realized the integration of monotonic information into the Gaussian process, which paved the way for subsequent research on integrating different knowledge into the Gaussian process[28]. Lenk and Choi proposed a method to estimate the shape limiting function using prior Gaussian process. The proposed model implements shape limiting by assuming that the derivative of the function is the square of the Gaussian process[29]. Wang and Welch used Gauss process to combine monotonicity constraints, and used for the objective function that can be decomposed into the sum of monotonic functions[30].

The existing work is summarized in Table 1, classified by the number of items of knowledge, the number of data points, and the applications. An item of knowledge represents a relationship between an input parameter and an output parameter.

Notably, there is a lack of a surrogate model construction algorithm for developing requirements for engineering problems in which both the diversity of available knowledge and the scarcity of data need to be considered. Therefore, to address this gap, we create an algorithm that can integrate multiple types or items of knowledge, which can be effectively used in the case of limited simulation data.

Unlike the existing methods, we adopt an evolutionary algorithm to integrate the simulation data and engineering knowledge into a neural network. Our previous work has demonstrated the capability of integrating data and knowledge with a very simple evolutionary algorithm [31]. Compared with our previous work, the coding method of neural network and the representation method of engineering experience are

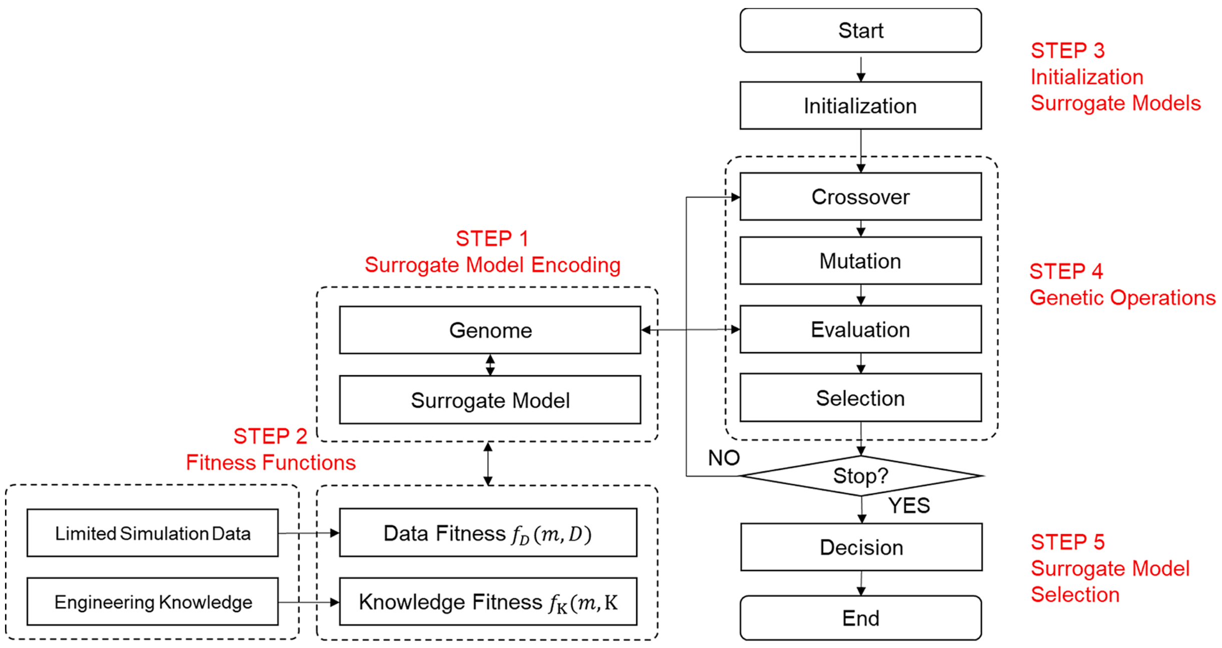


Fig. 1. The main steps of eDaKnow.

Table 2
The primitives for representing monotonicity engineering knowledge.

Id	Primitive	Explanation	Example
1	IF	The condition declarative primitive	**<u>IF</u>** x_1 INCREASEs
2	THEN	The result declarative primitive	IF x_1 INCREASEs **<u>THEN</u>** y_1 INCREASEs
3	WHEN	The additional condition declarative primitive	IF x_1 INCREASEs THEN y_1 INCREASEs **<u>WHEN</u>** x_2 is EQUAL to 1
4	INCREASE/ DECREASE	The primitives to define conditions and result	IF x_1 **<u>INCREASEs</u>** THEN y_1 **<u>DECREASEs</u>** WHEN x_2 is EQUAL to 1
5	APPROACH		IF x_1 INCREASEs THEN y_1 DECREASEs WHEN x_1 **<u>APPROACH</u>** to 1
6	EQUAL		IF x_1 INCREASEs THEN y_1 DECREASEs WHEN x_2 is **<u>EQUAL</u>** to 1
7	GREATER/ LOWER/ BETWEEN		IF x_1 INCREASEs THEN y_1 DECREASEs WHEN x_1 is **<u>GREATER</u>** than 1 AND **<u>LOWER</u>** than 5
8	AND/ OR	The logical operations of two conditions	IF x_1 INCREASES **<u>AND</u>** x_2 DECREASEs THEN y_1 DECREASEs

improved. We use the self-learning method of network structure, and there is no need to manually adjust the network structure. In this way, the optimal network structure can be found more quickly. In addition, we represent the knowledge in the form of gradient, which can optimize the fitness function and accelerate the training speed of the model. The following subsection details the technical foundations of the work developed here.

3. Method

Both limited simulation data and engineering knowledge contain critical information for constructing surrogate models [32]. Due to the heterogeneous representation of these sources, it is difficult to integrate them directly. Therefore, in this section a method is illustrated in which the two sources are converted into fitness functions and further integrated into a neural network using an evolutionary process[33]. This method is referred to as eDaKnow, which stands for building surrogate model by integrating limited simulation <u>D</u>ata and engineering <u>Know</u>ledge with <u>e</u>volutionary neural networks. Following this idea, building surrogate model is an optimization problem.

$$\begin{cases} \max f_D(m,D) \\ \max f_K(m,K) \end{cases} \tag{1}$$

where m indicates the surrogate model (in this case, a neural network); D represents the limited simulation data while K represents the engineering knowledge; $f_D(m,D)$ is the data fitness function, which is used to measure the degree of limited simulation data satisfied by the surrogate model; $f_K(m,K)$ is the knowledge fitness function, which is used to measure the degree of engineering knowledge satisfied by the surrogate model; The main steps of eDaKnow are illustrated in Fig. 1.

As shown in Fig. 1, the main process of eDaKnow is the same as is commonly used in evolutionary algorithms. To implement this process, five critical steps are involved.

STEP 1 *Surrogate Model Encoding*: Representing a neural network in way which is computable and amenable to evolution.

STEP 2 *Fitness Function*: Evaluating the surrogate model in terms of satisfying the limited simulation data and engineering knowledge.

Step 3 *Initialization Surrogate Models*: Initializing the first generation of the evolutionary algorithm.

Step 4 *Genetic Operations*: Configuring crossover, mutation, and selection operations for evolutionary algorithms.

Step 5 *Surrogate Model Selection*: Selecting the final surrogate model from the evolutionary results.

3.1. *Limited simulation data and monotonic engineering knowledge*

3.1.1. *Limited simulation data*

Limited simulation data commonly includes inputs and outputs: the inputs are design variables of products or processes, while the outputs represent the performance of the products or processes. In this paper, datasets with the following characteristics are regarded as limited simulation data.

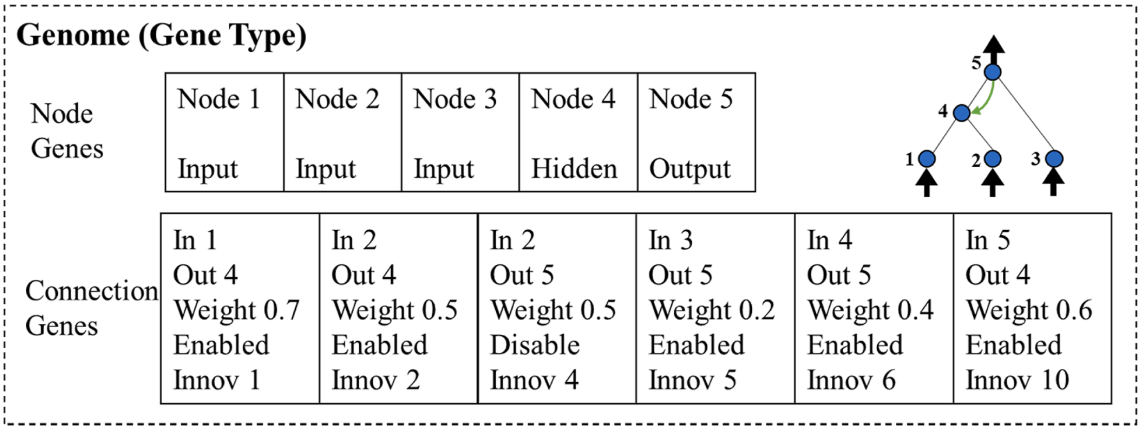

Genome (Gene Type)
Node Genes
Node 1 Input
Node 2 Input
Node 3 Input
Node 4 Hidden
Node 5 Output
Connection Genes
In 1 Out 4 Weight 0.7 Enabled Innov 1
In 2 Out 4 Weight 0.5 Enabled Innov 2
In 2 Out 5 Weight 0.5 Disable Innov 4
In 3 Out 5 Weight 0.2 Enabled Innov 5
In 4 Out 5 Weight 0.4 Enabled Innov 6
In 5 Out 4 Weight 0.6 Enabled Innov 10
5
4
1
2
3

Fig. 2. A genetic encoding diagram of a 5 node topology for building a surrogate model [36].

