Expert Systems With Applications 109 (2018) 25–34

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A multilevel block building algorithm for fast modeling generalized separable systems

Chen Chen[a,b], Changtong Luo[a,*], Zonglin Jiang[a,b]

[a] State Key Laboratory of High Temperature Gas Dynamics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

A R T I C L E I N F O

Article history:
Received 27 June 2017
Revised 1 May 2018
Accepted 18 May 2018
Available online 19 May 2018

Keywords:
Symbolic regression
Genetic programming
Generalized separability
Multilevel block building

A B S T R A C T

Symbolic regression is an important application area of genetic programming (GP), aimed at finding an optimal mathematical model that can describe and predict a given system based on observed input-response data. However, GP convergence speed towards the target model can be prohibitively slow for large-scale problems containing many variables. With the development of artificial intelligence, convergence speed has become a bottleneck for practical applications. In this paper, based on observations of real-world engineering equations, generalized separability is defined to handle repeated variables that appear more than once in the target model. To identify the structure of a function with a possible generalized separability feature, a multilevel block building (MBB) algorithm is proposed in which the target model is decomposed into several blocks and then into minimal blocks and factors. The minimal factors are relatively easy to determine for most conventional GP or other non-evolutionary algorithms. The efficiency of the proposed MBB has been tested by comparing it with Eureqa, a state-of-the-art symbolic regression tool. Test results indicate MBB is more effective and efficient; it can recover all investigated cases quickly and reliably. MBB is thus a promising algorithm for modeling engineering systems with separability features.

1. Introduction

Symbolic regression seeks to identify an optimal mathematical model that can describe and predict a given system based on observed input-response data. Unlike conventional regression methods that require a preset explicit expression of the target model, symbolic regression can extract an appropriate function (model) from a space of all possible expressions $\mathcal{S}$ defined by a set of given binary operations (e.g., $+$, $-$, $\times$, $\div$) and mathematical functions (e.g., $\sin$, $\cos$, $\exp$, $\ln$), which can be described as follows:

$$f^* = \arg\min_{f\in\mathcal{S}} \sum_i \left\| f\left(\boldsymbol{x}^{(i)}\right) - y_i \right\|, \tag{1}$$

where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are sample data, f is the target model, and f^* is the regression model.

Symbolic regression has been widely applied in many engineering sectors, such as industrial data analysis (e.g., Li, Zhang, Bailey, Hoagg, & Martin, 2017; Luo, Hu, Zhang, & Jiang, 2015), circuits analysis and design (e.g., Ceperic, Bako, & Baric, 2014; Shokouhifar & Jalali, 2015; Zarifi, Satvati, & Baradaran-nia, 2015), signal processing (e.g., Volaric, Sucic, & Stankovic, 2017; Yang, Wang, & Soh, 2005), empirical modeling (e.g., Gusel & Brezocnik, 2011; Mehr & Nourani, 2017), and system identification (e.g., Guo & Li, 2012; Wong, Yip, & Li, 2008). Genetic programming (GP) (Koza, 1992) is a classical method of symbolic regression. Theoretically, GP can obtain an optimal solution provided that the computation time is sufficiently long. However, the computational cost of GP for large-scale problems with many input variables is still quite high. This situation can be further exacerbated by increasing problem size (i.e., the number of involved independent variables) and complexity of the target function.

GP has been refined in several ways. Some variants focus on the coding plan. For example, grammatical evolution (GE) (O'Neill & Ryan, 2001) suggests using a variable-length binary string as the genotype of a target function, and parse-matrix evolution (PME) (Luo & Zhang, 2012) suggests using a parse-matrix with integer entries to retain more information from the parse tree. Some other variants have tested different evolutionary strategies, such as clone selection programming (Gan, Chow, & Chau, 2009) and artificial bee colony programming (Karaboga, Ozturk, Karaboga, & Gorkemli, 2012). GP variants can simplify the coding process and provide al-

* Corresponding author.
E-mail addresses: chenchen@imech.ac.cn (C. Chen), luo@imech.ac.cn (C. Luo), zljiang@imech.ac.cn (Z. Jiang).

https://doi.org/10.1016/j.eswa.2018.05.021
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

ternative evolutionary strategies; however, these methods do little to improve convergence speed when solving large-scale problems.

In the past decades, increasing attention has been paid to reducing search space. For instance, McConaghy (2011) presented the first non-evolutionary algorithm, fast function eXtraction (FFX), which confined its search space to a generalized linear space. However, computational efficiency is gained by sacrificing the generality of the solution. More recently, Worm (2016) proposed a deterministic machine learning algorithm, prioritized grammar enumeration (PGE), in his thesis. PGE merges isomorphic chromosome presentations (equations) into a canonical form, yet a debate is ongoing regarding how simplification affects the solving process (Kinzett, Johnston, & Zhang, 2009; Kinzett, Zhang, & Johnston, 2008; McRee, Software, & Park, 2010).

More recently, a favorable feature in the symbolic regression method, separability, has been addressed based on the fact that the target model is separable in many scientific or engineering problems (Luo, Chen, & Jiang, 2017). A divide-and-conquer (D&C) method for GP has also been presented to make use of the separability feature. The solving process is accelerated by dividing the target function into a number of sub-functions. Compared to conventional GP, the D&C method can reduce computational effort (complexity) by orders of magnitude. Chen, Luo, and Jiang (2018) recently proposed an improved version of D&C, block building programming (BBP), in which the target function is partitioned into blocks and factors so it can further reduce the complexity of sub-functions.

However, the separability defined in Luo et al. (2017) and Chen et al. (2018) is limited in that it does not allow for recurrence of the same variable in different sub-functions; it would otherwise be considered non-separable. As a result, the sub-function size could still be large in many practical applications, which will be demonstrated in the following sections. This drawback motivates us to broaden the prospective applications of D&C and BBP in this work.

First, a generalized separability is defined to allow for recurrence of the same variable in different sub-functions. More specifically, the variables involved are classified into two types: repeated variables and non-repeated variables. The structure of the target function and the type of variables (repeated or non-repeated) are identified by a new proposed algorithm, multilevel block building (MBB), in which the blocks could be further decomposed into a higher level of blocks and factors until they are confirmed to be minimal blocks and factors. Therefore, the sub-functions (i.e., minimal factors) may have smaller sizes and be more easily identified. The minimal blocks and factors are then assembled together properly to form the target function. The block building process is similar to that of BBP.

In short, the new algorithm is an improved version of BBP (Chen et al., 2018) with more general application potential. The efficiency of the proposed MBB has been compared with the results of Eureqa, a state-of-the-art symbolic regression tool. Numerical results show that the proposed algorithm is more effective and can recover all investigated cases quickly and reliably.

The rest of this paper is organized as follows. Section 2 analyzes different types of separability in practical engineering. Section 3 is devoted to establishing the mathematical model of the GS system. In Sections 4 and 5, we propose an MBB algorithm and illustrate it using a case study. Section 6 presents numerical results and discussions for the proposed algorithm. The paper concludes with Section 8, which provides remarks on future work.

2. Observation of separability types

Recall that the separability introduced in Luo et al. (2017) can be described as follows.

Definition 2.1 (Separability)**.** A scalar function $f(X)$ with n continuous variables $X = \{x_i : i = 1, 2, \cdots, n\}$ ($f : \mathbb{R}^n \mapsto \mathbb{R}$, $X \subset \Omega \in \mathbb{R}^n$, where Ω is a closed bounded convex set, such that $\Omega = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$), is said to be separable if and only if it can be written as

$$f(X) = c_0 \otimes_1 c_1 \varphi_1(X_1) \otimes_2 c_2 \varphi_2(X_2) \otimes_3 \cdots \otimes_m c_m \varphi_m(X_m), \tag{2}$$

where the variable set X_i is a proper subset of X, such that $X_i \subset X$ with $\bigcup_{i=1}^m X_i = X$, $\bigcap_{i=1}^m X_i = \emptyset$, and the cardinal number of X_i is denoted by $\mathrm{card}(X_i) = n_i$, for $\sum_{i=1}^m n_i = n$ and $i = 1, 2, \cdots, m$. Sub-function φ_i is a scalar function where $\varphi_i : \mathbb{R}^{n_i} \mapsto \mathbb{R}$. The binary operator $\otimes_i$ could be plus (+) and times ($\times$).

This separability suits the separability type of many engineering models as demonstrated below.

