Fast knot optimization for multivariate adaptive regression splines using hill climbing methods

Xinglong Ju[a,*], Victoria C. P. Chen[a], Jay M. Rosenberger[a], Feng Liu[b,c]

[a]*Department of Industrial, Manufacturing, & Systems Engineering*
The University of Texas at Arlington, Arlington, TX 76019, USA
[b]*Department of Anesthesia, Critical Care and Pain Medicine*
Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, USA
[c]*The Picower The Picower Institute for Learning and Memory*
Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Abstract

Multivariate adaptive regression splines (MARS) is a statistical modeling approach with wide real-world applications. In the MARS model building process, knot positioning is a critical step that potentially affects the accuracy of the final MARS model. Identifying well-positioned knots entails assessing the quality of many knots in each model building iteration, which requires much computation efforts. By exploring the change in the residual sum of squares (RSS) within MARS, we find that local optima from previous iterations can be very close to those of the current iteration. In our approach, the prior change in RSS information is used to "warm start" an optimal knot positioning. We propose two methods for MARS knot positioning. The first method is a hill climbing method (HCM), which ignores prior change in RSS information. The second method is a hill climbing method using prior change in RSS information (PHCM). Numerical experiments are conducted on data with up to 30 dimensions. Our results show that both versions of hill climbing methods outperform Chen's MARS knot selection method on datasets with different noise levels. Further, PHCM using prior change in RSS information performs best in both accuracy and computational speed. In addition,

*This is to indicate the corresponding author.
Email address: `xinglong.ju@mavs.uta.edu` (Xinglong Ju)

an open source Python code will be available upon acceptance of the paper on GitHub (`https://github.mit.edu/fengliu/MARSHC`).

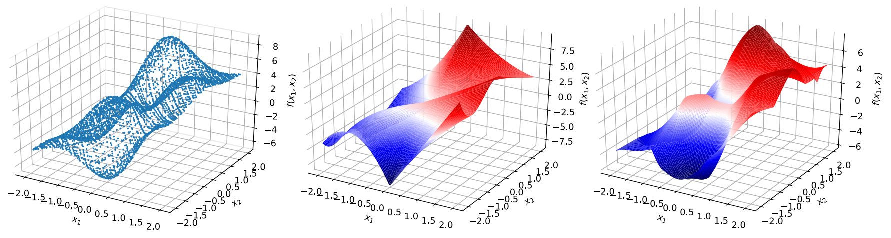
Keywords: MARS, Regression, Knot optimization, Knot positioning, Hill climbing

2010 MSC: 00-01, 99-00

1. Introduction

As a popular non-parametric regression technique, multivariate adaptive regression splines (MARS) algorithm was first introduced by Friedman in 1991 [1]. Because of its flexibility and accuracy, MARS has been used in many studies including predicting distributions of freshwater diadromous fish [2], analyzing relationships between the distributions of 15 freshwater fish species and their environment [3], mining the customer credit [4], modeling direct response behavior [5], building a decision-making framework for ozone pollution control [6], assessment of gully erosion susceptibility [7], estimating heating load in buildings [8], modeling daily dissolved oxygen concentration [9] etc.

Knot positioning is a time-consuming step in the MARS building processing, and it highly affects the accuracy of the final MARS model. The situation can getting worse for high dimensional regression model. In this research, it is desirable to reduce the computational cost of knot positioning, while maintaining an accurate model. Friedman [1] used all values from the predictive variables as candidate knot locations and used a greedy algorithm to select specific knot positions among all the candidates. The knot position that provides the greatest improvement in the residual sum of squares (RSS) was selected in each iteration of the MARS algorithm. Selecting a knot position from all possible data values is time-consuming when the data set is large. Chen et al. [10] used a fixed number of candidate knots that was a subset of the data values, such that the candidate knots were equally spaced. It is also possible to select a subset of the data values, such that candidate knots have the minimum number of data values between them (referred to as MinSpan in the R code “earth”), which can speed



(a) Training dataset (b) Model 1: R^2=0.94 (c) Model 2: R^2=0.99

Figure 1: MARS model with different knot positions

up the knot positioning process, but may miss some potentially superior knot positions that could achieve a more accurate MARS model. Koc and Iyigun [11] introduced a mapping approach to use more representative data points as candidate knots in the MARS knot positioning process. This approach can yield efficiency in knot positioning when the data are not evenly scattered over the input space. Miyata and Shen [12] proposed knot optimization using an evolutionary algorithm. Their approach can be generally applied for various forms of spline basis functions, but was only demonstrated for one input dimension and required additional computational effort compared to existing approaches.

If knots are not positioned well, the MARS model may not represent the relationships properly because basis functions will only bend at these positions. Suppose we fit two input dimensions, as illustrated in Figure 1a, where x_1 and x_2 are the input variables, and the surface has multiple peaks and valleys. MARS models with different knot positions are shown in Figures 1b and 1c. MARS model 1 achieved a coefficient of determination of $R^2 = 0.94$, and MARS model 2 achieved $R^2 = 0.99$, which indicates that MARS model 2 is better fit to the data than MARS model 1, as can be seen visually in the figures. Hence, limiting the set of candidate knots can degrade the model fit; however, an exhaustive search of knot positions is computational expensive.

In this research, we propose improved knot positioning mechanisms during the MARS building process. We propose two new methods for MARS knot positioning that seek to reduce the computational effort of knot positioning without degrading the quality of fit. We refer to these methods as the hill climbing method (HCM) and the hill climbing with prior information (PHCM) where the objective is to decrease the RSS. Numerical experiments using different dataset sizes and different numbers of candidate knots with different noise levels are investigated in this paper.