(1) The number of data points is small

A set of input and output is regarded as an item of data. We cannot say the number of data points is small absolutely, because this depends on the desired accuracy. For a d-dimension *Lipschitz-continuous* function, which satisfies equation (2), the number of data points must reach $(C/2\epsilon)^d$ to obtain an accuracy prediction error ϵ [34].

$$\|f(x_1) - f(x_2)\| \leq C\|x_1 - x_2\| \tag{2}$$

For example, if we have a 2-dimensional function with a C of 1, 100 data points are needed to reach prediction error of 0.05. Here, a dataset in which the number of data points is smaller than $(C/2\epsilon)^d$ is regarded as limited simulation data.

(2) The information is incomplete

A dataset must contain information about a mapping relationship among all inputs and outputs. If a dataset contains a substantial number of data records that cover the entire design space and satisfies equation (2), it can be used directly to create a surrogate model. In contrast, if a dataset contains data records that cover a portion of the design space or few data points, then it contains incomplete information about the mapping.

3.1.2. *Monotonic engineering knowledge*

Monotonic engineering knowledge is another source of information for surrogate model construction, and it is often available for many engineering problems [35]. To clearly define the term monotonic engineering knowledge, a structured representation is proposed. In Table 2 we list primitives for representing monotonic engineering knowledge.

In Table 2, "IF", "THEN" and "WHEN" are the most critical primitives, since they constitute the basic structure of monotonic engineering knowledge. The "INCREASE/ DECREASE", "APPROACH", "EQUAL" and "GREATER/ LOWER/BETWEEN" are primitives that define concrete conditions. The "AND/OR" is used to express a logical relationship among two or more conditions. Some examples using these primitives are also listed in Table 2.

3.2. *Surrogate model encoding (Step 1 in Fig. 1)*

This work borrows idea from Neuro Evolution of augmenting topologies (NEAT) [36] to encode the surrogate model. In the example in Fig. 2, a two-layer method is adopted for encoding surrogate models, one layer contains a list of node genes and another layer contains a list of connection genes. For each node in the first layer, its identification (ID) and type (Input, hidden or output) are assigned. For each connection in the second layer, its identification (ID), connecting nodes (input and output), weights (numerical values, between 0 and 1), and status (enabled or disabled) are assigned. The ID of each connection is indicated by the innovation number (Innov). Connections between different nodes have their own innovation number. For example, as shown on the left of Fig. 2, if another connection is regenerated between 3 and 4. If such a connection has been generated before, that Innov number will be used. If not, a new number will be created. In the top-right corner of Fig. 2 the topology of the neural network corresponding to the genome is given.

3.3. *Fitness function (Step 2 in Fig. 1)*

Because of the heterogeneous representations of limited simulation data and engineering knowledge, they are converted into fitness functions and then embedded into surrogate models using an evolutionary process. Therefore, defining fitness functions corresponding to limited simulation data and monotonic engineering knowledge respectively is critical for this work. The data fitness function is used to assess the surrogate model in terms of satisfying the limited simulation data while the knowledge fitness function is used to measure the surrogate model in terms of satisfying the engineering knowledge.

3.3.1. *Data fitness function*

For the data fitness function, the fitness value is measured by the data prediction error. If the data prediction error is high, then the data fitness value is low, and vice versa. The data prediction error is calculated using the Mean Squared Error (MSE) of the surrogate model prediction values. Compared with other accuracy metrics, MSE can accurately evaluate the accuracy of the model and is easy to calculate, which is important for time-consuming evolutionary algorithms. Therefore, we choose MSE to measure the accuracy of the surrogate model.

$$MSE(m) = \frac{1}{N}*\sum_{i=1}^{N}\left(y^i_{prediction} - y^i_{real}\right)^2 \tag{3}$$

$$f_D(m, D) = 1 - MSE(m) \tag{4}$$

where $MSE(m)$ represents the mean square error of the m-th individual; N is the number of data points; $y_{prediction}$ is the data prediction value; y_{real} is the real value of the limited simulation data; $f_D(m, D)$ is the data fitness function, which is used to measure the degree of limited simulation data satisfied by the surrogate model. The higher the $f_D(m, D)$, the higher the fitness of the individual.

3.3.2. *Knowledge fitness function*

For the knowledge fitness function, the fitness value is measured by knowledge pass rate (KPR). If the KPR is high, then the knowledge fitness value is high, and vice versa. In this paper, KPR is regarded as the knowledge fitness function. The underlying idea of calculating KPR is to use finite differences to represent the local gradient of the surrogate model at multiple points. When all local gradients are consistent with

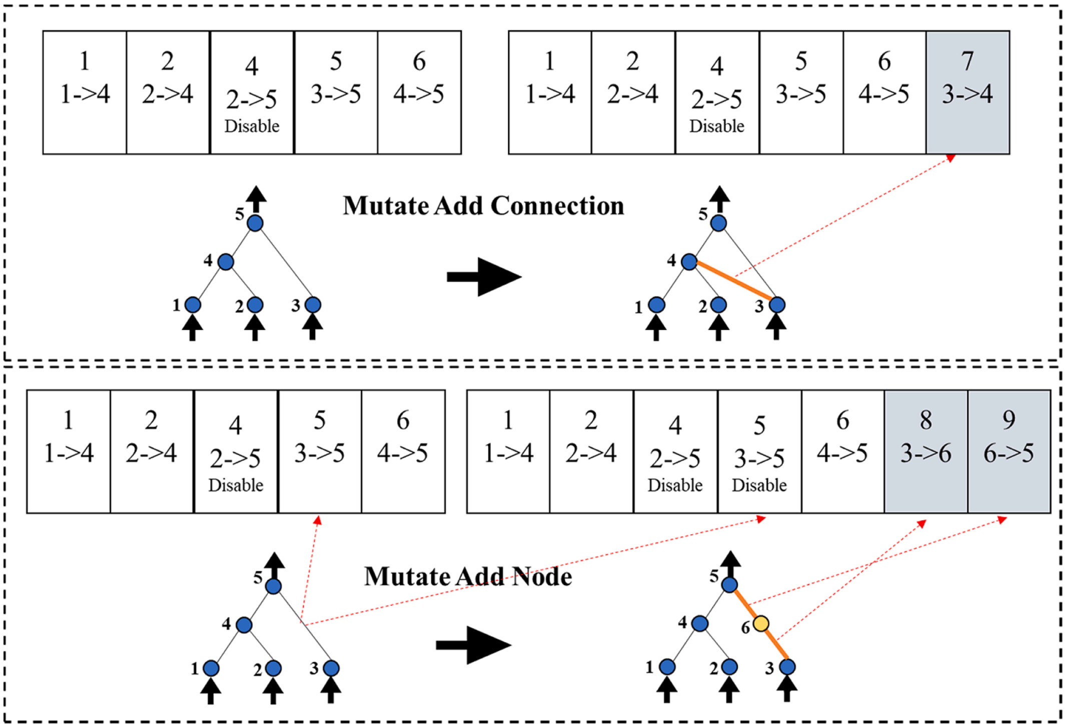


Fig. 3. The mutation operations used in eDaKnow [36].

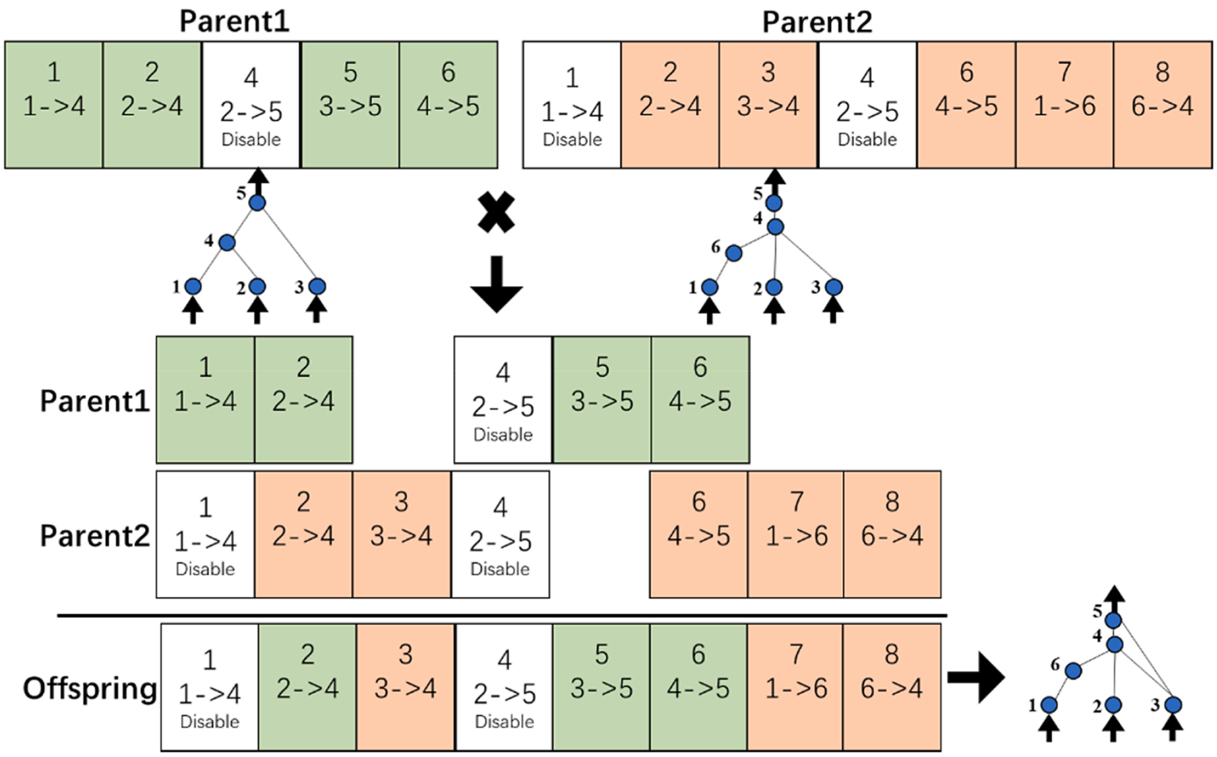


Fig. 4. The crossover operation used in eDaKnow [36].

monotonic engineering knowledge, the surrogate model passes and the KPR will be 1. When only a few of the local gradients are consistent with monotonic engineering knowledge, the KPR will be less than 1.