Example 2.1. When developing a rocket engine, it is crucial to model the internal flow of high-speed compressible gas through the nozzle. The closed-form expression for the mass flow through a choked nozzle (Anderson, 2006) is

$$\dot{m} = \frac{p_0 A^*}{\sqrt{T_0}} \sqrt{\frac{\gamma}{R}\left(\frac{2}{\gamma+1}\right)^{(\gamma+1)/(\gamma-1)}}. \tag{3}$$

In Eq. (3), the five independent variables (p_0, T_0, A^*, R, and γ) are all separable. The equation can be called a multiplicatively separable function, expressed as follows:

$$\begin{aligned} \dot{m} &= f(p_0, A^*, T_0, R, \gamma) \\ &= \varphi_1(p_0) \times \varphi_2(A^*) \times \varphi_3(T_0) \times \varphi_4(R) \times \varphi_5(\gamma). \end{aligned} \tag{4}$$

The target function in this example is multiplicatively separable. The sub-functions are connected to the binary operator times ($\times$). Similarly, the target function might be additively separable, where the sub-functions are combined with the binary operator plus (+). These classes of separable functions are common in engineering. In practice, the binary operator between two sub-functions could be plus (+) or times ($\times$), which can be illustrated as follows.

Example 2.2. In aircraft design, the lift coefficient of a whole aircraft (Zhang, 2002) can be expressed as

$$C_L = C_{L\alpha}(\alpha - \alpha_0) + C_{L\delta_e}\delta_e \frac{S_{\mathrm{HT}}}{S_{\mathrm{ref}}}, \tag{5}$$

where variables $C_{L\alpha}$, $C_{L\delta_e}$, δ_e, S_{HT}, and S_{ref} are separable. Variables α and α_0 are not separable, but their combination (α, α_0) is. Hence, Eq. (5) can be expressed as

$$\begin{aligned} C_L &= f\left(C_{L\alpha}, \alpha, \alpha_0, C_{L\delta_e}, \delta_e, S_{\mathrm{HT}}, S_{\mathrm{ref}}\right) \\ &= \varphi_1(C_{L\alpha}) \times \varphi_2(\alpha, \alpha_0) + \varphi_3\left(C_{L\delta_e}\right) \times \varphi_4(\delta_e) \\ &\quad \times \varphi_5(S_{\mathrm{HT}}) \times \varphi_6(S_{\mathrm{ref}}). \end{aligned} \tag{6}$$

However, for many other engineering models, the separability type defined in Definition 2.1 is not broad enough; refer to the subsequent example.

Example 2.3. The flow past a circular cylinder is a classic problem in fluid dynamics. A valid stream function for the inviscid, incompressible flow over a circular cylinder of radius R (Anderson, 2011) is

$$\psi = (V_\infty r \sin\theta)\left(1 - \frac{R^2}{r^2}\right) + \frac{\Gamma}{2\pi}\ln\frac{r}{R}, \tag{7}$$

which can be re-expressed as

$$\begin{aligned} \psi &= f(V_\infty, \sin\theta, R, r, \Gamma) \\ &= \varphi_1(V_\infty) \times \varphi_2(\sin\theta) \times \boxed{\varphi_3(r, R)} + \varphi_4(\Gamma) \times \boxed{\varphi_5(r, R)}. \end{aligned} \tag{8}$$

Note that the target model in Example 2.3 is inconsistent with the separability type defined in Definition 2.1 because some variables (e.g., variables V_∞, θ, and Γ of Eq. (7)) appear only once in a given target model, whereas other variables (e.g., variables r and R appear more than once in Eq. (7). In other words, the recurrence of the same variable in different sub-functions has occurred. According to Definition 2.1, the target function here is not separable – and this is not a rare case. It is common for some parameters (e.g., one or two) to couple widely with other variables in real-world problems.

The inconsistency renders standard D&C and BBP (Chen et al., 2018) inapplicable for this kind of problem; however, this finding is unexpected. This discrepancy motivates us to define a more general model of separability to make these problems *separable* so as to reduce the computational complexity. The widely coupled variables (r and R in this case) are considered special variables, called repeated variables, and handled using specific techniques. Details are described in the following section.

3. Generalization of separability

As can be seen from Definition 2.1, each variable appears only once in the model function. However, as mentioned above, some variables might appear twice or more in practical applications. Thus, the standard D&C and BBP methods lost their basis of working mechanism and cannot be used to model such systems. In this section, to let the symbolic regression algorithm take more advantage of *separability*, variables are distinguished as **repeated** variables and **non-repeated** variables, and a more general concept of separability is defined as follows.

Definition 3.1 (Generalized separability)**.** A scalar function $f(X)$ with n continuous variables on a closed convex set Ω is said to be generalized separable (GS) if it can be rewritten as

$$\begin{aligned} f(X) &= f\left(X^{\mathrm{R}}, X^{\mathrm{NR}}\right) = c_0 + \sum_{i=1}^{m} c_i \varphi_i\left(X_i^{\mathrm{R}}, X_i^{\mathrm{NR}}\right) \\ &= c_0 + \sum_{i=1}^{m} c_i \tilde{\omega}_i\left(X_i^{\mathrm{R}}\right) \tilde{\psi}_i\left(X_i^{\mathrm{NR}}\right) \\ &= c_0 + \sum_{i=1}^{m} c_i \prod_{j=1}^{p_i} \omega_{i,j}\left(X_{i,j}^{\mathrm{R}}\right) \prod_{k=1}^{q_i} \psi_{i,k}\left(X_{i,k}^{\mathrm{NR}}\right). \end{aligned} \tag{9}$$

In Definition 3.1, superscript R denotes repeated variables, and NR denotes non-repeated variables. The set of variables X is partitioned into complementary sets X^{R} and X^{NR}. The set of variables X^{R}, X_i^{R}, X^{NR} and X_i^{NR} are further partitioned into respective non-overlapping subsets X_i^{R}, X_{ij}^{R}, X_i^{NR} and X_{ij}^{NR} (see Chen, Luo, & Jiang, 2017 for more details).

The intermediate functions $\varphi_i(\cdot)$ is called the ith minimal block of $f(X)$. The terminal sub-functions $\omega_{i,\,j}(\cdot)$ and $\psi_{i,\,k}(\cdot)$ are called the jth and kth factors of the repeated variables and non-repeated variables in the ith minimal block $\varphi_i(\cdot)$, respectively. The constants c_0, c_1, $\cdots$, c_m are coefficients.

Note that the standard separable function (see Definition 2.1) does not involve repeated variables. So it is a special case of this definition when the number of repeated variables is 0, or $q_i = 0$ in Eq. (9). The function structure in Definition 3.1 can describe the separability types of all typical examples discussed in Section 2.

The block, minimal block, and factor are different levels of sub-functions. The blocks are connected with the binary operator plus (+), while factors are connected with the binary operator times ($\times$). The minimal block is the block in which all sub-functions (factors) are connected with times ($\times$). The minimal factors can be considered as the *minimum elements* or *terminal sub-functions* of the target function.

Next, an example is given to illustrate these concepts.

Example 3.1. Columns 4–6 in Table 2 show the repeated variables, the number of minimal blocks, and the factors of 10 cases given in Appendix A, respectively. All minimal blocks in these cases are boxed.

4. Multilevel block building

The function structure of a given system with standard separability is detected by BiCT (Luo et al., 2017), a statistical method in which the target function can be divided into a number of additively or multiplicatively separable sub-functions. However, due to the presence of repeated variables, the GS function $f(X)$ is no longer separable in terms of standard BiCT; that is, the standard BiCT method cannot be used directly. It is necessary to carry out a deeper probe to determine the function structure. To identify the function structure of a given system with possible generalized separability (see Eq. (9) based on provided input-response data, three steps must occur:

(1) Distinguish repeated variables and non-repeated variables;
(2) Determine the proper number of minimal blocks and factors;
(3) Determine the membership of each variable (in which block and factor).

To fulfill these missions, a multilevel block search method is designed. It is an improved version of BiCT as described below.

4.1. Minimal block detection

The multilevel block search is designed based on an important fact: the number of blocks in terms of standard separability can increase if the repeated variables are fixed. Suppose the model function $f(X)$ of a GS system could initially be written as m_0 additively separable blocks ($m_0 \geq 1$). The test variable set X^t is a subset of the repeated-variable set X^{R} of $f(X)$, namely $X^t \subseteq X^{\mathrm{R}}$, if the following two conditions are satisfied:

(1) When the variables belonging to X^t are all fixed, the number of additively separable blocks of m will increase; that is, $m > m_0$.
(2) When the variables belonging to $\tilde{X}^t$ (in which $\tilde{X}^t \neq X^t$ is the element of the power set $\mathscr{P}X^t$ of X^t) are all fixed, the number of additively separable blocks of m will remain unchanged; that is, $m = m_0$.

In fact, based on the meaning of a repeated variable, if the repeated variables of a certain block are all fixed, then the non-repeated variables of this block become additively separable. Hence, m will increase. On the other hand, if a set $\tilde{X}^t$ belongs to $\mathscr{P}X^t$, then $f(X)$ can be rewritten as more than m_0 additively separable sub-functions. The difference set $\mathscr{P}X^t \backslash \tilde{X}^t$ must be a non-repeated-variable set.

Suppose that the test variable sets $X_{(i)}^t \neq \emptyset$ for $i = 1, 2, \cdots, k$, satisfy the two conditions for a given GS function $f(X)$. The repeated-variable set of $f(X)$ could be determined as $X^{\mathrm{R}} = \bigcup_{i=1}^{k} X_{(i)}^t$. This could be easily obtained if we let $X^t = X^{\mathrm{R}}$ in the above statement.