The rest of the paper are organized as follows. In Section 2, the original MARS algorithm is introduced. Section 3 provides the description of the datasets. In Section 4, the knot optimization for MARS using hill climbing methods is described in detail. Section 5 presents the experimental results, and, finally, concluding remarks are given in Section 6.

2. MARS background

MARS is introduced for the regression setting with multiple input variables and a response variable. In MARS model, the approximated MARS function is composed from a linear model of basis functions, which is defined from hinge functions or multiplication of hinge functions. The MARS model can be written as follows:

$$\hat{f}(\mathbf{x}) = \sum_{m=0}^{M} \left\{ a_m \cdot B_m(\mathbf{x}) \right\}, \tag{1}$$

where $\hat{f}(\mathbf{x})$ is the MARS model and $B_m(\mathbf{x})$ is called the basis function. Here m denotes the index of the basis function and M indicates the total number of basis functions in the MARS model. The coefficient of m-th basis function is denoted as a_m and $\mathbf{x} \in \mathbb{R}^n$ denotes the predicting variable vector. MARS uses a product form for the basis function:

$$B_m(\mathbf{x}) = \prod_{k=1}^{K_m} b_{k,m}. \tag{2}$$

Here $b_{k,m}$ is the k-th univariate function in $B_m(\mathbf{x})$ and K_m denotes the total number of univariate functions in $B_m(\mathbf{x})$. When $K_m = 1$, then the basis function is univariate. Otherwise, K_m is the degree of the interaction term.

In each basis function, the refraction points are the knots for the basis function. The $b_{k,m}$ are truncated linear functions of the form:

$$b(x|t) = [+(x-t)]_+ = \max\{+(x-t), 0\}, \tag{3}$$

or

$$b(x|t) = [-(x-t)]_+ = \max\{-(x-t), 0\}, \tag{4}$$

where the location t is called **knot** for the basis function.

Let $\{\mathbf{x}_i, y_i\}_{i=1}^N$ represent a dataset, where $\mathbf{x}_i \in \mathbb{R}^n$ denotes the i-th data point in predicting variable dataset, and the i-th data point for the response variable is defined as y_i. The sample size is denoted as N and i is the index of the data point ($i = 1, 2, 3, \ldots, N$).

The residual sum of squares between the observed value and the predicted value, denoted as e, is defined as:

$$e = \frac{1}{N}\sum_{i=1}^{N}\left[y_i - \hat{f}(\mathbf{x}_i)\right]^2. \tag{5}$$

In general, a smaller e is considered to be a better fit to the data. In MARS, a penalty term with e is used to avoid overfitting, but e alone is used for selecting among knot positions within the MARS algorithm.

The MARS forward stepwise algorithm [1] using the truncated linear univariate basis function is given in Algorithm 1 where $\{\mathbf{x}_i, y_i\}_{i=1}^N$ is the input dataset and $M_{\max}$ is the maxmimum number of basis functions. In each MARS iteration, the algorithm seeks a pair of basis functions to add to its current set. Candidate basis functions can be new univariate terms or interaction terms that are split from the current set. The innermost loop of the algorithm (line 5-11) considers all possible knot positions for a univariate term or additional split of an interaction term. In line 1 of Algorithm 1, the MARS model starts with a constant. The current best residual sum of squares e^* is initialized to be ∞. From line 2 to line 17, it adds basis functions until M_{max} basis functions are added to the MARS model. From line 3 to line 13, the regression process tries to split on all already added basis functions. The set $\{v(k,m)\}_{k=1}^{K_m}$ is the variable index set of the basis function $B_m(\mathbf{x})$ and v denotes the variable index. For example, if

$$B_m(\mathbf{x}) = [+(x_1 - 0.2)]_+ \cdot [-(x_3 - 0.6)]_+, \tag{6}$$

then the set

$$\{v(k,m)\}_{k=1}^{K_m} = \{1,3\}. \tag{7}$$

The candidate knot set of the basis function $B_m(\mathbf{x})$ at v-th variable is denoted as $\{\mathbf{x}_{j,v}|B_m(\mathbf{x}_j) > 0\}_{j=1}^{N}$ and it consists of the v-th variable values of the data points which make the basis function positive. In line 7, the new e for MARS model with new basis functions is calculated . From line 8 to 10, e is compared with e^*. If e is less than $e*$, it indicates the new model is better, and we store the related information, $e*$, the index of the basis function m^*, the variable index v^* and the knot value t^*. In line 14 and line 15, two new basis functions are added to the MARS model.

Algorithm 1: MARS forward stepwise algorithm

```
    Input: $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$,$M_{max}$
    Result: MARS regression model $\hat{f}(\mathbf{x})$
 1  $B_1(\mathbf{x}) = 1, M = 1, \mathrm{e}^* = \infty$
 2  while $M < M_{max}$ do
 3      for $m = 1\, to\, M$ do
 4          for $v \notin \{v(k,m)\}_{k=1}^{K_m}$ do
 5              for $t \in \{\mathbf{x}_{j,v}|B_m(\mathbf{x}_j) > 0\}_{j=1}^{N}$ do
 6                  $\hat{f} = \sum_{i=1}^{M} a_i B_i(\mathbf{x}) + a_{M+1} B_m(\mathbf{x})[+(x_v - t)]_+ +$
                      $a_{M+2} B_m(\mathbf{x})[-(x_v - t)]_+$
 7                  $e = \min_{a_1,\ldots,a_{M+2}} e(\hat{f})$
 8                  if $e < e^*$ then
 9                      $e^* = e, m^* = m, v^* = v, t^* = t$
10                  end
11              end
12          end
13      end
14      $B_{M+1}(\mathbf{x}) = B_{m^*}(\mathbf{x})[+(x_{v^*} - t^*)]_+$
15      $B_{M+2}(\mathbf{x}) = B_{m^*}(\mathbf{x})[-(x_{v^*} - t^*)]_+$
16      $M = M + 2$
17  end
```