Assuming we have a monotonically increasing engineering knowledge $K(X)$ within a domain. The derivative $g(X_i, \nabla h)$ at each point in the domain can be obtained by finite difference based on equation (5), where∇h is the minimal quantity that tends to zero.

$$g(X_i, \nabla h) = \frac{K(X_i) - K(X_i + \nabla h)}{\nabla h} \tag{5}$$

We uniformly sample M points $X = [X_1, X_2, \cdots, X_M]$in the domain and when $M \rightarrow +\infty$, we will have $X_{i+1} - X_i \cong \nabla h$. If every $g(X_i, \nabla h)$ is positive, we can infer that $\nabla K(X) > 0$, where $\nabla K(X)$ represents the gradient

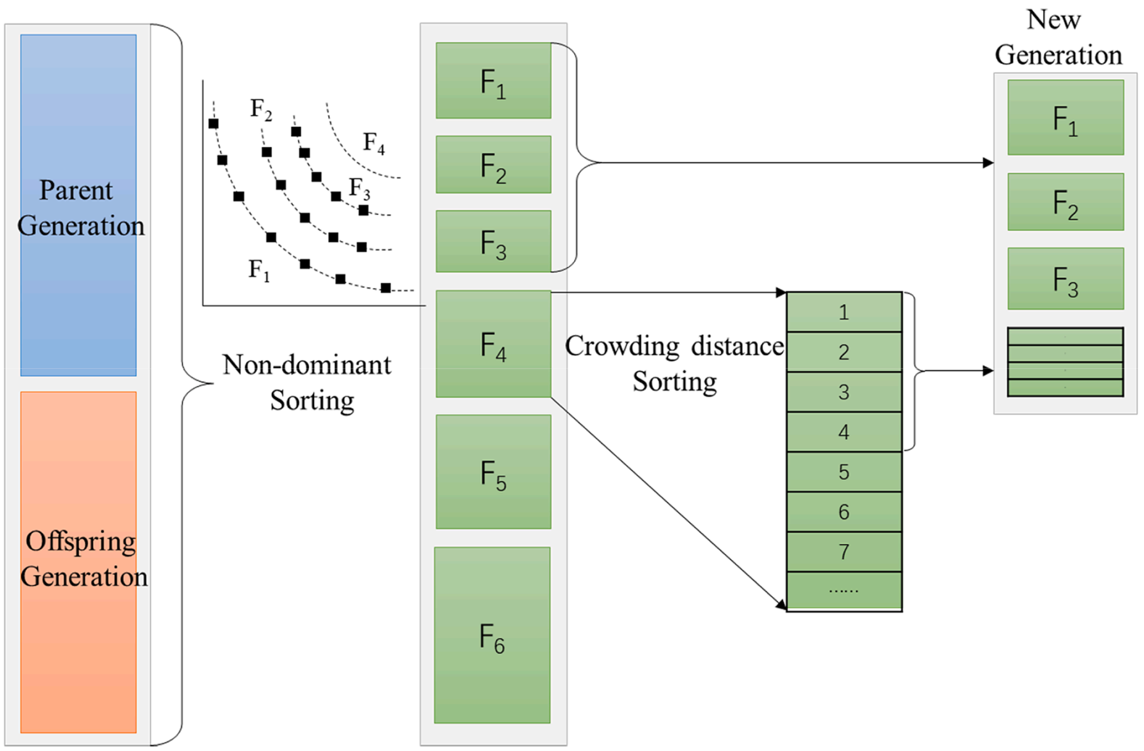


Fig. 5. The selection operation used in eDaKnow.

of the domain. It means the knowledge is monotonically increasing. Based on this idea the KPR is defined by equation (6) and (7). If KPR is high, then the fitness of knowledge and model is high.

$$f_K(m,K) = KPR = \frac{1}{M}\sum_{i=1}^{M} Z(g(X_i,\nabla h)) \tag{6}$$

$$Z(g(X_i,\nabla h)) = \begin{cases} 1 & g > 0 i = 1,2,\cdots M \\ 0 & g \le 0 i = 1,2,\cdots M \end{cases} \tag{7}$$

3.4. *Initialization surrogate models (Step 3 in Fig. 1)*

The first-generation surrogate model is initialized in two steps: the first is to create a vector that represents the input nodes and the output nodes, which are determined by the problem under consideration. Empirically, we find that using 3 hidden layer nodes yield the best results in this case. The accuracy of the experimental result is the same as those with 0 hidden layer nodes, and the convergence speed can be accelerated. Therefore, in this model, three hidden nodes are created and all neural networks in the initial generation use this as their hidden layer. Next a Boolean vector is randomly generated to indicate whether the connections are enabled, and a randomly generated real vector is created to indicate the weights of the connections.

3.5. *Genetic operations (Step 4 in Fig. 1)*

This section illustrates the technical detail of genetic operations for encoding the surrogate model, including mutant, crossover and selection. We build on the constructs of mutant and crossover are from NEAT while the idea of selection is the foundation for the Nondominated Sorting Genetic Algorithm-II for multi-objective optimization (NSGA-II) [14].

3.5.1. *Mutant operation*

A mutant operation can change either the weights (weight mutant) or the structure (structure mutant) of a surrogate model.

(1) Weight mutation

The vector of weights of connections are obtained from random numbers from a Gaussian distribution whose mean value is 0 and standard deviation is 1. The operation is controlled by the rate of mutation, which is determined by manual adjustment for multiple experiments. Here we seek to shorten the computation time to be less than 20 min under the condition that the accuracy is not reduced. The code runs on a ThinkPad T460 i5-6200 8G GTX 940 M.

(2) Structure mutation

The structure is mutated by two types of operations: connection addition and node addition, as shown in Fig. 3.

For connection addition, two nodes are connected directly if these nodes have not been connected previously. For node addition, the operation breaks a previously existing connection and sets as being disabled. Then, the two existing nodes are connected to the newly added node. When any new connections or nodes are added to the genome, they are assigned unique numbers which are shown in the top of the gene in Fig. 3. The operation is controlled by the rate of mutation, and by keeping this rate small, overfitting can be suppressed since it regulates the pace of expanding the capacity of the neural network.

3.5.2. *Crossover operation*

A crossover operation cannot be implemented directly since different genomes have different structures. Therefore, two parents are first aligned based on identical innovation numbers. In Fig. 4, the green blocks are Parent1, and the red blocks are from Parent2, and the white blocks have disabled connections. Then, the crossover operation is implemented using the following three rules: (1) If one connection is disabled in the parent, then the connection is also disabled in the offspring. (2) If the connection in both of the parents is enabled, then it is also enabled in the offspring. (3) If a connection is contained only in one parent, then the connection is also enabled in the offspring.

3.5.3. *Selection operation*

In our problem it is necessary to consider both data fitness and knowledge fitness; this is a multi-objective problem. This work builds on

Table 3
Details of benchmark functions and knowledge about them.

	Domain	Nonlinearity	Knowledge
B1	$-\infty \le x_1, x_2 \cdots x_n \le +\infty$	0.06	IFx_1INCREASEsORx_2INCREASEsTHENf_1INCREASEs
B2	$-10 \le x_1, x_2 \cdots x_n \le 10$	0.09	IFx_1APPROACHto0ANDx_2APPROACHto0THENf_1DECREASEs
B3	$-10 \le x_1, x_2 \cdots x_n \le 10$	0.15	IF x_1 APPROACH to 1 AND x_2 APPROACH to 3 THEN f_1 DECREASEs
B4	$-4.5 \le x_1, x_2 \cdots x_n \le 4.5$	0.4	IF x_1 APPROACH to 3 AND x_2 APPROACH to 0.5 THEN f_1 DECREASEs
B5	$-5 \le x_1, x_2 \cdots x_n \le 5$	0.42	IF x_2^2 INCREASEs THEN f_1 INCREASEs
B6	$-\infty \le x_1, x_2 \cdots x_n \le +\infty$	0.54	IF x_1 APPROACH to 1 THEN f_1 DECREASEs WHEN x_2 EQUAL to 1
B7	$-3 \le x_1, x_2 \cdots x_n \le 3$	0.76	IFx_1INCREASEsTHENf_2INCREASEs
B8	$1 \le x_1, x_2 \cdots x_6 \le 50 \le x_6, x_7 \cdots x_n \le 10$	0.79	IFx_1INCREASEsORx_2INCREASEsTHENf_2INCREASEs
B9	$0 \le x_1, x_2 \cdots x_n \le 1$	0.82	IFx_2INCREASEsORx_3INCREASEsTHENf_3INCREASEs
B10	$-2 \le x_1, x_2 \cdots x_n \le 2$	0.89	IF x_1 APPROACH to 0 THEN f_1 DECREASEs WHEN x_2 EQUAL to 1

Table 4
The tick marks indicate the experiments performed for various numbers of data points for different dimensions of the benchmark functions. The results are in the index.

Data Dimension	10	20	50	100	300	500
2	√	√	√			
5			√	√		
10				√	√	
15					√	√
20						√

the idea of selection from NSGA-II, which is a commonly used multi-objective algorithm, and it offers an efficient method for selection operations.

Selection is used to select a new generation based on the parent generation and offspring generation, as shown in Fig. 5. The parent and offspring generations are first combined and then divided into several Pareto fronts (F_1, F_2, F_3, F_4, F_5, and F_6). To select the new generation, these fronts can be added one by one until the number of genomes exceeds the generation size. The generation size is determined based on the size of the parent generation. The offspring generation is the same size as the parent generation and then the new generation is determined by selecting the best half of the offspring generation. To select the genomes from the last added front, similarly to NGSA-II, crowding distance sorting is used to retain the diversity of the genomes. In this way, a new generation is determined.