Based on these facts, the minimal block detection technique can be succinctly described by Algorithm 1:

4.2. Factor detection

Once the repeated-variable set X^{R} and non-repeated-variable set X_i^{NR} ($i = 1, 2, \cdots, m$) have been obtained by the minimal block

Algorithm 1 Minimal block detection.

```
1: Input: Target model $f : \mathbb{R}^n \mapsto \mathbb{R}$; Variable set $X = \{x_1, \cdots, x_n\}$; Sample data $\mathbf{S} \in \mathbb{R}^{N \times n}$
2: Let all variables be randomly sampled
3: Detect the additively separable blocks by BiCT
4: Update non-repeated variable sets of each block
5: Carry out the multi-level block search as follows:
6: for $i = 1 : n$ do
7:     Create $\binom{n}{i}$ test variable sets $X^t_{(j)} (j = 1, 2, \cdots, \binom{n}{i})$, which consist of all possible combinations of the set $X$ taken $i$ variables at a time
8:     for $j = 1 : \binom{n}{i}$ do
9:         Keep the test variables belong to $X^t_{(j)}$ fixed; Re-generate $\mathbf{S}'$; Call BiCT
10:        if More blocks are detected by BiCT then
11:            Add all $x^t \in X^t_{(j)}$ to $X^{\mathrm{R}}$
12:            Update the non-repeated variable sets of each block
13: Output: Repeated variable set $X^{\mathrm{R}}$, and non-repeated variable sets of each minimal block $X^{\mathrm{NR}}_i$, for $i = 1, 2, \cdots, m$
```

detection algorithm, the next step is to determine the repeated-variable set of each block, namely X^{R}_i, and the separability of each block, namely $X^{\mathrm{R}}_{i,j}$ and $X^{\mathrm{NR}}_{i,k}$, for $j = 1, 2, \cdots, p_i$, $k = 1, 2, \cdots, q_i$. For non-repeated variables, note that in Eq. (9), factors $\psi_{i,k}\left(X^{\mathrm{NR}}_{i,k}\right)$ are multiplicatively separable. The separability of the variables of $X^{\mathrm{NR}}_{i,k}$ in the ith block of Eq. (9) can be easily derived via standard BiCT method when all variables in X^{R} and $X^{\mathrm{NR}}_{\xi} (\xi \neq i)$ are fixed.

To determine the repeated variables in each block as well as their separability, two test functions, $\tilde{f}_1$ and $\tilde{f}_2$ are formulated for BiCT:

$$f(X)|_{\tilde{x}^t \to x_\alpha, x^{\mathrm{NR}} \to x_\beta, \tilde{x}^{\mathrm{NR}} \to x_\alpha} \triangleq \tilde{f}_1\left(X^t\right) \tag{10}$$

and

$$f(X)|_{x^t \to x_\alpha, x^{\mathrm{NR}} \to x_\beta, \tilde{x}^{\mathrm{NR}} \to x_\alpha} \triangleq \tilde{f}_2\left(\tilde{X}^t\right), \tag{11}$$

where $x^t_1, x^t_2, \cdots$ are test variables belonging to the test variable set X^t, such that $x^t \in X^t \subset X^{\mathrm{R}}$. Variables $\tilde{x}^t_1, \tilde{x}^t_2, \cdots$ belong to $\tilde{X}^t$, where $\tilde{X}^t$ is the complementary set of the test variable set X^t in the repeated-variable set X^{R}, where $\tilde{x}^t \in \tilde{X}^t = \complement_{X^{\mathrm{R}}} X^t$. $x^{\mathrm{NR}}_1, x^{\mathrm{NR}}_2, \cdots$ are the non-repeated variables of the ith block, such that $x^{\mathrm{NR}} \subset X^{\mathrm{R}}_i$. $\tilde{x}^{\mathrm{NR}}_1, \tilde{x}^{\mathrm{NR}}_2, \cdots$ are the non-repeated variables of all the blocks except the ith block, such that $\tilde{x}^{\mathrm{NR}} \subset \complement_{X^{\mathrm{NR}}} X^{\mathrm{NR}}_i$. x_α and x_β are two certain points that remain fixed in BiCT.

To eliminate the effects of other blocks when determining the separability of the repeated variables in the ith block, four groups involving eight sampling iterations should be evaluated using Eqs. (10) and (11), respectively. The first sampling group is evaluated as follows: set x_α to the point x_G, and set x_β to x_A and x_B, respectively. Let variables $\bar{x}_\xi$ be randomly sampled. Then, two function-value vectors, $\boldsymbol{y}^A_1$ and $\boldsymbol{y}^B_1$, are obtained. Let $\boldsymbol{y}_1 = \boldsymbol{y}^A_1 - \boldsymbol{y}^B_1$. The other three sampling groups are simply interpreted as follows:

$$\boldsymbol{y}_2 = \boldsymbol{y}^C_2 - \boldsymbol{y}^D_2, \quad \boldsymbol{y}_3 = \boldsymbol{y}^A_3 - \boldsymbol{y}^B_3, \quad \text{and} \quad \boldsymbol{y}_4 = \boldsymbol{y}^C_4 - \boldsymbol{y}^D_4$$

The test variables $x^t_1, x^t_2, \cdots \in X^t \subset X^{\mathrm{R}}$ are separable, and the factor $\omega_{i,\cdot}(X^t)$ involves the ith block if and only the following two conditions are satisfied: (1) vectors $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are linearly independent, while $\boldsymbol{y}_3$ and $\boldsymbol{y}_4$ are linearly independent; and (2) both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are not constant vectors. This conclusion can be easily obtained from Theorem 1 proved in Luo et al. (2017). For the above Condition 2, when both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are constant vectors, the ith block does not involve a repeated-variable set factor $\omega_{i,\cdot}(X^t)$. Thus, we can set $\omega_{i,\cdot}(X^t) := 1$. The corresponding factor detection procedure is described in Algorithm 2:

Algorithm 2 Factor detection.

```
1: Input: Target system $f : \mathbb{R}^n \mapsto \mathbb{R}$; Sample data $\mathbf{S} \in \mathbb{R}^{N \times n}$; Set $X^{\mathrm{R}}$ and $X^{\mathrm{NR}}_i$, for $i = 1, 2, \cdots, m$
2: for $i = 1 : m$ do
3:     Let $s_i$ be the number of variables of $X^{\mathrm{NR}}_i$, namely $s_i = \mathrm{card}\left(X^{\mathrm{NR}}_i\right)$
4:     for $j = 1 : s_i$ do
5:         Create $\binom{s_i}{j}$ test variable set $X^t_{(k)}$, $k = 1, 2, \cdots, \binom{s_i}{j}$, which consist of all possible combinations of the set $X^{\mathrm{NR}}_i$ taken $j$ variables at a time; Let Count1 = 1
6:         for $k = 1 : \binom{s_i}{j}$ do
7:             Keep the test variable in $X^t_{(k)}$ and $X^{\mathrm{R}}$ fixed; Re-generate $\mathbf{S}'$; Call BiCT
8:             if $X^t_{(k)}$ is separable detected by BiCT then
9:                 $X^{\mathrm{NR}}_{i,\mathrm{Count1}} := X^t_{(k)}$; Count1 := Count1 + 1
10:    Let $r_i$ be the number of variables of $X^{\mathrm{R}}_i$ (i.e., $X^r$), namely $r_i = \mathrm{card}\left(X^{\mathrm{R}}_i\right)$
11:    for $j = 1 : r_i$ do
12:        Create $\binom{r_i}{j}$ test variable set $X^t_{(k)}$, $k = 1, 2, \cdots, \binom{r_i}{j}$, which consist of all possible combinations of the set $X^{\mathrm{R}}_i$ taken $j$ variables at a time; Let Count2 = 1
13:        for $k = 1 : \binom{r_i}{j}$ do
14:            Generate test functions $\tilde{f}_1$ and $\tilde{f}_2$; Obtain $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$ and $\boldsymbol{y}_4$; Call BiCT
15:            if $X^t_{(k)}$ satisfy separability then
16:                $X^{\mathrm{R}}_{i,\mathrm{Count2}} := X^t_{(k)}$; Count2 := Count2 + 1
17: Output: Variable sets $X^{\mathrm{R}}_{i,j}$ and $X^{\mathrm{NR}}_{i,k}$, for $j = 1, 2, \cdots, p_i$ and $k = 1, 2, \cdots, q_i$
```

4.3. Procedure of multilevel block building

The identification of the separability structure of a target function is a top-to-bottom process. Next, the specific expression of each minimal factor needs to be determined and assembled properly with optimal coefficients to form an optimized model function. This is a bottom-to-top process; that is, MBB involves two phases: structure delectation and model determination (see the MBB flowchart in Fig. 1).

The model determination includes two steps: inner optimization and outer optimization.