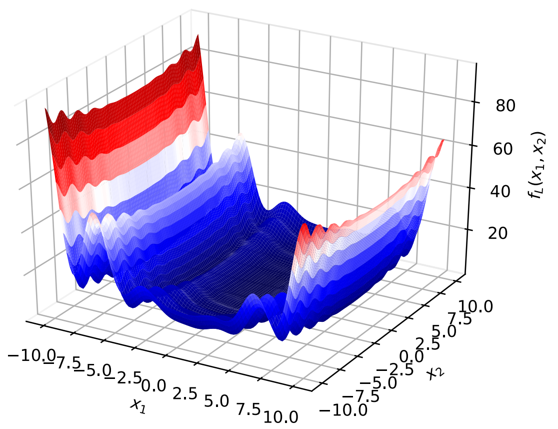


Figure 2: $f_L(x_1, x_2)$ $d = 2$

3. Datasets

In this paper, 7 datasets are used to investigate and verify the proposed new methods. The first 6 datasets are generated from 6 functions and the Sobol sequence is used to sample values in the input space [13]. The seventh dataset is a wind farm power distribution dataset [14].

The first dataset D_L is generated from the Levy function $f_L(\mathbf{x})$ [15] as

$$\begin{aligned} f_L(\mathbf{x}) =& \sin^2(\pi w_1) + \sum_{i=1}^{d-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2\left[1 + \sin^2(2\pi w_d)\right] \\ & w_i = 1 + \frac{x_i - 1}{4}, \text{ for all } i = 1, \ldots, d \\ & -10 \leqslant x_i \leqslant 10, \end{aligned} \tag{8}$$

where $\mathbf{x}$ is the independent variable. The dimension of $\mathbf{x}$ is denoted as d, and in this paper, $d = 30$ which indicates $\mathbf{x}$ is 30-dimensional. Figure 2 is the surface of Levy function when $d = 2$.

Datasets D_1, D_2, D_3, D_4 and D_5 are generated from functions f_1, f_2, f_3, f_4 and f_5 [11], respectively. In dataset D_1, $\mathbf{x}$ has 7 dimensions. In dataset D_2, $\mathbf{x}$ is 10-dimensional. For D_3, $\mathbf{x}$ has 10 dimensions and for D_4, $\mathbf{x}$ is 3-dimensional.

For D_5, $\mathbf{x}$ is 21-dimensional with $\boldsymbol{\alpha} = \{0.15,\ -0.96,\ 0.09,\ 0.84,\ 0.55,\ -0.58,\ 0.21,\ 0.50,\ 0.1,\ -0.90\}$ and $\mathbf{x}$ of D_6 is also 2-dimensional. Figure 3 shows the function surfaces when limiting the dimension to 2.

$$\begin{aligned} f_1(\mathbf{x}) =& \sum_{i=1}^{7} \left[\ln^2(x_i - 2) + \ln^2(10 - x_i)\right] - \left(\prod_{i=1}^{7} x_i\right)^2 \\ & 2.1 \leqslant x_i \leqslant 9.9,\ i = 1, 2, 3, \ldots, 7 \end{aligned} \tag{9}$$

$$\begin{aligned} f_2(\mathbf{x}) =& \sum_{j=1}^{10} \exp(x_j) \left(c_j + x_j - \ln \sum_{k=1}^{10} \exp(x_k)\right) \\ & \mathbf{c} = [-0.6089, -17.164, -34.054, -5.914, -24.721, -14.986, -24.100, -10.708, \\ & -26.662, -22.179] \\ & -10 \leqslant x_i \leqslant 10 \end{aligned} \tag{10}$$

$$\begin{aligned} f_3(\mathbf{x}) =& x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 - 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + 2(x_{10} - 7)^2 + 45 \\ & -10 \leqslant x_i \leqslant 10 \end{aligned} \tag{11}$$

$$\begin{aligned} f_4(\mathbf{x}) =& \sin\left(\frac{\pi x_1}{12}\right) \cos\left(\frac{\pi x_2}{16}\right) \\ & -10 \leqslant x_1 \leqslant 10, -20 \leqslant x_2 \leqslant 20 \end{aligned} \tag{12}$$

$$
\begin{aligned}
f_5(\mathbf{x},\mathbf{y}) =& \sum_{i=1}^{d} \alpha_i [3(1-x_i)^2 \exp(-x_i^2 - (y_i+1)^2) - 10(\frac{x}{5} - x_i^3 - y_i^5) \exp(-x_i^2 - y_i^2) \\
&- \frac{1}{3} \exp(-(x_i+1)^2 - y_i^2) + 2x_i], \\
&\sum_{i=1}^{d} \alpha_i = 1, \text{for all } i = 1, \ldots, d, \\
&-2 \leqslant x_i \leqslant 2, -2 \leqslant y_i \leqslant 2 \qquad (13)
\end{aligned}
$$

In the experiments, we added Gaussian noise with different levels (5%, 10%, and 20%) to the datasets to investigate and verify the robustness of the proposed methods HCM and PHCM. The signal-to-noise ratio (SNR) is defined as

$$
\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} \qquad (14)
$$

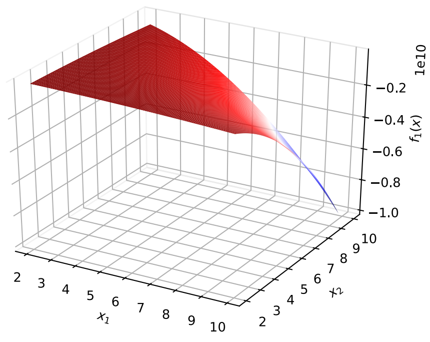
where P_{singal} is the average power of the signal and P_{noise} is the average power of the noise [16]. Figure 4 and Figure 5 show D_4 and D_6 with different noise levels.