3.6. Surrogate model selection (Step 5 in Fig. 1)

When the evolution process is finished, a list of neural networks can be obtained in the non-dominant front. In this work we use a sub-set of data that is not used in the training process to evaluate all neural networks in the non-dominant front, and the following metric is used.

$$g_{error} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y^i_{prediction} - y^i_{real}}{y^i_{real}}\right| \tag{8}$$

where the g_{error} indicates the generalized prediction error of the surrogate model; N is the number of data points; $y_{prediction}$ is the data prediction value; y_{real} is the real value of the limited simulation data.

4. Experiments with eDaKnow

The proposed eDaKnow is tested with ten benchmark functions and two engineering problems. The proposed knowledge of benchmark functions is obtained by deriving test function. The knowledge of the first engineering problem is obtained by fitting polynomials and then deriving them. The knowledge of the second engineering problem is the subjective judgment obtained by asking the designer. This section details these functions, problems and knowledge.

4.1. Benchmark functions

The ten benchmark functions used are 1) Sphere function, 2) Matyas function, 3) Booth function, 4) Beale function, 5) Three-hump camel functions, 6) Rosen-Brock function, 7) Viennet function, 8) Osyczka and Kundu function, 9) Zitzler–Deb–Thiele's function, and 10) Goldstein-Price function [37]. In what follows, these benchmark functions are referred to as B1, B2, B3, B4, B5, B6, B7, B8, B9, and B10 for convenience. We get derivative relationships between different parameters by deriving benchmark functions, and take them as the engineering experience. The details of the ten benchmark functions are listed in Table 3, including the domain, range, nonlinearity and knowledge. The nonlinearity is a metric for measuring the functions' degree of complication, which will be used in Section 5.

The linearity of the problem can be expressed as the power value of the Pearson correlation coefficient when using first or second-order polynomial models[38]. Since the first-order polynomial model is the most suitable model for highly linear functions, we fit it to each function and calculate the nonlinearity value for them. 'Nonlinearity' is used as a continuous numerical description of the complexity of the problems [39]. The closer 'Nonlinearity' to 1, the more complex the problem is. The equation of Linearity and Nonlinearity can be expressed as follows:

$$Linearity = \left(\frac{cov(f_{\text{real}}, f_{\text{predict}})}{\sqrt{D(f_{\text{real}})}\sqrt{D(f_{\text{predict}})}}\right)^2 \tag{9}$$

$$Nonlinearity = 1 - \left(\frac{cov(f_{\text{real}}, f_{\text{predict}})}{\sqrt{D(f_{\text{real}})}\sqrt{D(f_{\text{predict}})}}\right)^2 \tag{10}$$

where f_{real} and f_{predict} denote the predicted and the real response respectively. $\text{cov}(f_{\text{real}}, f_{\text{predict}})$ represents the covariance of f_{real} and f_{predict}, while $\text{D}(f_{\text{real}})$ and $\text{D}(f_{\text{predict}})$ are the variance of f_{real} and f_{predict} respectively.

The benchmark functions listed in Table 3 have different dimensions. We use different numbers of data points when using the different dimensions to create the surrogate models. Table 4 shows the configurations of number of data points and the dimensions of benchmark functions. Experiments were performed as indicated.

4.2. Engineering problem 1 - Hot rod Rolling (HRR)

Steel is often used as a structural material in industry. The steel processing process is complex due to many the variables and constraints which must be considered. To meet the desired properties with physical experiments, many trials are required to obtain an optimal solution. This is an expensive and time-consuming process. Luckily, designers have accumulated some items of engineering knowledge through previous studies and a process can be obtained by simulation rather than by

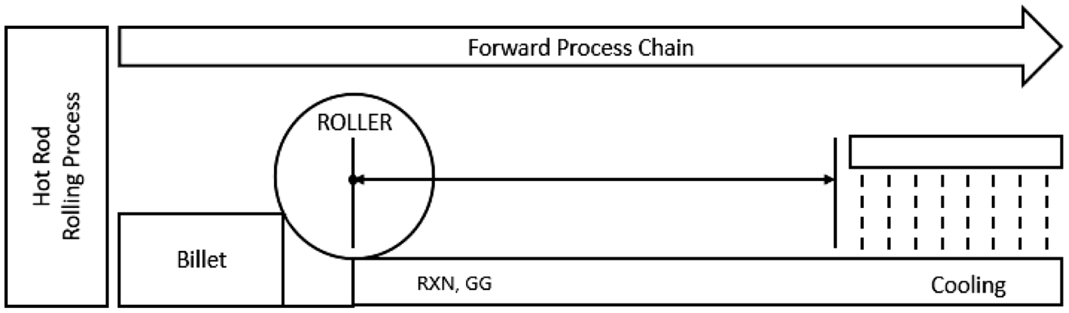


Fig. 6. Hot rod rolling (HRR) process chain [31].

Table 5
The variables of the Hot Rod Rolling Problem.

Type	Name	Range	Symbol
Design variable	The composition of Si	0.18 ~ 0.3%	x_1
	The composition of N	0.007 ~ 0.009%	x_2
	Ferrite grain size (D_α)	8 ~ 25μm	x_3
	The pearlite interlamellar spacing (S_0)	0.15 ~ 0.25μm	x_4
	The phase fractions of ferrite (X_f)	0.1 ~ 0.9	x_5
Performance	Tensile strength (TS)		y_1

Table 6
The engineering knowledge used in the Hot Rod Rolling problem.

ID	Knowledge
1	IF x_1 INCREASEs THENy_1 INCREASEs when x_1 is BETWEEN 0.23 and 0.3; IF x_1 INCREASEs THEN y_1 DECREASEs when x_1 is BETWEEN 0.18 and 0.23.
2	IF x_2 INCREASEs THEN y_1 INCREASEs
3	IF x_3 INCREASEs THEN y_1 INCREASEs when x_3 is BETWEEN 19 and 25; IF x_3 INCREASEs THEN y_1 DECREASEs when x_3 is BETWEEN 8 and 16.
4	IF x_4 INCREASEs THEN y_1 INCREASEs
5	IF x_5 INCREASEs THEN y_1 DECREASEs

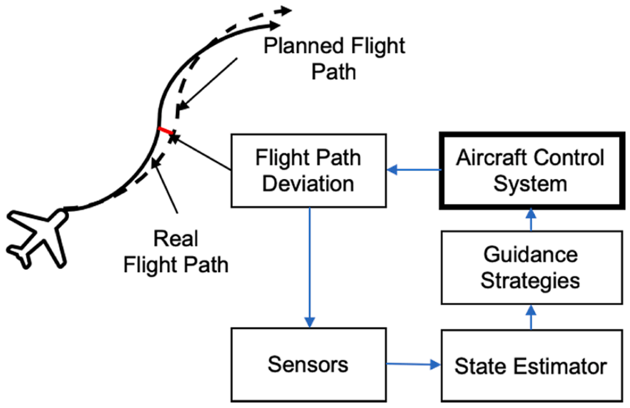


Fig. 7. The working principle of the ACS problem.

physical experiments. Fig. 6 shows the HRR process for creating a round rod from a square billet of steel, which forms the input material for automotive steel gear production [40]. In the HRR engineering problem, not all design variables are considered. We extract five key variables from all design variables as input variables.

Tensile strength (TS) is a key property of steel, which indicates the resistance to breaking under a tensile load. In this experiment, we wish to build a surrogate model to learn the mapping between TS and a group of variables, including the composition of C ([C]), the composition of M ([M]), ferrite grain size (D_α), the pearlite interlamellar spacing (S_0) and the phase fractions of ferrite (X_f). Table 5 shows the details of the variables. We fit the HRR engineering problem with polynomials, and then derive the polynomials to obtain the derivative relationship between different parameters, which is regarded as the engineering experience of HRR. Table 6 shows the related engineering knowledge. This is a complex multilevel problem which involves combining models for various steel properties, the formulation for the HRR problem can be found in our previous work [41]. To build a surrogate model by eDaKnow, we randomly sample a dataset with 100 data records. 75 data records are used to train the surrogate model while another 25 data records are used to test the surrogate model. Using equation (10), the nonlinearity of the HRR problem is 0.21.

Table 7
The input and output of the control system.

Type	Name	Variable
Input	First Order Time Constant	x_1
	Frequency	x_2
	Damping Coefficient	x_3
	Gain Adjustment Factor (Damp Loop)	x_4
	Gain Adjustment Factor (Acceleration Loop)	x_5
Output	Rise Time	y_1
	Transfer Coefficient	y_2
	Rudder Angle	y_3
	Amplitude Margin	y_4
	Phase Margin	y_5

4.3. *Engineering problem 2 - aircraft control system (ACS)*

As shown in Fig. 7, aircraft often have planned flight path, but there are always deviations between the planned flight path and actual flight path due to the uncertainty of the sensors and control system. The sensors installed in the aircraft receive signals and the state estimator can calculate the actual states of the aircraft. With the state information, the guidance strategy module generates guidance strategies and the controls system executes the strategy. This loop continues during the flight of the aircraft.

During the design stage of the control system, designers run computer simulations frequently to determine the parameters of the control systems, such as the first order time constant, frequency, damping coefficient and so on. The design variables and the performance output are shown in Table 7. In the ACS engineering problem, not all design variables are considered. We extract five key variables from all design variables as input variables. To build a surrogate model by eDaKnow, we randomly sample a dataset with 200 data records and 150 data records are used to train the surrogate model while another 50 data records are used to test the surrogate model. For the ACS engineering problem, we ask designers to get their judgments about the derivative relationship

Table 8
Engineering knowledge about the control system.

Id	Knowledge
1	IF x_1 INCREASEs THEN y_1 INCREASEs
2	IF x_2 INCREASEs THEN y_2 INCREASEs
3	IF x_3 INCREASEs THEN y_3 DECREASEs
4	IF x_4 INCREASEs THEN y_4 DECREASEs
5	IF x_5 INCREASEs THEN y_5 DECREASEs

Table 9
The parameter settings of eDaKnow used to build surrogate models.