Inner optimization is invoked to determine the function expressions and coefficients of all minimal factors $\omega_{i,j}$ and $\psi_{i,k}$. It can also be referred to as factor modeling. Note that the minimal factors usually involve only a small number of variables (usually one or two) and have much less complexity than the original target function. Therefore, factor modeling is relatively easy for most conventional GP or other non-evolutionary algorithms, including PME (Luo & Zhang, 2012), low-dimensional simplex evolution (LSDE) (Luo & Yu, 2012), and artificial bee colony programming (Karaboga et al., 2012). If we choose a genetic programming algorithm or another symbolic regression algorithm as the terminal optimization engine, we must pre-establish a set of arithmetic operations (e.g., + , - , $\times$, $\div$) and mathematical operators (e.g., sin , cos , exp , ln) according to the actual requirements. If we choose a global optimization as the terminal optimization engine, we need to pre-establish a library of element functions. For instance, typical functions (e.g., $a \cdot x + b$, x^2, $1 - \frac{x_2}{x_1}$, $\cdots$) that involve uni-variables and bi-variables could be set as elementary functions. The fac-

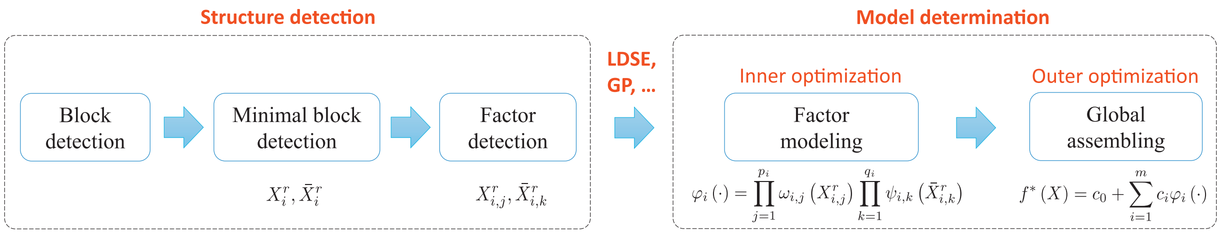


Fig. 1. Flowchart of MBB.

tors are optimized to have small enough fitting errors (e.g., mean square error $\leqslant 10^{-8}$). The minimal factors are multiplied together to produce a minimal block $\varphi_i(\cdot)$.

The outer optimization involves assembling the minimal blocks together to form the final model function. This step is quite simple because the minimal blocks are linearly combined according to the separability structure. Therefore, general linear regression algorithms could accomplish this mission to determine the optimal coefficients c_i, $i = 0, 1, \cdots, m$.

The general procedure of MBB for modeling a GS system can be briefly described as follows:

Procedure of multilevel block building:

Step 1. (Minimal block detection) Partition a GS system into a number of minimal blocks with all repeated variables fixed (see Algorithm 1);

Step 2. (Factor detection) Divide each minimal block into factors (see Algorithm 2);

Step 3. (Factor modeling) Determine the minimal factors by employing an optimization engine;

Step 4. (Global assembling) Combine the optimized factors into minimal blocks multiplicatively and then into an optimization model linearly with optimal coefficients.

5. Case study

In this section, a toy example (Eq. (12)) will be used to illustrate the implementation of the proposed MBB algorithm. The target function involves six independent variables, two of which (x_5 and x_6) are repeated variables.

$$f(\boldsymbol{x}) = \sin 3x_1 - 2(x_5 * x_6)\cos x_2 + e^{x_6}\ln x_3 + x_5 x_4. \tag{12}$$

5.1. Minimal block detection

1) **Level-1 block search.** The first step is a level-1 block search. This step can use the same method as the BBP method proposed in Chen et al. (2018). The black box function (12) is first divided into several additively separable blocks. Eq. (12) can be decomposed into two blocks as follows (Eq. (13)), where each block is boxed separately:

$$f(\boldsymbol{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_2, x_3, x_4, x_5, x_6)}. \tag{13}$$

2) **Level-2 block search.** The second step is the level-2 block search. In this step, the variables are fixed individually to detect whether the number of blocks will increase. In this example, for the detection test loop to variable x_5 (let x_5 be fixed), Eq. (13) can be further rewritten as three additively separate blocks. The number of blocks increases; in other words, the remaining variables or variable combinations, namely x_1, x_4, and (x_3, x_6), can be considered additively separable. This is illustrated in the following equation:

$$f(\boldsymbol{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_4, \cancel{x_5})} + c_3 * \boxed{\varphi_3(x_2, x_3, \cancel{x_5}, x_6)}. \tag{14}$$

Similarly, when x_6 is fixed, we get

$$f(\boldsymbol{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_3, \cancel{x_6})} + c_3 * \boxed{\varphi_3(x_2, x_4, x_5, \cancel{x_6})}. \tag{15}$$

The number of blocks is still three and does not increase, thus concluding the level-2 block search.

3) **Level-3 block search.** The third step is the level-3 block search. All possible two-variable combinations are fixed one-by-one to detect whether the number of blocks will increase. When the variable combination (x_5, x_6) is fixed to a certain point, the number of blocks increases to four:

$$f(\boldsymbol{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_4, \cancel{x_5})} + c_3 * \boxed{\varphi_3(x_2, \cancel{x_6}, \cancel{x_5})} + c_4 * \boxed{\varphi_4(x_3, \cancel{x_6})}. \tag{16}$$

As shown in Eq. (16), only one variable exists in each block (x_5 and x_6 are fixed). These blocks are called minimal blocks. Thus far, we have decomposed all variables into repeated variables (x_5, x_6), non-repeated variables (x_1, x_2, x_3, x_4), and four minimal blocks, rendering further decomposition unnecessary. We can obtain the following structure of the black box function from Eq. (17):

$$f(\boldsymbol{x}) = c_1 * \boxed{\tilde{\omega}_1(x_5, x_6)\tilde{\psi}_1(x_1)} + c_2 * \boxed{\tilde{\omega}_2(x_5, x_6)\tilde{\psi}_2(x_2)} + c_3 * \boxed{\tilde{\omega}_3(x_5, x_6)\tilde{\psi}_3(x_3)} + c_4 * \boxed{\tilde{\omega}_4(x_5, x_6)\tilde{\psi}_4(x_4)} \tag{17}$$

Fig. 2illustrates an example of the 3-level block search as described in Eq. (12). We must emphasize that, in the process of minimal block detection, only the variable type and the number of minimal blocks are determined; the specific expression of each block has yet to be identified.

5.2. Factor detection

In this process, the minimal blocks will be further decomposed into a number of minimal factors. Similarly, only the function structure of each factor, not the function specific, needs to be determined.

(1) **Factors of non-repeated variables.** In this step, the task is to identify non-repeated variable factors, which can be implemented using the same method as the factor detection process proposed in Chen et al. (2018). Essentially, after all repeated variables (i.e., x_5 and x_6) are fixed, the factors in each minimal block must be multiplicatively separable. In this example, the factors of non-repeated variables in each minimal block are $\psi_1(x_1)$, $\psi_2(x_2)$, $\psi_3(x_3)$, and $\psi_4(x_4)$.

(2) **Factors of repeated variables.** This step is slightly more difficult because the sampling of repeated variables in one minimal block will affect other minimal blocks. For example, when we try to detect the separability of x_5 and x_6 in minimal block-2, the sampling process affects the other minimal

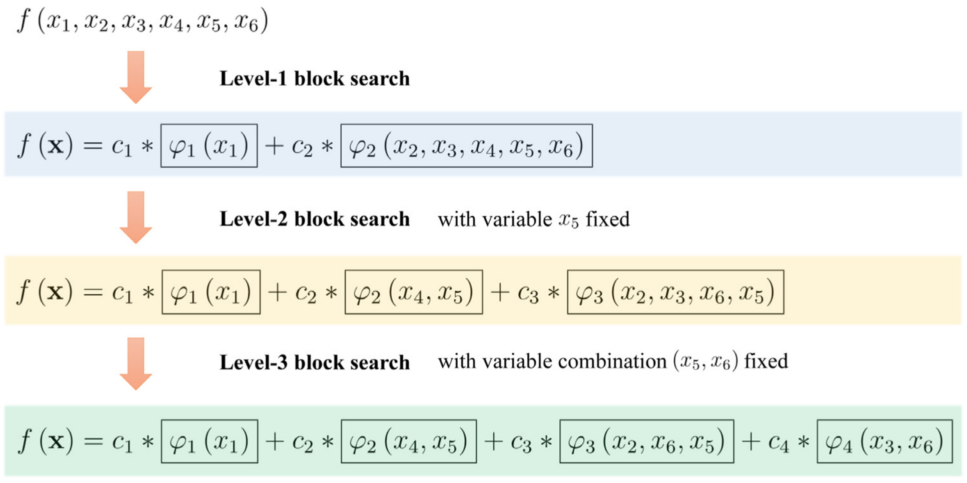

$f(x_1, x_2, x_3, x_4, x_5, x_6)$
Level-1 block search
$f(\mathbf{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_2, x_3, x_4, x_5, x_6)}$
Level-2 block search with variable x_5 fixed
$f(\mathbf{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_4, x_5)} + c_3 * \boxed{\varphi_3(x_2, x_3, x_6, x_5)}$
Level-3 block search with variable combination (x_5, x_6) fixed
$f(\mathbf{x}) = c_1 * \boxed{\varphi_1(x_1)} + c_2 * \boxed{\varphi_2(x_4, x_5)} + c_3 * \boxed{\varphi_3(x_2, x_6, x_5)} + c_4 * \boxed{\varphi_4(x_3, x_6)}$

Fig. 2. Illustration of 3-level block search for Eq. (12).