4. Knot optimization for MARS using hill climbing methods

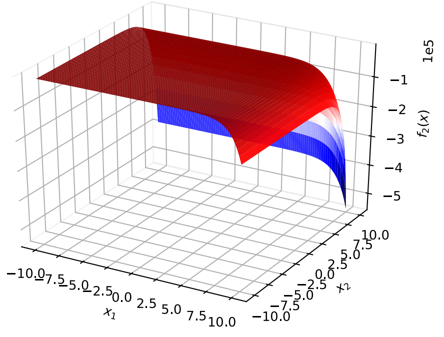
In our proposed MARS knot positioning process, we define the change in RSS as the objective function, given as

$$
\Delta e = e_p - e_c,
$$

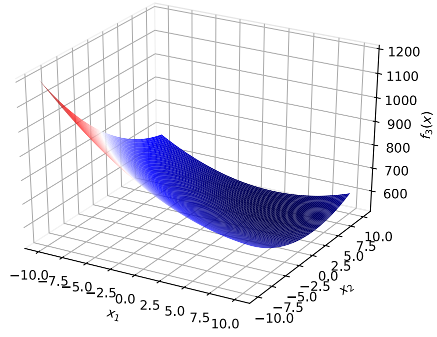
where e_p and e_c are the RSS values of the prior iteration and the current iteration, respectively. If Δe is negative, it indicates that the current MARS model is less accurate than the prior MARS model. If Δe is positive, it indicates that the current MARS model is more accurate and is an improvement over the prior MARS model. The larger the value of Δe, the more accurate the current MARS model is. Hence, we seek to maximize Δe to find the best fitting MARS model under current settings (adding one knot to the current MARS model).