Parameter	Value	Explanation
pop_size	100	Total number of neural networks in a generation
generation	1000	Total number of generations to run
num_hidden	3	Number of hidden layer nodes in the initial population
crossover_parent	0.7	The probability of crossover operations
mutate_add_conn	0.5	The probability of adding one connection
mutate_delete_conn	0.5	The probability of deleting one connection
mutate_add_node	0.2	The probability of adding one node
mutate_delete_node	0.2	The probability of deleting one node

Table 13
Empirical results for integrating monotonic knowledge in obtaining surrogate models for the benchmark functions.

	NE(%)	WE(%)
B1	4.62	1.48
B2	5.25	2.68
B3	5.61	2.46
B4	8.65	4.41
B5	15.68	9.01
B6	16.33	10.34
B7	13.49	8.16
B8	15.32	9.62
B9	17.53	11.52
B10	15.39	11.49
HRR	14.72	5.24
ACS	16.83	4.12

Table 10
The parameter settings of Kriging used to build surrogate models.

Parameter	Value	Explanation
poly	constant	Regression function type
Corr	['abs_exp','squar_exp','act_exp','matern52','matern32','gower']	Correlation function type
nugget	2.22e-14	a jitter for numerical stability
theta0	0.01	Initial hyperparameters
theta_bounds	[1e-06, 20.0]	bounds for hyperparameters
noise0	0.0	Initial noise hyperparameters
noise_bounds	[2.22e-14, 10ˆ10]	bounds for noise hyperparameters
n_start	10	number of optimizer runs (multistart method)

Table 11
The parameter settings of RBF used to build surrogate models.

Parameter	Value	Explanation
d0	1.0	basis function scaling parameter in exp(-dˆ2 / d0ˆ2)
poly_degree	−1	−1 means no global polynomial, 0 means constant, 1 means linear trend
reg	1e-10	Regularization coeff

Table 12
The parameter settings of ANN used to build surrogate models.

Parameter	Value	Explanation
activation_fun	Sigmoid	Activation function
layer_hidden	1	Number of hidden layers
num_hidden	5	Number of hidden layer nodes
Num	10,000	Number of training iterations

between parameters, and take them as engineering experience. Table 8 shows the related engineering knowledge. Using equation (10), the nonlinearity of ACS problem is 0.65.

4.4. *Empirical parameter setting*

In the algorithm comparison, we use four algorithms to compare with eDaKnow, which are Kriging, RBF, ANN and RSM. The RSM used in this paper refers to the second-order polynomial, which means that there is no need to set the parameters. The parameters of the eDaKnow, Kriging, RBF, ANN are shown in Tables 9–12. To reduce uncertainty of data collection and evolutionary process, each experiment is repeated 30 times. The method is implemented with Python code from open-source packages including the NSGA-II[1], NEAT[2] and SMT[3] packages.

5. Results and discussion

To verify the eDaKnow, two types of experiments are conducted. One is to verify whether the proposed eDaKnow can integrate monotonic knowledge, and the second is to verify whether eDaKnow is better than existing methods of surrogate modeling.

Table 14
Empirical results for using multiple items of knowledge.

	NE(%)	WE(%)		
		One (1)	Three (1/2/3)	Five
HRR	14.72	10.21	7.04	5.24
ACS	16.83	9.18	6.24	4.12

5.1. *Empirical results on integrating monotonic knowledge*

To verify whether eDaKnow can integrate monotonic knowledge, we build surrogate models with eDaKnow with knowledge, which is referred to as with engineering knowledge (WE). Another is to build surrogate model by eDaKnow without knowledge, these models are referred to as no engineering knowledge (NE). The empirical results for the benchmark functions are shown in Table 13, these values are the averaged prediction error of the 30 runs. Here the tables only show empirical results for the configuration of 2 dimensions and 50 data points. The empirical results are similar under different configurations. The results for the engineering problems are shown in Table 14.

[1] https://github.com/haris989/NSGA-II
[2] https://github.com/MorvanZhou/Evolutionary-Algorithm
[3] https://smt.readthedocs.io/en/latest/

Table 15
Prediction errors of different outputs with different amounts of engineering knowledge.

	NE	One (1)	Three (1/2/3)	Five
y_1	16.32	7.38 (54.78%)	5.86 (64.09%)	3.72 (77.21%)
y_2	17.01	10.57 (37.85%)	6.84 (59.79%)	4.69 (72.43%)
y_3	16.74	9.04 (46.00%)	6.19 (63.02%)	4.04 (75.87%)
y_4	16.67	8.23 (50.63%)	6.34 (61.97%)	3.88 (76.72%)
y_5	17.41	10.68 (38.66%)	5.97 (65.71%)	4.27(75.47%)

We see that the incorporation of engineering knowledge decreases the errors for all benchmark functions and engineering problems. This indicates that eDaKnow can be used to integrate monotonic knowledge and that integrating limited simulation data with monotonic knowledge is indeed helpful in decreasing the prediction errors of the surrogate models.

To investigate the influence of the number of items of knowledge, the empirical results when using the different numbers of items of knowledge are presented in Table 14. The "One (1)" means only the first item of monotonic knowledge in Table 8 is integrated, the "Three (1/2/3)" means the top three items of monotonic knowledge are integrated and the "Five" means all five items of monotonic knowledge are integrated.

Two important results are observed from this table. The first is that the adoption of the first item of engineering knowledge decreases the prediction error of the ACS from 16.83% to 9.18% by a rate of 45.45% and decreases the prediction error of HRR from 14.72% to 10.21% by a rate of 30.6%, which is an improvement. The second is that additional knowledge can further decrease the prediction error. However, this does not mean that the adoption of additional engineering knowledge will continuously decrease the prediction error at the same rate. As can be seen from Table 14, the pace of decreasing of the prediction errors becomes smaller when more items of engineering knowledge are integrated.

Unlike the benchmark functions and the HRR problem, the ACS problem has five outputs. One question can naturally arise: does the knowledge contribute to all outputs equally? To investigate this question, the prediction errors of all outputs when different knowledge is integrated are shown in Table 15. This table indicates that when a specific item of knowledge is integrated, not only the output that is directly related to the knowledge is improved, but also other outputs. For example, when the first item of knowledge is used, the prediction error of y_1 decreases from 16.32% to 7.38% with a rate of 54.78%. In addition, it also decreases the respective prediction errors of y_2, y_3, y_4 and y_5 from 17.01%, 16.74%, 16.67% and 17.41% to 10.57%, 9.04%, 8.23% and 10.68% by the rates of 37.86%, 46.00%, 50.63%, and 38.66%.

To investigate the evolutionary processes, the training errors in each generation are presented in Fig. 8, and the knowledge fitness values in each generation are presented in Fig. 9. We can see from these two figures that the training errors of WE are smaller than NE during the evolutionary process, and the knowledge fitness values can reach to 1 (100%) within 100 generations. The results further demonstrate that eDaKnow can integrate monotonic knowledge efficiently.

In Fig. 8, a-j, the training errors of each generation of the ten benchmark functions are presented. From this figure we can see clearly that when no knowledge is integrated, the training errors in all generations are large compared with those with integrated knowledge. Further, the use of additional knowledge is beneficial to the speed of convergence. In Fig. 8, i-j the training errors of each generation of the two engineering problems are presented. From this, here also, we can see that when additional items of knowledge are used, the training errors in all generations become smaller. For example, when three items of engineering knowledge are used, the training errors in almost all generations are smaller compared with those when only one item of engineering knowledge is used. Similarly, when five items of engineering knowledge are used, the training errors in all generations are smaller compared with those when three items of engineering knowledge are used. Further this result indicates that additional knowledge is more beneficial for the convergence speed.

In Fig. 9, the knowledge pass rates of each generation of the ten benchmark functions and two engineering problems are presented. From the three subplots, we find that in all experiments the pass rates reach the value of 1 (100%) within 100 generations. The more interesting finding is that the number of generations needed to make the pass rates of engineering knowledge converge is far less than those needed for