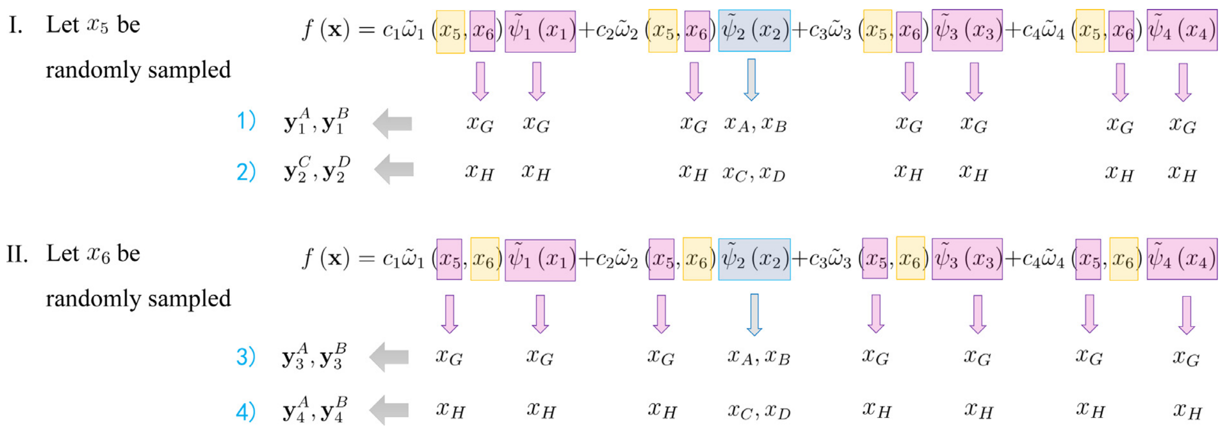

I. Let x_5 be randomly sampled
$f(\mathbf{x}) = c_1\tilde{\omega}_1(x_5, x_6)\tilde{\psi}_1(x_1) + c_2\tilde{\omega}_2(x_5, x_6)\tilde{\psi}_2(x_2) + c_3\tilde{\omega}_3(x_5, x_6)\tilde{\psi}_3(x_3) + c_4\tilde{\omega}_4(x_5, x_6)\tilde{\psi}_4(x_4)$
1) $\mathbf{y}_1^A, \mathbf{y}_1^B$
x_G x_G x_G x_A, x_B x_G x_G x_G x_G
2) $\mathbf{y}_2^C, \mathbf{y}_2^D$
x_H x_H x_H x_C, x_D x_H x_H x_H x_H
II. Let x_6 be randomly sampled
$f(\mathbf{x}) = c_1\tilde{\omega}_1(x_5, x_6)\tilde{\psi}_1(x_1) + c_2\tilde{\omega}_2(x_5, x_6)\tilde{\psi}_2(x_2) + c_3\tilde{\omega}_3(x_5, x_6)\tilde{\psi}_3(x_3) + c_4\tilde{\omega}_4(x_5, x_6)\tilde{\psi}_4(x_4)$
3) $\mathbf{y}_3^A, \mathbf{y}_3^B$
x_G x_G x_G x_A, x_B x_G x_G x_G x_G
4) $\mathbf{y}_4^A, \mathbf{y}_4^B$
x_H x_H x_H x_C, x_D x_H x_H x_H x_H

Fig. 3. Sampling method of detecting the separability of the repeated variables x_5 and x_6 in minimal block-2.

blocks (because x_5 and x_6 also exist in minimal blocks-3 and -4). Next, we take minimal block-2 as an example, namely

$$\varphi_2(x_2, x_5, x_6) = \tilde{\omega}_2(x_5, x_6) * \tilde{\psi}_2(x_2),$$

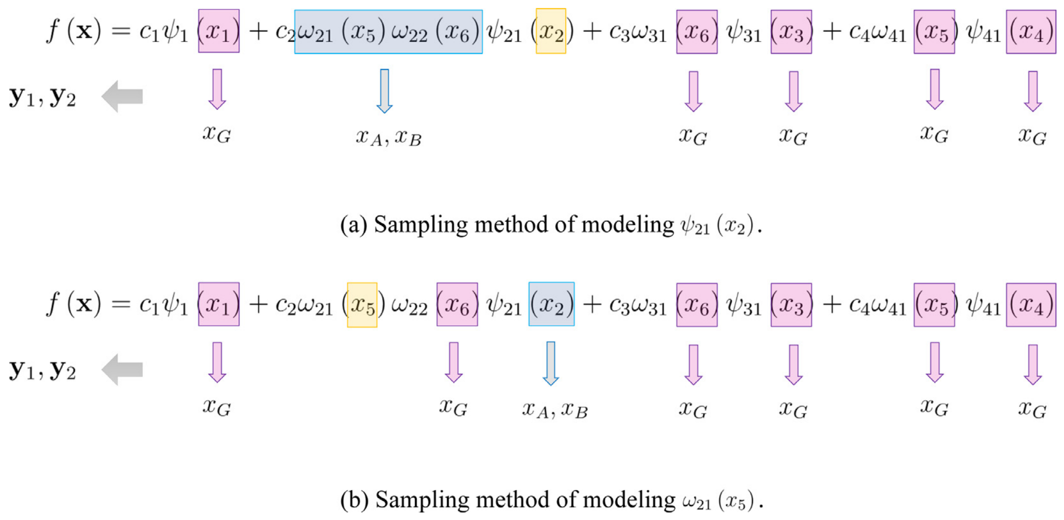
to illustrate the implementation of the separability detection of repeated variables x_5 and x_6. The main sampling process is shown in Fig. 3. Recall from Section 4.2 that four groups with eight sampling iterations should be evaluated. First, let variable x_5 be randomly sampled. In the first group, the non-repeated variable x_2 in minimal block-2 is fixed to x_A and x_B respectively; all other variables are fixed to x_G, involving the repeated variable x_6 and non-repeated variables x_1, x_2, x_3, and x_4. Then we can obtain two vectors, $\boldsymbol{y}_1^A$ and $\boldsymbol{y}_1^B$. Let $\boldsymbol{y}_1 = \boldsymbol{y}_1^A - \boldsymbol{y}_1^B$. Notably, we sample twice in the first group to eliminate the effects of repeated variables in the other minimal blocks. In the second group, the non-repeated variable x_2 in minimal block-2 is fixed to x_C and x_D, respectively; all other variables are fixed to x_H. Then we can obtain two vectors, $\boldsymbol{y}_2^C$ and $\boldsymbol{y}_2^D$. Let $\boldsymbol{y}_2 = \boldsymbol{y}_2^C - \boldsymbol{y}_2^D$. In the third and fourth groups, vectors $\boldsymbol{y}_3$ and $\boldsymbol{y}_4$ can be obtained similarly. Luo et al. (2017) demonstrated that if the corresponding components of function-value vectors $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are proportional, and those of vectors $\boldsymbol{y}_3$ and vector $\boldsymbol{y}_4$ are simultaneously proportional, then variables x_5 and x_6 are multiply separable. The other two factors, $\omega_{31}(x_6)$ and $\omega_{41}(x_5)$, can be obtained similarly. After the factor detection process, the structure of the black box function Eq. (12) can be written as follows:

$$f(\boldsymbol{x}) = c_1\psi_1(x_1) + c_2\omega_{21}(x_5)\omega_{22}(x_6)\psi_{21}(x_2) + c_3\omega_{31}(x_6)\psi_{31}(x_3) + c_4\omega_{41}(x_5)\psi_{41}(x_4) \quad (18)$$

5.3. Factor modeling

(1) **Modeling factors of non-repeated variables.** Here, we use an example, $\psi_{21}(x_2)$, to illustrate how to model factors of non-repeated variables. The main sampling process is illustrated in Fig. 4(a). Let the variable x_2 be randomly sampled, where the sample points are denoted by the vector $\boldsymbol{x}_{\text{train}}$. Fix the repeated variables, x_5 and x_6, to points x_A and x_B, respectively. Simultaneously, keep all other variables fixed to point x_G. Then, we can get two function vectors, $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. Let $\boldsymbol{y}_{\text{train}} = \boldsymbol{y}_1 - \boldsymbol{y}_2$. Substitute $\boldsymbol{y}_{\text{train}}$ and $\boldsymbol{x}_{\text{train}}$ into the fitting model $y_{\text{train}} = \beta \cdot f^*(x_{\text{train}})$. Keep in mind that the aim is to obtain the optimization model f^*, and the constant β will be discarded. This step could be realized via an existing optimization engine. Recall from Section 4.3 that any conventional GP method and global optimization (GO) algorithm could be chosen as the optimization engine of MBB.