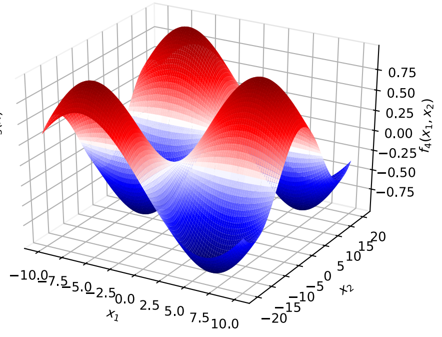
(a) $f_1(x_1, x_2)$ other $x_i = 4.0$



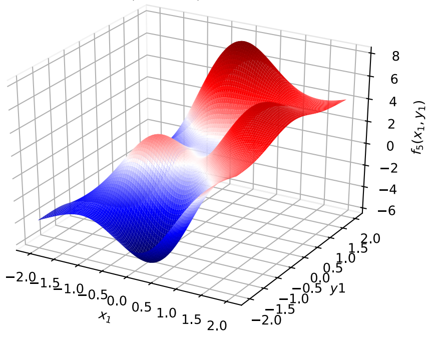
(b) $f_2(x_1, x_2)$ other $x_i = 3.0$



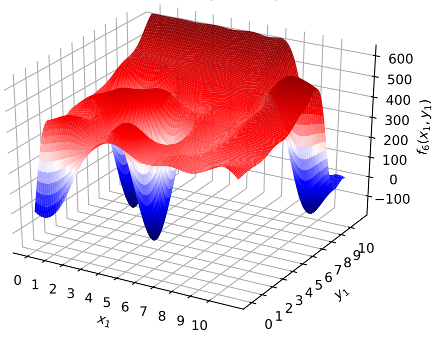
(c) $f_3(x_1, x_2)$ other $x_i = 4.0$



(d) $f_4(x_1, x_2)$

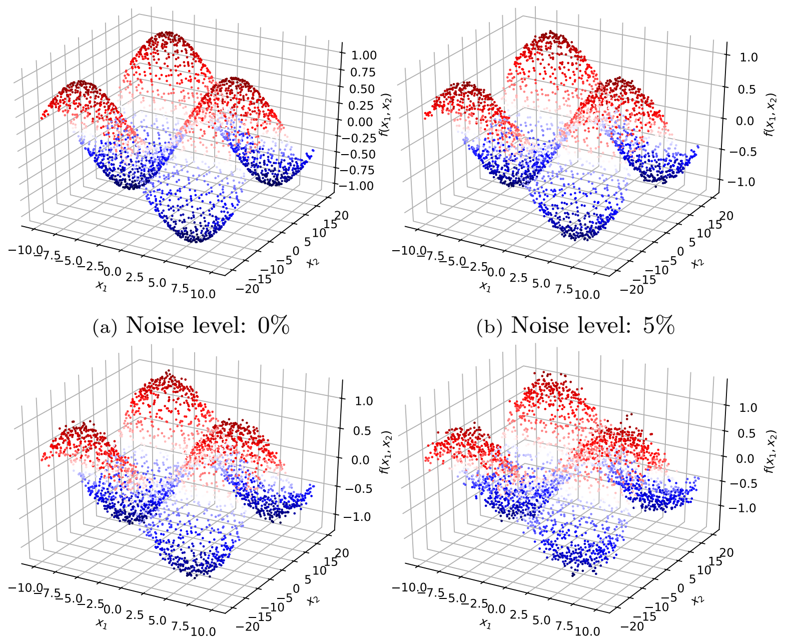


(e) $f_5(x_1, y_1)$ $d = 1$



(f) $f_6(x_1, y_1)$ $d = 1$

Figure 3: Surfaces of dataset functions



(a) Noise level: 0% (b) Noise level: 5%

(c) Noise level: 10% (d) Noise level: 20%

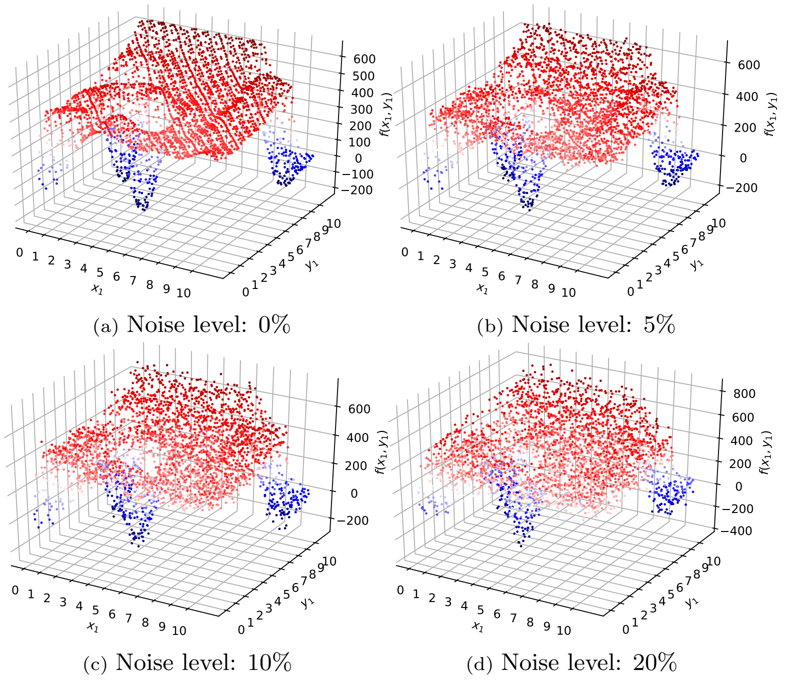
Figure 4: D_4 with different levels of noise

4.1. Exploring the change in residual sum of squares function

In the MARS knot positioning process, when we use Δe, lines 8 to 10 in in Algorithm 1 will become:

```
8  $\Delta e = e^* - e$
9  **if** $\Delta e > 0$ **then**
10     $e^* = e, m^* = m, v^* = v, t^* = t$
11 **end**
```

The Δe will be calculated repeatedly for different basis functions to choose the knot with the largest Δe value. Figure 6 shows Δe functions of variable x_1 in MARS from subsequent iterations on a representative dataset D_L. Assume x_1 is the variable that we are considering in creating the next basis function $B_m(\mathbf{x})$. Figure 6a is generated when there are no basis functions in the MARS model. Figures 6b and 6e are generated when there are already two and four basis functions, respectively, in the MARS model. From Figure 6a to Figure 6e,



(a) Noise level: 0% (b) Noise level: 5%

(c) Noise level: 10% (d) Noise level: 20%

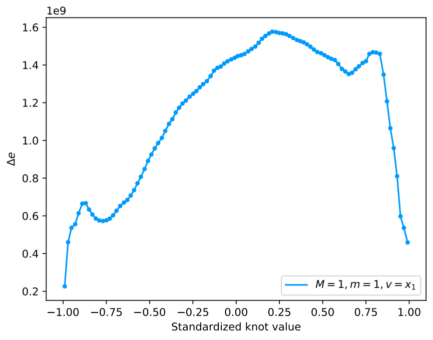
Figure 5: D_6 with different levels of noise

the three local maxima move only a little, and the global maximum is almost the same position, which is around 0.25. The principle that the local maxima of Δe functions move very little from iteration to iteration also applies to other cases. We refer to these local maxima as *key knots*, and we will use this principle in our new knot positioning methods.

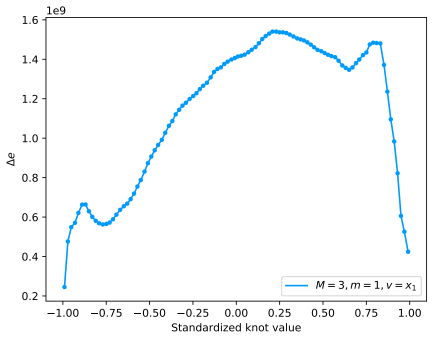
4.2. Hill climbing method

In this section, we introduce the hill climbing method for MARS knot positioning [17]. If a function is concave, then hill climbing will find a global maximum, if one exists. However, the Δe function may not be concave, so we require multiple starting points to get closer to the global maximum, as shown in Figure 7.