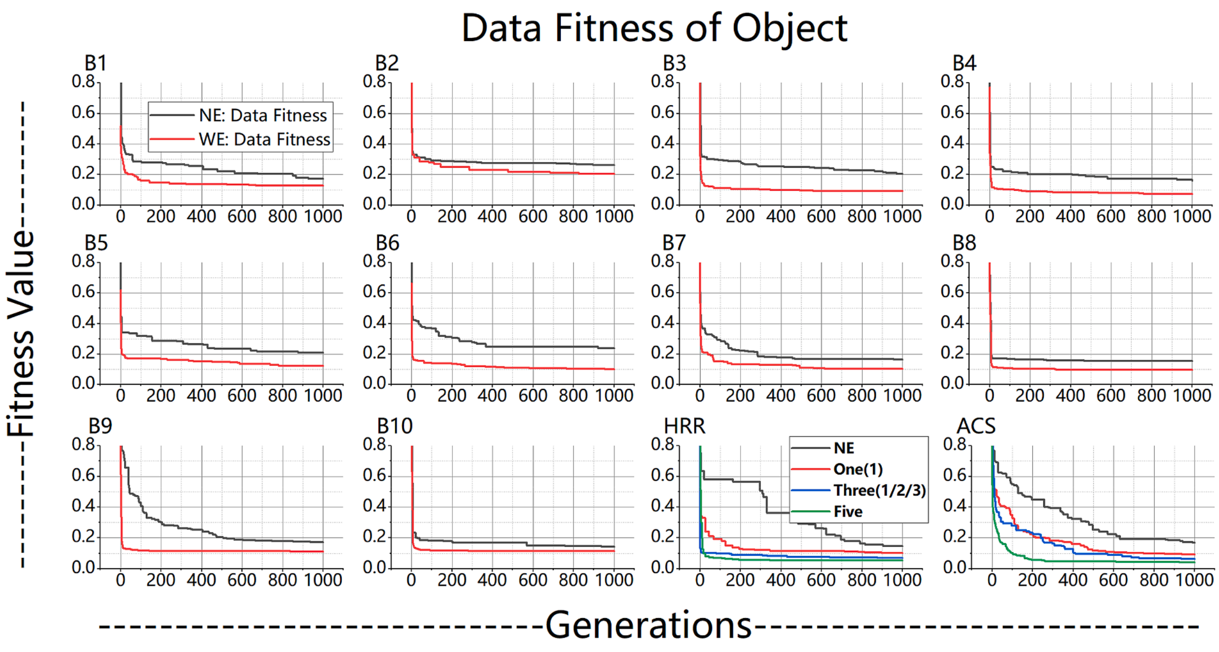
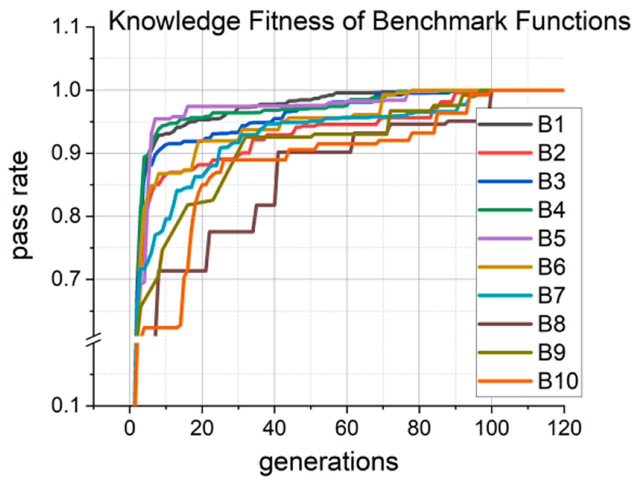
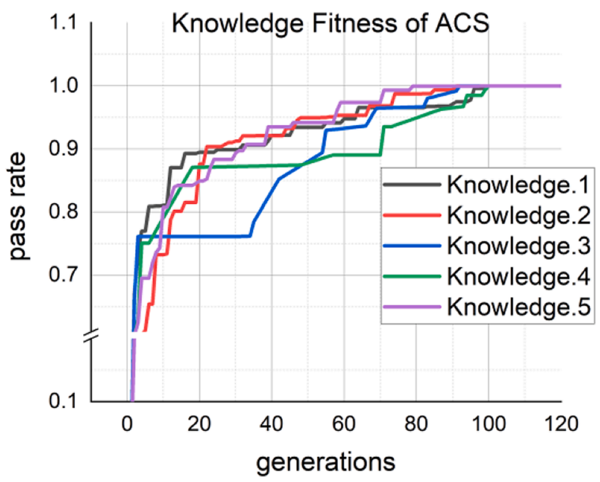


Fig. 8. The training error of benchmark functions and engineering problems through multiple iterations.





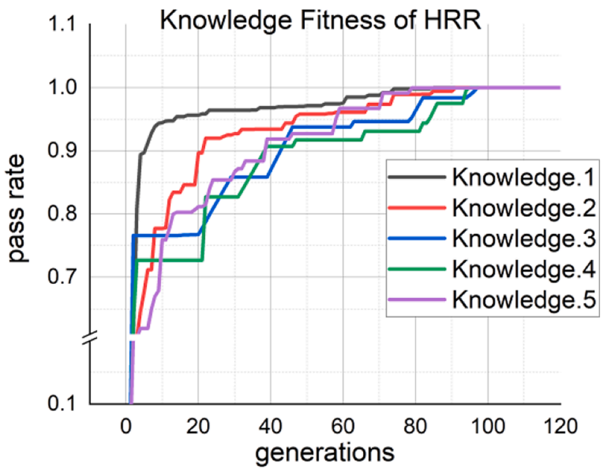


Fig. 9. The Knowledge Pass Rate (KPR) of benchmark functions and engineering problems.

Table 16
Empirical results of prediction errors on benchmark functions (the preferred method is underlined for each problem).

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	HRR	ACS
eDaknow	1.48	2.68	2.46	4.41	9.01	10.34	**8.16**	**9.62**	**11.52**	**11.49**	5.24	**4.12**
Kriging	0.45	0.75	0.92	7.29	**1.69**	**3.65**	10.47	15.37	26.82	46.8	4.27	8.37
RBF	0.32	0.81	**0.11**	**3.2**	6.37	28.66	14.28	19.25	42.69	206.58	**2.63**	153.28
RSM	**0.02**	**0.13**	0.79	9.38	25.21	166.6	186.2	238.2	347.1	799.1	7.17	327.09
ANN	26.8	18.5	1.35	4.01	35.78	14.63	19.74	26.18	29.17	14.91	11.89	13.75
Nonlinearity	0.06	0.09	0.15	0.4	0.42	0.54	0.76	0.79	0.82	0.89	0.21	0.65

Table 17
Empirical results for benchmark functions in different dimensions and numbers of data points. In the Configuration column, the first number is the number of dimensions and the number following the hyphen is the number of groups of data points.

Configuration	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10
2–50	Rs	Rs	Rb	Rb	K	K	eDK	eDK	eDK	eDK
5–50	Rs	Rb	Rb	Rb	K	K	eDK	eDK	eDK	eDK
5–100	Rs	Rs	Rb	Rb	K	K	eDK	eDK	eDK	eDK
10–100	Rs	Rs	Rs	Rb	K	K	eDK	eDK	eDK	eDK
10–300	Rs	Rs	Rb	Rb	K	K	eDK	eDK	eDK	eDK
15–300	Rs	Rs	Rb	Rb	K	K	eDK	eDK	eDK	eDK
15–500	Rs	Rs	Rs	Rb	K	K	eDK	eDK	eDK	eDK
20–500	Rs	Rs	Rs	Rb	K	K	eDK	eDK	eDK	eDK

*Rs = RSM; Rb = RBF; K = Kriging; eDK = eDaKnow; "n-m" means n dimension and m data points.

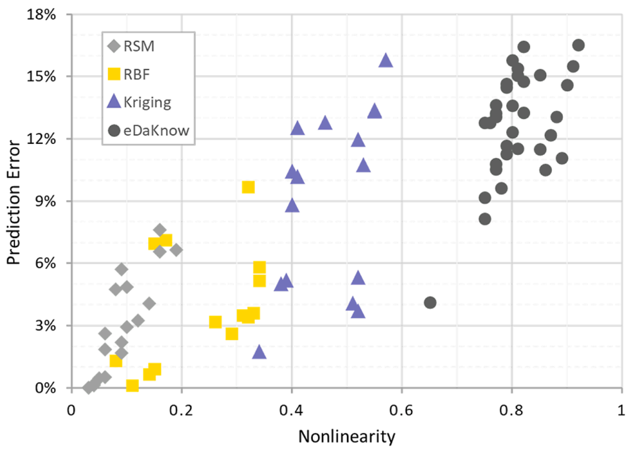


Fig. 10. The surrogate model with the least prediction error and its error value under different nonlinearity.

the training errors. For example, from subplot A of Fig. 9, when no engineering knowledge is used the training errors obtain convergence after about 600 generations, when five items of engineering knowledge are used, the training errors still need more than 200 generations to obtain convergence. However, all the knowledge pass rates converge within 100 generations. Thus, the integration of engineering knowledge does not increase the required number of iterations.

5.2. *Empirical results on compared with existing methods*

To verify whether eDaKnow is better than existing methods for building surrogate models, it is compared with benchmark functions of eDaKnow, Kriging, Radial Basis Function (RBF), Response Surface Model (RSM) and Artificial Neural Network (ANN). The results are presented in Table 16. From this table, we can see that the eDaKnow has obvious advantages over other methods when the nonlinearity of the problem is high (B7, B8, B9, B10 and ACS).

To investigate the influence of the dimensions of the benchmark functions and the number of data points, ten experiments for different configurations of dimensions (from 2 to 20) and the number of data points (from 50 to 500) are conducted, as shown in Table 4. The full empirical results are presented in the Appendix, and in Table 17 the best method for various configurations is underlined.

In Fig. 10, the surrogate models with the least prediction error and its error value under different nonlinearity are presented. Obviously, RSM has a very strong advantage in terms of low nonlinearity. In this case, its prediction error is the smallest among the five surrogate models, and it can achieve high prediction accuracy. With the increase of nonlinearity, the performance of RBF gradually outperforms other algorithms. In the nonlinearity interval of 0.1 to 0.35, RBF performs best. Similarly, Kriging performs best in the nonlinearity interval of 0.35 to 0.6. In the nonlinearity interval of 0.6 to 1, eDaKnow performs best. Due to the lack of data, ANN is not the best performer anywhere.

With the increase of nonlinearity, the prediction error of the best-performing surrogate model in each interval is gradually increasing. Nonlinearity greatly affects the prediction accuracy of the surrogate model.

We also performed a numerical analysis on the prediction error values of five surrogate models in all experiments, calculated the mean and variance. The results are shown in Fig. 11.

It can be seen from Fig. 11 that compared with other methods, the performance of eDaKnow is the most stable. Facing the problems of different condition settings, the variance of eDaKnow is the smallest among the five types of surrogate models. At the same time, eDaKnow's average prediction error is also the smallest, which means that it performs well in most problems. Considering that the nonlinearity of engineering problems is often very high and the constraints are complex, we believe that edaware is more suitable for engineering problem modeling.