(2) **Modeling factors of repeated variables.** Modeling factors of repeated variables uses the same procedure as the previous method. Here, we use the factor $\omega_{21}(x_5)$ to illustrate it. The main sampling process is depicted in Fig. 4 (b). Let the variable x_5 be randomly sampled, where the sample points are denoted by the vector $\boldsymbol{x}_{\text{train}}$. Similarly, after sam-



$f(\mathbf{x}) = c_1\psi_1(x_1) + c_2\omega_{21}(x_5)\omega_{22}(x_6)\psi_{21}(x_2) + c_3\omega_{31}(x_6)\psi_{31}(x_3) + c_4\omega_{41}(x_5)\psi_{41}(x_4)$
$\mathbf{y}_1, \mathbf{y}_2$
x_G
x_A, x_B
(a) Sampling method of modeling $\psi_{21}(x_2)$.
(b) Sampling method of modeling $\omega_{21}(x_5)$.

Fig. 4. Sampling methods of modeling factors $\psi_{21}(x_2)$ and $\omega_{21}(x_5)$.

pling twice, we can get two function vectors, $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$. Let $\boldsymbol{y}_{\text{train}} = \boldsymbol{y}_1 - \boldsymbol{y}_2$. Substitute $\boldsymbol{y}_{\text{train}}$ and $\boldsymbol{x}_{\text{train}}$ into the fitting model $y_{\text{train}} = \beta \cdot f^*(x_{\text{train}})$. Using the optimization engine, we can obtain the regression model f^*.

5.4. Global assembling

The final task is to assemble these factors into minimal blocks. The minimal blocks are considered as basic functions of the target model (12). To determine the global parameter c_k, $k = 0, 1, \cdots, m$, of Eq. (12), we can use the conventional linear fitting method, which can be obtained by the following equation:

$$\mathbf{f}_{\text{train}} = c_0 + \sum_{i=1}^{m} c_i \prod_{j=1}^{p_i} \omega_{i,j}\left(\boldsymbol{x}_{\text{train}}^{\text{R}}\right) \prod_{k=1}^{q_i} \psi_{i,k}\left(\boldsymbol{x}_{\text{train}}^{\text{NR}}\right). \tag{19}$$

6. Numerical results

In our implementation, LDSE (Luo & Yu, 2012) is chosen as the optimization engine. LDSE is a hybrid evolutionary algorithm for continuous global optimization. The efficiency of LDSE-powered MBB is tested by comparing the method with a state-of-the-art symbolic regression tool, Eureqa (Schmidt & Lipson, 2009), a proprietary A.I.-powered modeling engine based on GP, developed by Dr. Hod Lipson from the Computational Synthesis Lab at Cornell University. The efficiency is evaluated by the structure optimization and coefficient optimization abilities.

Two test groups are taken into account: Group A consists of 10 toy cases (see Appendix A), and Group B consists of three real-world cases (see Section 2). The range of variables is $(-3, 0) \cup (0, 3)$ for all toy cases ((1, 3) for Case 6) and $(0, +\infty)$ for all engineering cases.

We take the MSE as the fitting error in the numerical study of for each run, which is used to evaluate the regression model. The MSE is defined by Eq. (20),

$$\text{MSE}(f, f^*) = \frac{\|f - f^*\|_2^2}{N}, \tag{20}$$

where N is the number of sampling points, and f and f^* are vectors of predicted and observed values at these sampling points. The calculation of predicted vector f is similar to Eq. (19). The observed vector f^* is obtained from the original system.

The MBB computing time consists of three parts, $t = t_1 + t_2 + t_3$, where t_1 denotes time for separability detection, t_2 denotes time for factor modeling, and t_3 denotes time for global assembly. In Luo et al. (2017) demonstrated that the separability detection and function recovery processes are double-precision operations and thus cost much less time than the factor determination process; therefore, the factor modeling process consumes most of the total computing time (i.e., $t \approx t_2$).

Table 1
Pre-established function models of uni-variable and bi-variables in LDSE powered MBB.

No.	Uni-variable model	Bi-variables model
1	x^{m_1}	$m_1x_1 + m_2x_2$
2	e^{m_1x}	$e^{m_1x_1x_2}$
3	$\sin(m_1x + m_2)$	$(x_1/x_2)^{m_1} + m_2(x_1/x_2)^{m_3} + m_4$
4	$\log(m_1x + m_2)$	$\sin(m_1x_1 + m_2x_2 + m_3x_1x_2 + m_4)$

6.1. Toy cases

In this test group, 10 problems (see Appendix A) are chosen to test the efficiency of MBB. In Table 2, the case number, dimension, number of sampling points, number of minimal blocks, number of factors, and MSE variance from the 20 runs are denoted as Case no., Dim, No. samples, No. blocks, No. factors, and MSE, respectively.

6.1.1. Control parameter setting

For calculation conditions, the number of sampling points for each independent variable is 200. The control parameters in LDSE are as follows: the upper and lower bounds of the fitting parameters are set as −50 and 50. The population size N_p is set to $N_p = 10 + 10d$, where d is the dimension of the problem. The maximum generations are set to $3N_p$. The maximum number of partially separable variables in all target models in our tests is 2; hence, our uni-variable and bi-variable function library of MBB is set as shown in Table 1, in which m_1, m_2 are the parameters to be fitted. The sequence search and optimization method comprise a suitable GO strategy. The search will exit immediately if the MSE is small enough (MSE $\le \varepsilon_{\text{target}}$), and the tolerance (fitting error) is $\varepsilon_{\text{target}} = 10^{-6}$. To reduce the effect of randomness, each test case was executed 20 times.

6.1.2. Numerical results and comparison

Table 2 shows the comparative results of the average efficiency of MBB and Eureqa for the 20 independent runs with different

Table 2
Comparison of average efficiency of LDSE powered MBB and Eureqa for modeling 10 toy cases listed in Appendix A.

Case No.	Dim	No. samples	LDSE powered MBB					Eureqa		
			Repeated variable	No. block	No. factor	CPU time[a]	MSE	CPU time[b]	MSE	Remarks
1	2	400	None	1	2	**7 s**	$\leq\varepsilon_{\text{target}}$	7 s	$\leq\varepsilon_{\text{target}}$	Solutions are all exact
2	3	600	None	2	2	**9 s**	$\leq\varepsilon_{\text{target}}$	$>$ 4 m 12 s	$[0, 2.33]\times10^{-8}$	10 runs failed[c]
3	3	600	None	2	3	**9 s**	$\leq\varepsilon_{\text{target}}$	$>$ 1 m 9 s	$\leq\varepsilon_{\text{target}}$	2 runs failed
4	3	600	x_3	2	4	**11 s**	$\leq\varepsilon_{\text{target}}$	55 s	$\leq\varepsilon_{\text{target}}$	Solutions are all exact
5	4	800	x_4	2	5	**14 s**	$\leq\varepsilon_{\text{target}}$	$>$ 2 m 28 s	$\leq\varepsilon_{\text{target}}$	3 runs failed
6	5	1000	x_5	4	6	**21 s**	$\leq\varepsilon_{\text{target}}$	$\gg$ 6 m 25 s	$[4.79, 14.2]\times10^{-6}$	All runs failed
7	5	1000	x_4, x_5	3	7	**16 s**	$\leq\varepsilon_{\text{target}}$	$\gg$ 8 m 38 s	$[4.05, 7.68]\times10^{-4}$	All runs failed
8	5	1000	x_4	3	6	**15 s**	$\leq\varepsilon_{\text{target}}$	$\gg$ 6 m 44 s	$[2.89, 122.86]\times10^{-2}$	All runs failed
9	6	1200	None	1	4	**9 s**	$\leq\varepsilon_{\text{target}}$	$\gg$ 6 m 59 s	$[1.4, 8.54]\times10^{-1}$	All runs failed
10	7	1400	x_7	2	6	**11 s**	$\leq\varepsilon_{\text{target}}$	$\gg$ 6 m 51 s	$[7.58, 399.5]\times10^{-4}$	All runs failed

[a] LDSE powered MBB is implemented in MATLAB and executed on a single processor.
[b] Eureqa is implemented in C language and executed in parallel on 8 processors.
[c] Maximum generation is set to 100,000.

initial populations. Numerical results show that the LDSE-powered MBB successfully recovered all target models exactly in double precision. Once the uni- and bi-variables models are preset, the sequence search method allows MBB to easily identify the best regression model. In practical applications, more function models could be added to the pre-established function models of MBB, provided they are needed. As shown in the second column of Table 2, Eureqa failed to reach the target accuracy in all 20 runs within the maximum generation (100,000) for Cases 6–10.