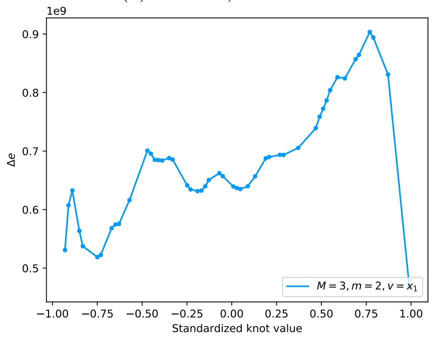
Figure 7 shows how the hill climbing method works on an example Δe function, where the vertical axis is the Δe value and the horizontal axis is the knot value. Suppose S_0, S_1, and S_2 are three candidate knot positions that are arbi-



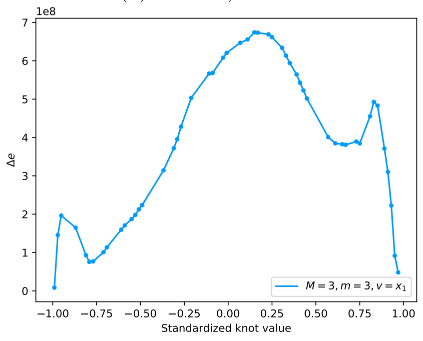
(a) $M = 1, m = 1$



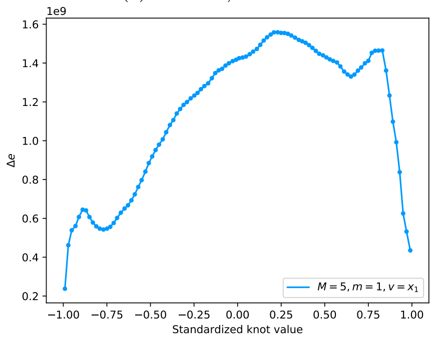
(b) $M = 3, m = 1$



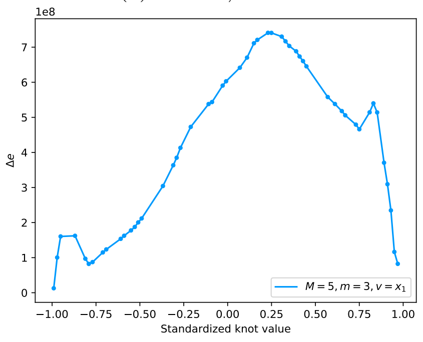
(c) $M = 3, m = 2$



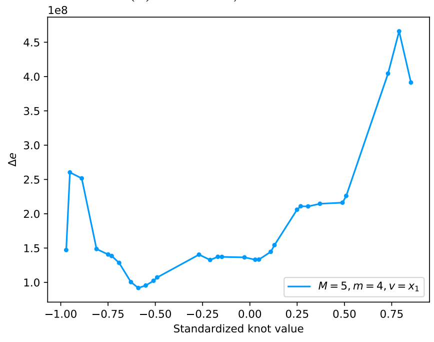
(d) $M = 3, m = 3$



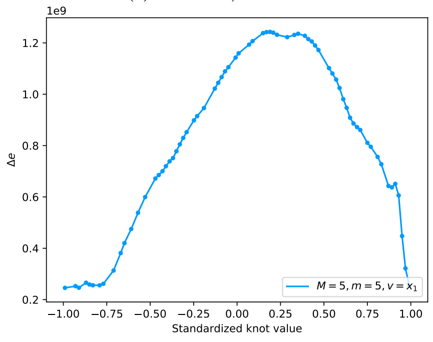
(e) $M = 5, m = 1$



(f) $M = 5, m = 3$



(g) $M = 5, m = 4$



(h) $M = 5, m = 5$

Figure 6: Explore the change in the residual sum of squares (Δe) function for $v = x_1$

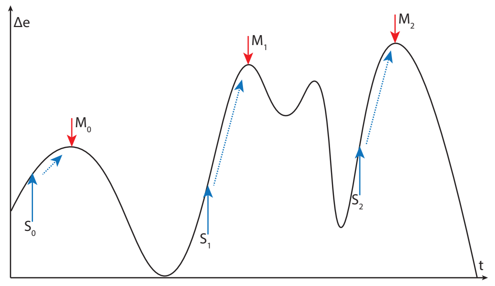


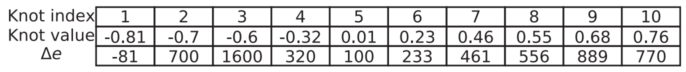
Figure 7: Illustration of the hill climbing method

trarily chosen from the candidate knot set (line 5 in Algorithm 1). If we start from S_0, S_1, and S_2 and try to maximize Δe, then we will end with knot values M_0, M_1, and M_2, respectively. Only the knot values in $[S_0,\ M_0]$, $[S_1,\ M_1]$ and $[S_2,\ M_2]$ will be traversed, and the other knot values in the domain of Δe will be ignored, so using hill climbing methods will speed up the knot positioning process by reducing the search process.

The starting points play an important role in the hill climbing method, which heavily affects the convergence speed and the last achieved optimum value. If the starting points are very close to a local maximum, the optimization process will end up at a local optimum, as shown in Figure 7 where we are trying to maximize Δe. Fortunately, by exploring Δe functions of different datasets, we find that the current key knots move a little from the prior key knots. Intuitively, we can use the key knots from a prior iteration as the starting points of the current iteration. By doing experiments on different datasets, we find that it works the same way on other datasets. This pattern of the key knots' changes can be helpful when a basis function is added using that x-variable.

*4.3. Hill climbing method **without using** prior change in RSS information for MARS knot positioning*

The first new method we propose for MARS knot positioning is a general hill climbing method with multiple starting points, called HCM. The HCM algorithm starts with multiple starting points and converges to the local maxima of Δe. Then the knot with the largest Δe from the local maxima is chosen as



Knot index	1	2	3	4	5	6	7	8	9	10
Knot value	-0.81	-0.7	-0.6	-0.32	0.01	0.23	0.46	0.55	0.68	0.76
Δe	-81	700	1600	320	100	233	461	556	889	770

Figure 8: Illustration of candidate knots

the new knot to be added to the MARS model.