6. Conclusion

Surrogate models are being used in simulation-based design of engineering systems. However, when only limited simulation data are available, building high performance surrogate models is difficult. In this paper, a method for the integration of monotonic engineering knowledge and limited simulation data to build surrogate models is proposed. Thus, the limited simulation data is complemented by engineering knowledge to build more accurate surrogate models. We find that (1) the proposed method can be used efficiently to combine limited simulation data and monotonic engineering knowledge; (2) the inclusion of monotonic engineering knowledge can significantly decrease the prediction errors of the surrogate model; (3) additional engineering

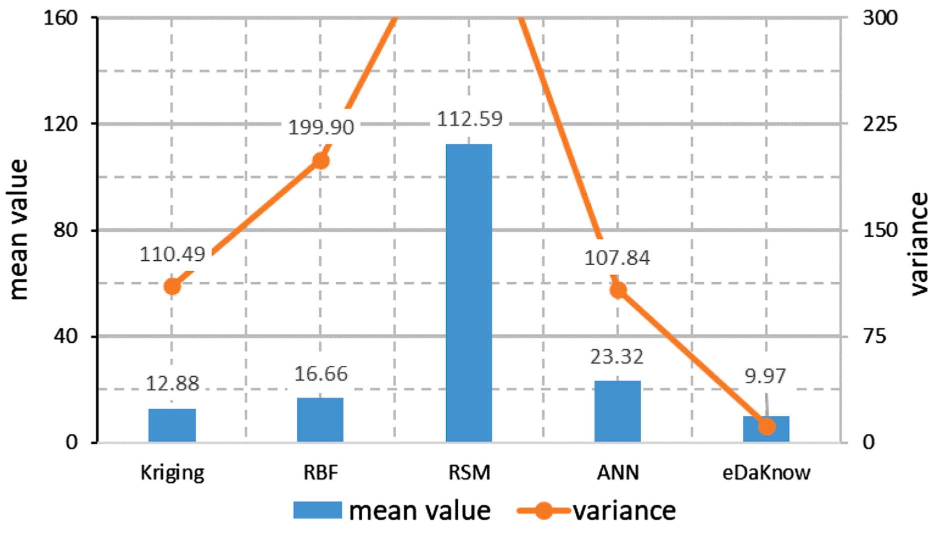


Fig. 11. The mean and variance of the surrogate model prediction error.

knowledge yields higher accuracy; and (4) Adding engineering knowledge can improve the convergence rate.

The way we integrate the limited simulation data and monotonic engineering knowledge is different from the existing methods that tune the parameters of the model directly. Instead, we integrate the two types of information into a surrogate model with a data fitness function and a knowledge fitness function. Through the self-learning method of network structure, we avoid manual adjustment and improve the training speed of the algorithm. This method is demonstrated based on benchmark functions and engineering examples but may be extended to use various types of engineering problems.

But there are still problems to be solved. The first is the complexity of the algorithm. Integrating knowledge will increase the complexity of the algorithm. The more knowledge involved, the higher the complexity of the algorithm. This will bring about two problems, which will greatly extend the training time of the algorithm and make it difficult for the algorithm to converge. The second is the diversity of knowledge. This paper focuses on monotonic knowledge, but there are many types of knowledge, how to use different types of knowledge is a problem to be solved. The third is the uncertainty of knowledge. The engineering knowledge obtained from designers is subjective, which cannot guarantee that the knowledge is absolutely correct. This will affect the prediction accuracy of the surrogate model. How to solve the knowledge uncertainty is a problem to be solved.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper is an outcome of the International Systems Realization Partnership between the Institute for Industrial Engineering @ The Beijing Institute of Technology, The Systems Realization Laboratory @ The University of Oklahoma and the Design Engineering Laboratory @ Purdue. Janet K. Allen gratefully acknowledges the John and Mary Moore Chair at the University of Oklahoma. The author thanks Farrokh Mistree for his suggestions on improving the initial manuscript. The author thanks Anand Balu Nellippallil for the use of the hot rod rolling case. The authors thank Reza Alizadeh Kordabad of the Systems Realization Laboratory for his suggestions on improving the initial manuscript.

Appendix

As discussed in Section 5.2, the empirical results for studies on the benchmark functions are presented in Table A.1. The data in the table are prediction errors except nonlinearity. (See Table A2, Table A3, Table A4, Table A5, Table A6, Table A7, Table A8).

Table A1
Empirical results for benchmark functions in **2**D and **50** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	4.62	5.25	5.61	8.65	15.68	16.33	13.49	15.32	17.53	15.39
WE (%)	1.48	2.68	2.46	4.41	9.01	10.34	**8.16**	**9.62**	**11.52**	**11.49**
Kriging (%)	0.45	0.75	0.92	7.29	**1.69**	**3.65**	10.47	15.37	26.82	46.8
RBF (%)	0.32	0.81	**0.11**	**3.2**	6.37	28.66	14.28	19.25	42.69	206.58
RSM (%)	**0.02**	**0.13**	0.79	9.38	25.21	166.55	186.23	238.19	347.06	799.1
ANN (%)	26.83	18.5	1.35	4.01	35.78	14.63	19.74	26.18	29.17	14.91
Nonlinearity	0.03	0.04	0.11	0.26	0.34	0.52	0.75	0.78	0.81	0.85

*RSM = response surface model; RBF = radial basis function; ANN = artificial neural network.
*Best results have been marked in bold.

Table A2
Empirical results for benchmark functions in **5**D and **50** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	6.77	6.76	7.15	10.31	16.03	17.98	16.43	16.88	19.31	17.68
WE (%)	3.48	3.26	4.82	5.69	9.99	13.59	**10.53**	**11.28**	**13.59**	**12.17**
Kriging (%)	1.39	1.63	1.68	8.11	**5.13**	**5.27**	11.95	16.70	37.68	48.66
RBF (%)	1.48	**1.30**	**0.91**	**3.49**	7.80	29.12	15.02	20.89	43.83	217.89
RSM (%)	**0.45**	1.46	1.01	10.25	25.37	171.09	187.24	239.51	348.74	810.46
ANN (%)	28.56	18.62	2.40	5.80	37.44	14.68	19.93	26.76	30.26	15.12
Nonlinearity	0.05	0.08	0.15	0.31	0.39	0.52	0.77	0.79	0.8	0.87

Table A3
Empirical results for benchmark functions in **5**D and **100** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	6.16	7.09	8.14	9.74	15.86	16.73	15.64	16.05	17.82	16.72
WE (%)	3.11	3.10	4.13	4.99	8.06	12.59	**9.18**	**10.78**	**11.67**	**10.50**
Kriging (%)	0.24	1.49	1.37	7.93	**4.95**	**4.01**	10.77	15.40	37.62	47.22
RBF (%)	0.22	0.55	**0.65**	**2.63**	7.68	28.43	14.61	20.29	43.00	201.78
RSM (%)	**0.17**	**0.53**	0.77	9.10	24.75	165.01	186.89	238.18	347.39	809.10
ANN (%)	27.07	18.60	1.29	5.77	36.31	13.67	19.76	26.14	29.48	15.57
Nonlinearity	0.04	0.06	0.14	0.29	0.38	0.51	0.75	0.77	0.79	0.86

Table A4
Empirical results for benchmark functions in **10**D and **100** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	8.92	9.42	10.35	12.60	17.52	17.27	16.11	17.97	19.86	19.67
WE (%)	5.20	6.19	8.34	10.08	10.90	14.24	**13.24**	**14.47**	**15.06**	**11.07**
Kriging (%)	4.77	8.21	9.66	11.45	**8.75**	**10.69**	13.93	17.38	43.92	51.15
RBF (%)	3.07	6.48	8.21	**9.69**	13.45	28.91	21.32	28.62	46.62	222.60
RSM (%)	**2.63**	**5.70**	**7.60**	17.09	31.86	189.50	192.77	240.60	351.74	834.47
ANN (%)	28.98	26.06	8.41	13.78	41.19	14.45	27.01	30.85	30.09	21.12
Nonlinearity	0.06	0.09	0.16	0.32	0.4	0.53	0.77	0.79	0.81	0.89

Table A5
Empirical results for benchmark functions in **10**D and **300** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	8.94	9.83	12.23	13.78	16.64	16.14	13.71	15.64	18.21	16.08
WE (%)	6.86	7.83	8.96	10.88	12.71	13.56	**12.77**	**12.32**	**13.27**	**13.05**
Kriging (%)	6.32	9.35	7.02	11.43	**10.37**	**11.91**	13.17	16.60	43.14	50.42
RBF (%)	2.01	5.45	**6.95**	**3.42**	13.29	28.27	20.76	27.96	46.61	210.96
RSM (%)	**1.85**	**4.74**	7.08	16.15	31.16	180.73	191.97	240.01	350.80	813.56
ANN (%)	28.92	25.84	7.56	12.86	40.61	13.90	26.13	29.85	30.07	20.20
Nonlinearity	0.06	0.08	0.15	0.32	0.4	0.52	0.75	0.8	0.82	0.88

Table A6
Empirical results for benchmark functions in **15**D and **300** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	9.51	10.25	12.53	14.15	19.44	17.49	16.39	18.35	20.91	21.68
WE (%)	7.66	8.04	9.59	11.65	15.10	14.58	**13.07**	**15.80**	**16.43**	**15.50**
Kriging (%)	8.25	10.74	8.11	12.21	**12.47**	**13.33**	15.01	21.24	44.20	52.64
RBF (%)	3.15	6.06	**7.14**	**5.82**	13.56	32.73	24.50	30.18	48.51	219.53
RSM (%)	**2.18**	**4.85**	7.65	16.85	34.01	187.63	193.13	244.74	351.51	854.27
ANN (%)	32.91	27.12	10.76	12.96	43.62	17.14	26.79	33.31	30.63	22.98
Nonlinearity	0.09	0.1	0.17	0.34	0.41	0.55	0.77	0.8	0.82	0.91

Table A7
Empirical results for benchmark functions in **15**D and **500** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	8.23	8.74	9.83	9.92	18.94	16.80	16.23	17.92	20.16	19.45
WE (%)	5.56	6.91	7.60	7.95	14.73	14.20	**12.77**	**14.64**	**14.77**	**14.59**
Kriging (%)	5.77	8.51	7.09	10.38	**10.11**	**13.26**	13.16	18.90	43.09	50.02
RBF (%)	2.65	6.03	9.46	**3.62**	12.22	31.28	23.84	27.97	45.75	213.68
RSM (%)	**1.67**	**2.94**	**6.57**	14.39	31.27	180.08	190.94	243.94	351.10	821.45
ANN (%)	30.33	26.32	10.01	9.96	42.22	15.43	24.15	31.43	30.10	22.63
Nonlinearity	0.09	0.1	0.16	0.33	0.41	0.55	0.76	0.79	0.82	0.9

Table A8
Empirical results for benchmark functions in **20**D and **500** groups of data points.