The results of Eureqa were quite unexpected at first sight as Eureqa is assumed to be a state-of-the-art symbolic regression software implemented in C language and executed in parallel on eight processors. However, the result is reasonable because the **separability features of these cases are not used in Eureqa**. The complexity of Cases 6–10 is rather high because of the relative large number of **active variables** and the combinations of different types of functions along with the non-integer coefficients inside the subfunctions. The search space is indeed quite large; hence, stochastic algorithms are easy to get premature.

For MBB, a stochastic search algorithm is used only for minimal factor modeling, and the search space is substantially reduced. Other processes of MBB are nearly determinate.

6.2. Real-world cases

In this test group, numerical experiments on four real-world problems discussed in Section 2 are conducted, namely Eqs. (3), (5), and (7). This test group helps to evaluate the potential feasibility of MBB in practical applications.

6.2.1. Control parameter setting

For Eq. (3), the sample set consists of 100 observations uniformly distributed in a box in R^3 (i.e., $A^* = 0.5 : 1.5\ \text{m}^2$; $p_0 = 4\times10^5 : 6\times10^5$ Pa; $T_0 = 250 : 260$ K). The specific gas constant R is set to $R = 287\ \text{J}/(\text{kg}\cdot\text{K})$, and the ratio of specific heat γ is set to $\gamma = 1.4$ while detecting and modeling Eq. (3).

For Eq. (5), the sample set consists of 100 observations uniformly distributed in a box in R^6 (i.e., $C_{L\alpha} = 0.4 : 0.8$; $\alpha = 5 : 10^\circ$; $C_{L\delta_e} = 0.4 : 0.8$; $\delta_e = 5 : 10^\circ$; $S_{\text{HT}} = 1 : 1.5\ \text{m}^2$; $S_{\text{ref}} = 5 : 7\ \text{m}^2$). The zero-lift angle of attack α_0 is -2°.

For Eq. (7), the sample set consists of 100 observations uniformly distributed in a box in R^5 (i.e., $V_\infty = 60 : 65$; m/s; $\theta = 30 : 40^\circ$; $r = 0.2 : 0.5$ m; $R = 0.5 : 0.8$ m; $\Gamma = 5 : 10\ \text{m}^2/\text{s}$). In our tests, the matrix of 100 sample data is directly used for global assembling to determine the coefficients of the final model function. Meanwhile, the colons corresponding to unfixed variables of the matrix are also used in all phases of MBB including block detection, minimal block detection, factor detection, and factor modeling. The colons corresponding to fixed variables are fixed to the elements of the first row. That is, the samples are partially renewed for every BiCT detection, but the number of using samples always remains at 100.

Table 3
Average efficiency of LDSE powered MBB for modeling 3 real-world examples (minimal block determination).

Target minimal block	CPU Time[a]	Result expression
Case 11: Eq. (3)		
$\varphi_1(p_0, A^*, T_0)$	7s	$\varphi_1 = 0.04 * p_0 * A^* * T_0^{-0.5}$
Case 12: Eq. (5)		
$\varphi_1(C_{L\alpha}, \alpha)$	5s	$\varphi_1 = C_{L\alpha} * (\alpha - 2)$
$\varphi_2(C_{L\delta_e}, \delta_e, S_{\text{HT}}, S_{\text{ref}})$	8s	$\varphi_2 = C_{L\delta_e} * \delta_e * S_{\text{HT}} * 1/S_{\text{ref}}$
Case 13: Eq. (7)		
$\varphi_1(V_\infty, r, \sin\theta, R)$	7s	$\varphi_1 = V_\infty * \sin\theta * (r - R^2/r)$
$\varphi_2(\Gamma, r, R)$	5s	$\varphi_2 = 0.1592 * \Gamma * \ln(r/R)$

[a] The CPU time refers to the time cost of the minimal block determination.

The control parameters and uni-variable and bi-variable model library in LDSE are the same as in the first test group. Similarly, the search will exit immediately if the MSE is small enough (MSE $\leqslant \varepsilon_{\text{target}} = 10^{-8}$). To reduce the effect of randomness, each test case was executed 20 times, and average values are reported.

6.2.2. Numerical results and comparison

The results of this test group facilitate further analysis of the efficiency of the LDSE-powered MBB. Table 3 shows MBB results for the minimal block detections in the four cases. Table 4 shows the comparative results of the average efficiency of MBB and Eureqa on the 20 independent runs. As indicated in Table 3, the minimal blocks were detected correctly and fitted accurately with MSE $\leqslant \varepsilon_{\text{target}} = 10^{-8}$ for all 20 runs.

Meanwhile, the comparison of MSEs in Table 4 indicates that LDSE-powered MBB demonstrated a much lower MSE for all cases. From the above results, we can determine that the LDSE-powered MBB is highly capable of detecting the complex structures of GS systems and of modeling complex practical problems. Good efficiency for the modeling of a complex system and the capability of structure optimization and coefficient optimization reflect the potential of MBB for use in practical applications.

The above results show that the LDSE-powered MBB has better computational efficiency than the state-of-the-art symbolic tool, Eureqa. In fact, the efficiency could be much better because the LDSE-powered MBB was executed on a single processor in numerical tests, whereas Eureqa was executed in parallel using eight processors.

Table 4
Comparison of average efficiency of LDSE powered MBB and Eureqa for modeling 3 real-world cases.

Case No.	Dim	LDSE powered MBB				Eureqa		
		No. block	No. factor	CPU time	MSE	CPU time	MSE	Remarks
11	3	1	3	**10 s**	$\le \varepsilon_{\text{target}}$	53 s	$\le \varepsilon_{\text{target}}$	2 runs failed
12	6	2	6	**15 s**	$\le \varepsilon_{\text{target}}$	$\gg$7 m 3 s	$[2.04, 5.63] \times 10^{-4}$	All runs failed
13	5	2	6	**16 s**	$\le \varepsilon_{\text{target}}$	$>$4 m 41 s	$[0, 1.96] \times 10^{-5}$	11 runs failed

7. Discussion

So far, the proposed method has been described using functions with explicit expressions. In fact, MBB only works if we have full control over the underlying system and are free to take samples, such as when attempting to identify a simple function to approximate a computationally expensive computational fluid dynamic (CFD) simulation or to identify a more concise equivalent formula with a given symbolic expression (known as exact simplification and transformation; see Stoutemyer, 2012). This is a limitation of current MMB, although it could be modified in future studies.

Note that Eureqa as well as most other symbolic regression algorithms and tools do not need to take samples while modeling. That is, Eureqa works with the data at hand and has broader application fields in this sense. In other words, MMB can outperform Eureqa only if we have full control over the underlying system to take samples freely and the target model function processes some types of separability features.

8. Conclusion

Based on the observations of different separability types in practical engineering formulas, a more general concept of separability is defined to handle repeated variables that appear more than once in the target model. To identify the structure of a function with a possible GS feature, an MBB algorithm is proposed in which variables are distinguished as repeated variables and non-repeated variables and the target model is decomposed into a higher level of blocks and factors until they are confirmed to be minimal blocks and factors. The minimal factors often involve only a small number of variables (usually one or two) and are much less complex than the original target function. Thus, it is much easier to be identified for most conventional GP or other non-evolutionary algorithms. The minimal blocks and factors are then assembled properly to form the target function.

The newly proposed MBB is an improved version of BBP with more general application potential. The efficiency of MBB was tested through a comparison with Eureqa, a state-of-the-art symbolic regression tool. Test results indicate that MBB is more effective, efficient, and can recover all investigated cases quickly and reliably. MBB is a promising algorithm to model engineering systems with separability features.

In future work, we plan to study the robustness of the proposed algorithm in practical applications with sample data that involve noise (e.g., aerodynamic force/heating predictions using experimental data from wind tunnels in which measurement errors and/or system noise are inevitable). Special techniques are necessary to make MBB applicable. For example, a soft BiCT might be developed to replace the standard BiCT used in the current MBB algorithm. In soft BiCT, the correlation coefficient $|\rho|$ could be defined as $|\rho| = 1 - \varepsilon$ (where ε is a small positive number) instead of $|\rho| = 1$ in standard BiCT. In addition, the noise could be further suppressed via multiple BiCTs, where each variable is fixed to more pairs of vectors (currently one pair in standard BiCT). Finally, a preconditioning step should be designed to provide sample points for the MBB method. For practical applications in aerodynamic prediction, this step might include a set of CFD simulations (Blazek, 2015) or a sufficiently accurate surrogate model of the dataset (Forrester, Sobester, & Keane, 2008). The CFD simulation or surrogate model could be used to take samples for block detection in MBB. Detailed discussions will be provided in future studies.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 11532014). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on the earlier versions of this manuscript.

Appendix A. 10 toy cases of Section 6.1

The target models which are tested in Section 6.1 with all the minimal blocks boxed are given as follows:

Case 1. $f(\boldsymbol{x}) = 0.5 * \boxed{e^{x_1} * \sin 2x_2}$, where $x_i \in [-3, 3], i = 1, 2$.