Algorithm 2 shows the HCM knot positioning algorithm. The initial step in line 2 sorts the candidate knots in ascending order, and the knots are referenced by their ordered knot index. An positive integer step size r is defined to increment the knot index, which allows the algorithm to traverse the candidate knots. As recommended by Friedman [1], candidate knots are located only at data values. Figure 8 is an illustration of candidate knots for x_1. If the current knot index is 4 (knot value, -0.32) and the next knot index is 6 (knot value, 0.23), then $r = |6 - 4| = 2$. When r takes a large number, the knot positioning process will converge fast, but it is not stable because it may skip and miss an optimal knot. When r takes a small number, the knot selection process will converge slowly but is stable. Line 5 in Algorithm 2 follows the original MARS algorithm to define the potential candidate knot set for $B_m(\mathbf{x}_j)$ for the v-th input variable. In Algorithm 2, we refer to this set as Φ. Let $\Phi = \{-0.81, -0.7, -0.6, -0.32, 0.01, 0.23, 0.46, 0.55, 0.68, 0.76\}$ as shown in Figure 8. The starting knot set is $\Phi_{\mathbf{S}}$, where $\mathbf{S}$ is the knot index set of the starting knots, and we generate $\Phi_{\mathbf{S}}$ by taking equally indexed knots for a given starting knot number. As shown in Figure 8, if the starting knot number is 3, then $\mathbf{S}$ can be $\{1, 5, 9\}$ and $\Phi_{\mathbf{S}}$ is $\{-0.81, 0.01, 0.68\}$.

Lines 8 to 30 conduct HCM, which starts from each starting knot value in $\Phi_{\mathbf{S}}$. Line 9 obtains a starting knot value t_s, and line 10 calculates the new MARS model with two new basis functions by using the new knot value t_s. Line 11 calculates the e_s value, where e_s is the RSS value for the new MARS model by using knot value t_s. From lines 12 to 20, knots are traversed to the left of the starting knot, while from lines 21 to 29, knots are traversed to the right of the starting knot. As illustrated in Figure 8, if t_s is 0.01, the knots to the left are $\{-0.7, -0.6, -0.32\}$ and the knots to the right are $\{0.23, 0.46, 0.55\}$.

In line 12, when traversing knots to the left of t_s, the knot index s_- is initialized to s, and the current e for knot $\Phi[s_-]$ is e_c. In line 14, the e value e_c for the prior knot becomes the prior e value e_P for the current knot. Line 15 updates the information on the best knot. Line 16 moves the current knot index to the left by r and updates the current knot value t to $\Phi[s_-]$. Lines 17 and 18 update the MARS model with the current knot and calculate the e value e_c for the current knot. Line 19 calculates the decrease in e value Δe. If Δe is greater than a predefined small positive scalar ϵ, the current knot index will move to the left by r and repeat line 14 to 19 again. Otherwise, the algorithm will stop traversing to the left and will begin traversing to the right of the starting knot $\Phi[s]$, and in this case, the process will skip a part of knots and save time. Lines 22 to 29 traverse knots to the right of the starting knot $\Phi[s]$. The knot sets Φ_S divides the whole searching space into intervals, and if a knot t has already been traversed, the current search stops.

*4.4. Hill climbing method **using** prior change in RSS information for MARS knot positioning*

In HCM, all candidate knots are equally likely to be chosen for the starting point set. As was shown earlier in Figure 6 shows that the local optima do not move much from iteration to iteration and the local optima should be considered with much higher priority [18]. The second new method is a hill climbing method using the prior Δe information (PHCM), where the starting point set consists of the key knots from the prior iteration. By exploring the Δe function in Section 4.2, we find that local maxima from the prior iteration are usually near those of the current iteration, so we expect PHCM to converge faster than HCM.

The difference between HCM and PHCM is how to determine $\Phi_{\mathbf{S}}$. In HCM, $\Phi_{\mathbf{S}}$ is generated by taking equally indexed knots for all iterations in Algorithm 2 line 7. In PHCM, for the first iteration, all candidate knots are chosen as $\Phi_{\mathbf{S}}$, and for the other iterations, $\Phi_{\mathbf{S}}$ is the local maxima set of Δe identified in the