	Benchmark Functions									
	B1	B2	B3	B4	B7	B5	B8	B9	B10	B6
NE (%)	8.65	9.35	10.89	11.40	19.63	18.69	16.46	19.16	20.94	20.75
WE (%)	6.74	7.05	7.82	8.74	15.87	16.04	**13.64**	**15.40**	**15.07**	**16.54**
Kriging (%)	8.69	12.40	6.69	11.80	**12.72**	**15.73**	17.97	22.39	44.00	51.65
RBF (%)	4.95	10.16	12.34	**5.17**	13.33	34.66	26.53	30.37	49.21	217.56
RSM (%)	**3.25**	**4.07**	**6.65**	17.87	33.15	189.24	192.70	247.05	351.50	840.35
ANN (%)	33.16	30.96	14.24	10.52	46.89	18.88	27.34	31.55	34.85	27.36
Nonlinearity	0.12	0.14	0.19	0.34	0.46	0.57	0.77	0.81	0.85	0.92

References

[1] G.L. Rocca, Knowledge Based Engineering: Between AI and CAD. Review of a Language Based Technology to Support Engineering Design, Adv. Eng. Inf. 26 (2) (2012) 159–179.

[2] M.-A. Sicilia, Ontology of Systems and Software Engineering, Adv. Eng. Inf. 21 (2) (2007) 117–118.

[3] İ. Karen, N. Kaya, F. Öztürk, Intelligent Die Design Optimization Using Enhanced Differential Evolution and Response Surface Methodology, J. Intell. Manuf. 26 (5) (2015) 1027–1038.

[4] R. Mallipeddi, M. Lee, An Evolving Surrogate Model-Based Differential Evolution Algorithm, Appl. Soft Comput. 34 (2015) 770–787.

[5] W. Gong, A. Zhou, Z. Cai, A Multioperator Search Strategy Based on Cheap Surrogate Models for Evolutionary Optimization, IEEE Trans. Evol. Comput. 19 (5) (2015) 746–758.

[6] M. Giselle Fernández-Godino, S. Balachandar, R.T. Haftka, On the Use of Symmetries in Building Surrogate Models, J. Mech. Des. 141 (6) (2019), 061402.

[7] T.W. Simpson, T.M. Mauery, J.J. Korte, F. Mistree, Kriging Models for Global Approximation in Simulation-Based Multidisciplinary Design Optimization, AIAA journal 39 (12) (2001) 2233–2241.

[8] Z. Hu, S. Mahadevan, A Single-Loop Kriging Surrogate Modeling for Time-Dependent Reliability Analysis, J. Mech. Des. 138 (6) (2016), 061406.

[9] K. Shimoyama, K. Sato, S. Jeong, S. Obayashi, Updating Kriging Surrogate Models Based on the Hypervolume Indicator in Multi-Objective Optimization, J. Mech. Des. 135 (9) (2013), 094503.

[10] L. Zhao, K.K. Choi, I. Lee, D. Gorsich, Conservative Surrogate Model Using Weighted Kriging Variance for Sampling-Based RBDO, J. Mech. Des. 135 (9) (2013), 091003.

[11] R.G. Regis, C.A. Shoemaker, Combining Radial Basis Function Surrogates and Dynamic Coordinate Search in High-Dimensional Expensive Black-Box Optimization, Eng. Optim. 45 (5) (2013) 529–555.

[12] Z. Qian, C.C. Seepersad, V.R. Joseph, J.K. Allen, C.J. Wu, Building Surrogate Models Based on Detailed and Approximate Simulations, J. Mech. Des. 128 (4) (2006) 668–677.

[13] Yu, T., 2007, “Incorporating Prior Domain Knowledge into Inductive Machine Learning : Its Implementation in Contemporary Capital Markets,” Thesis, University Of Technology Sydney.

[14] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, in: Proceedings of the 2000 Parallel Problem Solving from Nature, 2000, pp. 849–858.

[15] G. Towell, J. Shavlik, Knowledge-Based Artificial Neural Networks, Artif. Intell. 70 (1994) 119–165.

[16] 1994, “Support Theory: A Nonextensional Representation of Subjective Probability: Amos Tversky, Department of Psychlogy, Stanford University, Stanford, CA 94305-2130 USA,” Mathematical Social Sciences, 28(3), pp. 224–225.

[17] Lang, B., 2005, “Monotonic Multi-Layer Perceptron Networks as Universal Approximators,” Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005, Springer, Berlin, Heidelberg, pp. 31–37.

[18] Y.S. Abu-Mostafa, Learning from Hints in Neural Networks, Journal of Complexity 6 (2) (1990) 192–198.

[19] J. Sill, Monotonic Networks, in: M.I. Jordan, M.J. Kearns, S.A. Solla (Eds.), Advances in Neural Information Processing Systems 10 (NIPS 1997), MIT Press, Cambridge, MA, 1998, pp. 661–667.

[20] H. Daniels, M. Velikova, Monotone and Partially Monotone Neural Networks, IEEE Trans. Neural Networks 21 (6) (2010) 906–917.

[21] M.R. Gupta, D. Bahri, A. Cotter, K. Canini, in: Diminishing Returns Shape Constraints for Interpretability and Regularization, Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 6835–6845.

[22] Han, H.-G., Wu, X., Liu, Z., and Qiao, J., 2019, “Data-Knowledge-Based Fuzzy Neural Network for Nonlinear System Identification,” IEEE Transactions on Fuzzy Systems, PP, pp. 1–1.

[23] L. Yang, X. Meng, G. Karniadakis, B-PINNs: Bayesian Physics-Informed Neural Networks for Forward and Inverse PDE Problems with Noisy Data, J. Comput. Phys. 425 (2021), 109913.

[24] M.D. Morris, T.J. Mitchell, D. Ylvisaker, Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction, Technometrics 35 (3) (1993) 243–255.

[25] C. Williams, C. Rasmussen, Gaussian Processes for Regression, Advances in Neural Information Processing Systems 8 (1996) 8.

[26] Solak, E., Murray-Smith, R., Leithead, W. E., Leith, D., and Rasmussen, C. E., 2003, “Derivative Observations in Gaussian Process Models of Dynamic Systems,” Appear Adv Neural Inf Process Syst, 16.

[27] Stephenson, G., 2010, “Using Derivative Information in the Statistical Analysis of Computer Models.”.

[28] J. Riihimäki, A. Vehtari, Gaussian Processes with Monotonicity Information, Journal of Machine Learning Research - Proceedings Track 9 (2010) 645–652.

[29] “Peter J. Lenk and Taeryon Choi (2017). BAYESIAN ANALYSIS OF SHAPE-RESTRICTED FUNCTIONS USING GAUSSIAN PROCESS PRIORS. Vol 27 No. 1, 43-69.”.

[30] Wang, W., and Welch, W., 2018, “Bayesian Optimization Using Monotonicity Information and Its Application in Machine Learning Hyperparameter.”.

[31] Hao, J., and Ye, W., 2018, “Evolutionary Neural Network-Based Method for Constructing Surrogate Model with Small Scattered Dataset and Monotonicity Experience,” Proceedings of the 2018 Soft Computing & Machine Intelligence, Nairobi, Kenya.

[32] J. Hao, M. Zhou, G. Wang, L. Jia, Y. Yan, Design Optimization by Integrating Limited Simulation Data and Shape Engineering Knowledge with Bayesian Optimization (BO-DK4DO), J Intell Manuf (2020) 1–19.

[33] E. Elbeltagi, T. Hegazy, D. Grierson, Comparison among Five Evolutionary-Based Optimization Algorithms, Adv. Eng. Inf. 19 (1) (2005) 43–53.

[34] Brochu, E., Cora, V. M., and de Freitas, N., 2010, “A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning,” ArXiv, p. 49.

[35] A. Sharif Ullah, Modeling and Simulation of Complex Manufacturing Phenomena Using Sensor Signals from the Perspective of Industry 4.0, Adv. Eng. Inf. 39 (2019) 1–13.

[36] K.O. Stanley, R. Miikkulainen, Efficient Evolution of Neural Network Topologies, in: Proceedings of the 2002 Congress on Evolutionary Computation, 2002, pp. 1757–1762.

[37] “Test Functions for Optimization - Wikipedia” [Online]. Available: https://en.wikipedia.org/wiki/Test_functions_for_optimization. [Accessed: 24-Sep-2020].

[38] R. Jin, W. Chen, T. Simpson, Comparative Studies of Metamodeling Techniques Under Multiple Modeling Criteria, Struct. Multidiscip. Optim. 23 (2001) 1–13.

[39] Liangyue Jia, Reza Alizadeh, Jia Hao, Guoxin Wang, Farrokh Mistree, A Rule-Based Method for Automated Surrogate Model Selection, Adv. Eng. Inf. 45 (2020), 101123.

[40] A.B. Nellippallil, V. Rangaraj, B.P. Gautham, A.K. Singh, J.K. Allen, F. Mistree, A Goal-Oriented, Inverse Decision-Based Design Method to Achieve the Vertical and Horizontal Integration of Models in a Hot Rod Rolling Process Chain, American Society of Mechanical Engineers Digital Collection. (2017).

[41] Reza Alizadeh, Liangyue Jia, Anand Balu Nellippallil, Guoxin Wang, and Farrokh Mistree, 2019, “Ensemble of Surrogates and Cross-Validation for Rapid and Accurate Predictions Using Small Data Sets,” Artificial intelligence for engineering design analysis and manufacturing, pp. 1–18.