Case 2. $f(\boldsymbol{x}) = 2 * \boxed{\cos x_1} + \boxed{\sin(3x_2 - x_3)}$, where $x_i \in [-3, 3], i = 1, 2, 3$.

Case 3. $f(\boldsymbol{x}) = 1.2 + 10 * \boxed{\sin 2x_1} - 3 * \boxed{x_2^2 * \cos x_3}$, where $x_i \in [-3, 3], i = 1, 2, 3$.

Case 4. $f(\boldsymbol{x}) = \boxed{x_3 * \sin x_1} - 2 * \boxed{x_3 * \cos x_2}$, where $x_i \in [-3, 3], i = 1, 2, 3$.

Case 5. $f(\boldsymbol{x}) = 2 * \boxed{x_1 * \sin x_2 * \cos x_4} - 0.5 * \boxed{x_4 * \cos x_3}$, where $x_i \in [-3, 3], i = 1, 2, 3, 4$.

Case 6. $f(\boldsymbol{x}) = 10 + 0.2 * \boxed{x_1} - 0.2 * \boxed{x_5^2 * \sin x_2} + \boxed{\cos x_5 * \ln(3x_3 + 1.2)} - 1.2 * \boxed{e^{0.5x_4}}$, where $x_i \in [1, 4], i = 1, 2, \cdots, 5$.

Case 7. $f(\boldsymbol{x}) = 2 * \boxed{x_4 * x_5 * \sin x_1} - \boxed{x_5 * x_2} + 0.5 * \boxed{e^{x_3} * \cos x_4}$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 5$.

Case 8. $f(\boldsymbol{x}) = 1.2 + 2 * \boxed{x_4 * \cos x_2} + 0.5 * \boxed{e^{1.2x_3} * \sin 3x_1 * \cos x_4} - 2 * \boxed{\cos(1.5x_5 + 5)}$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 5$.

Case 9. $f(\boldsymbol{x}) = 0.5 * \boxed{\dfrac{\cos(x_3 x_4)}{e^{x_1} * {x_2}^2} * \sin(1.5x_5 - 2x_6)}$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 6$.

Case 10. $f(\boldsymbol{x}) = 1.2 - 2 * \boxed{\dfrac{x_1 + x_2}{x_3} * \cos x_7} + 0.5 * \boxed{e^{x_7} * x_4 * \sin(x_5 x_6)}$, where $x_i \in [-3, 3], i = 1, 2, \cdots, 7$.

References

Anderson, J. D. (2006). *Hypersonic and high-temperature gas dynamics* (2nd). Virginia: American Institute of Aeronautics and Astronautics, Inc..

Anderson, J. D. (2011). *Fundamentals of aerodynamics* (5th). New York: MacGraw-Hill.

Blazek, J. (2015). *Computational fluid dynamics: Principles and applications* (3rd). Kidlington, Oxford, UK: Elsevier Ltd..

Ceperic, V., Bako, N., & Baric, A. (2014). A symbolic regression-based modelling strategy of ac/dc rectifiers for rfid applications. *Expert Systems with Applications, 41*(16), 7061–7067. doi:10.1016/j.eswa.2014.06.021.

Chen, C., Luo, C., & Jiang, Z. (2017). Fast modeling methods for complex system with separable features. In *2017 10th international symposium on computational intelligence and design (ISCID), Hangzhou, China: 1* (pp. 201–204). doi:10.1109/ISCID.2017.144.

Chen, C., Luo, C., & Jiang, Z. (2018). Block building programming for symbolic regression. *Neurocomputing, 275*, 1973–1980. doi:10.1016/j.neucom.2017.10.047.

Forrester, A., Sobester, A., & Keane, A. (2008). *Engineering design via surrogate modelling: A practical guide*. John Wiley & Sons.

Gan, Z., Chow, T. W., & Chau, W. (2009). Clone selection programming and its application to symbolic regression. *Expert Systems with Applications, 36*(2), 3996–4005. doi:10.1016/j.eswa.2008.02.030.

Guo, H., & Li, Z. (2012). Structural damage identification based on Bayesian theory and improved immune genetic algorithm. *Expert Systems with Applications, 39*(7), 6426–6434. doi:10.1016/j.eswa.2011.12.023.

Gusel, L., & Brezocnik, M. (2011). Application of genetic programming for modelling of material characteristics. *Expert Systems with Applications, 38*(12), 15014–15019. doi:10.1016/j.eswa.2011.05.045.

Karaboga, D., Ozturk, C., Karaboga, N., & Gorkemli, B. (2012). Artificial bee colony programming for symbolic regression. *Information Sciences, 209*, 1–15. doi:10.1016/j.ins.2012.05.002.

Kinzett, D., Johnston, M., & Zhang, M. (2009). How online simplification affects building blocks in genetic programming. In *Proceedings of the 11th annual conference on genetic and evolutionary computation, Montreal, Canada* (pp. 979–986). doi:10.1145/1569901.1570035.

Kinzett, D., Zhang, M., & Johnston, M. (2008). *Using numerical simplification to control bloat in genetic programming* (pp. 493–502)). Berlin, Heidelberg: Springer.

Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection* (5th). Cambridge, MA: MIT Press.

Li, Z., Zhang, H., Bailey, S. C., Hoagg, J. B., & Martin, A. (2017). A data-driven adaptive Reynolds-averaged Navier–Stokes k model for turbulent flow. *Journal of Computational Physics, 345*, 111–131. doi:10.1016/j.jcp.2017.05.009.

Luo, C., Chen, C., & Jiang, Z. (2017). A divide and conquer method for symbolic regression. arXiv:1705.08061.

Luo, C., Hu, Z., Zhang, S.-L., & Jiang, Z. (2015). Adaptive space transformation: An invariant based method for predicting aerodynamic coefficients of hypersonic vehicles. *Engineering Applications of Artificial Intelligence, 46*, 93–103. doi:10.1016/j.engappai.2015.09.001.

Luo, C., & Yu, B. (2012). Low dimensional simplex evolution a new heuristic for global optimization. *Journal of Global Optimization, 52*(1), 45–55. doi:10.1007/s10898-011-9678-1.

Luo, C., & Zhang, S.-L. (2012). Parse-matrix evolution for symbolic regression. *Engineering Applications of Artificial Intelligence, 25*(6), 1182–1193. doi:10.1016/j.engappai.2012.05.015.

McConaghy, T. (2011). *Ffx: Fast, scalable, deterministic symbolic regression technology* (pp. 235–260)). New York: Springer.

McRee, R. K., Software, K., & Park, M. (2010). Symbolic regression using nearest neighbor indexing. In *Proceedings of the 12th annual conference companion on genetic and evolutionary computation, Portland, Oregon, USA* (pp. 1983–1990). doi:10.1145/1830761.1830841.

Mehr, A. D., & Nourani, V. (2017). A pareto-optimal moving average-multigene genetic programming model for rainfall-runoff modelling. *Environmental Modelling & Software, 92*, 239–251. doi:10.1016/j.envsoft.2017.03.004.

O'Neill, M., & Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation, 5*(4), 349–358. doi:10.1109/4235.942529.

Schmidt, M., & Lipson, H. (2009). Eureqa (version 1.24.0) [software]. Available from http://www.nutonian.com; [accessed: 2017.05.01].

Shokouhifar, M., & Jalali, A. (2015). An evolutionary-based methodology for symbolic simplification of analog circuits using genetic algorithm and simulated annealing. *Expert Systems with Applications, 42*(3), 1189–1201. doi:10.1016/j.eswa.2014.09.030.

Stoutemyer, D. R. (2012). Can the Eureqa symbolic regression program, computer algebra and numerical analysis help each other?arXiv:1203.1023.

Volaric, I., Sucic, V., & Stankovic, S. (2017). A data driven compressive sensing approach for time-frequency signal enhancement. *Signal Processing, 141*, 229–239. doi:10.1016/j.sigpro.2017.06.013.

Wong, K. Y., Yip, C. L., & Li, P. W. (2008). Automatic identification of weather systems from numerical weather prediction data using genetic algorithm. *Expert Systems with Applications, 35*(1), 542–555. doi:10.1016/j.eswa.2007.07.032.

Worm, A. (2016). *Prioritized Grammar Enumeration: A novel method for symbolic regression*. Binghamton University - State University of New York Ph.D. thesis..

Yang, Y. W., Wang, C., & Soh, C. K. (2005). Force identification of dynamic systems using genetic programming. *International Journal for Numerical Methods in Engineering, 63*(9), 1288–1312. doi:10.1002/nme.1323.

Zarifi, M. H., Satvati, H., & Baradaran-nia, M. (2015). Analysis of evolutionary techniques for the automated implementation of digital circuits. *Expert Systems with Applications, 42*(21), 7620–7626. doi:10.1016/j.eswa.2015.06.005.

Zhang, X. (2002). *Aircraft design handbook*: 6. Beijing, China: Aviation Industry Press.