Algorithm 2: HCM method for MARS knot positioning

```
Data: $\mathbf{x}, y, M\mathrm{max}, \epsilon, r$
Result: MARS regression model $\hat{f}(\mathbf{x})$
1  $B_1(\mathbf{x}) = 1, M = 1, e^* = \infty$
2  sort $\{\mathbf{x}_{j,v}\} \rightarrow \{\mathbf{x}_{(j),v}\}$ ascending
3  while $M < M_{max}$ do
4      for $m = 1\, to\, M$ do
5          for $v \notin \{v(k,m)\}_{k=1}^{K_m}$ do
6              $\Phi = \{\mathbf{x}_{j,v} | B_m(\mathbf{x}_j) > 0\}_{j=1}^{N}$
7              random $\Phi_{\mathbf{S}} \subseteq \Phi$
8              foreach $t_s \in \Phi_{\mathbf{S}}$ do
9                  $t = t_s \ (t_s = \Phi[s])$
10                 $\hat{f} = \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_{M+1} B_m(\mathbf{x})[+(x_v - t)]_+ +$
                    $a_{M+2} B_m(\mathbf{x})[-(x_v - t)]_+$
11                 $e_s = \min_{a_1,\ldots,a_{M+2}} e(\hat{f})$
12                 $s_- = s, e_c = e_s$
13                 do
14                     $e_p = e_c$
15                     if $e_p < e^*$ then $e^* = e_p, m^* = m, v^* = v, t^* = t$
16                     $s_- = s_- - r, t = \Phi[s_-]$
17                     $\hat{f} = \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_{M+1} B_m(\mathbf{x})[+(x_v - t)]_+ +$
                        $a_{M+2} B_m(\mathbf{x})[-(x_v - t)]_+$
18                     $e_c = \min_{a_1,\ldots,a_{M+2}} e(\hat{f})$
19                     $\Delta e = e_p - e_c$
20                 while $\Delta e > \epsilon$
21                 $t = t_s, s_+ = s, e_c = e_{t_s}$
22                 do
23                     $e_p = e_c$
24                     if $e_p < e^*$ then $e^* = e_p, m^* = m, v^* = v, t^* = t$
25                     $s_+ = s_+ + r, t = \Phi[s_+]$
26                     $\hat{f} = \sum_{i=1}^{M-1} a_i B_i(\mathbf{x}) + a_{M+1} B_m(\mathbf{x})[+(x_v - t)]_+ +$
                        $a_{M+2} B_m(\mathbf{x})[-(x_v - t)]_+$
27                     $e_c = \min_{a_1,\ldots,a_{M+2}} e(\hat{f})$
28                     $\Delta e = e_p - e_c$
29                 while $\Delta e > \epsilon$
30             end
31         end
32     end
33     $B_{M+1}(\mathbf{x}) = B_{m^*}(\mathbf{x})[+(x_{v^*} - t^*)]_+$
34     $B_{M+2}(\mathbf{x}) = B_{m^*}(\mathbf{x})[-(x_{v^*} - t^*)]_+$
35     $M = M + 2$
36 end
```

prior iteration. For example as shown in Figure 8, in the first iteration for x_1,

$$\begin{aligned} &\mathbf{S} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}, \\ &\Phi_{\mathbf{S}} = \{-0.81, -0.7, -0.6, -0.32, 0.01, 0.23, 0.46, 0.55, 0.68, 0.76\}, \end{aligned} \tag{15}$$

and for the second iteration for x_1,

$$\begin{aligned} &\mathbf{S} = \{3, 9\}, \\ &\Phi_{\mathbf{S}} = \{-0.6, 0.68\}. \end{aligned} \tag{16}$$

In this way, PHCM converges faster to the local maxima, so PHCM has superiority in dealing large datasets.

5. Experiments and results

In this section, we test the MARS knot selection method from Chen et al. [10] and our new methods, HCM and PHCM, on different datasets with varying noise levels.

5.1. Exploration of candidate knot numbers

In this section, the MARS knot selection method from Chen et al. [10] (CM), HCM, and PHCM methods are applied to six datasets under different candidate knot number settings, 10, 30, 50, 100, 200, 500 and 1000. Table 1 summarizes the training and testing R^2 results on dataset D_1 under different candidate knot number settings, 10, 30, 50, 100, 200, 500 and 1000. Let R^2_P, R^2_H, and R^2_C be the coefficients of determination for the PHCM method, the HCM method, and the knot positioning method from Chen et al. [10], respectively, where a higher R^2 indicated a better fit to the data. The number of candidate knot number is denoted as N_k.

From Table 1, we can see that as the candidate knot number increases, the R^2 value is going up. The table also shows there is no significant R^2 difference between training and testing dataset, so overfitting is not a problem. However,

Table 1: R^2 comparison on dataset D_1 over different candidate knot numbers: training vs testing

Noise	N_k		10	30	50	100	200	500	1000
0%	Train	R_C^2	0.769	0.809	0.860	0.871	0.884	0.920	0.941
		R_H^2	0.781	0.799	0.846	0.860	0.882	0.912	0.938
		R_P^2	0.769	0.805	0.860	0.870	0.884	0.920	0.940
	Test	R_C^2	0.738	0.780	0.832	0.840	0.855	0.892	0.913
		R_H^2	0.752	0.769	0.817	0.829	0.852	0.883	0.910
		R_P^2	0.738	0.775	0.832	0.841	0.855	0.892	0.914
5%	Train	R_C^2	0.769	0.825	0.834	0.854	0.880	0.895	0.909
		R_H^2	0.760	0.814	0.830	0.842	0.862	0.875	0.902
		R_P^2	0.769	0.816	0.832	0.852	0.880	0.895	0.906
	Test	R_C^2	0.742	0.799	0.808	0.829	0.856	0.872	0.886
		R_F^2	0.734	0.789	0.806	0.817	0.838	0.851	0.877
		R_P^2	0.742	0.790	0.807	0.827	0.856	0.872	0.883
10%	Train	R_C^2	0.769	0.816	0.831	0.843	0.853	0.869	0.891
		R_H^2	0.766	0.813	0.825	0.836	0.847	0.861	0.875
		R_P^2	0.769	0.804	0.831	0.838	0.853	0.869	0.890
	Test	R_C^2	0.750	0.794	0.812	0.824	0.834	0.851	0.873
		R_F^2	0.746	0.790	0.805	0.816	0.827	0.842	0.856
		R_P^2	0.750	0.785	0.812	0.819	0.834	0.851	0.870
20%	Train	R_C^2	0.761	0.792	0.802	0.825	0.834	0.860	0.887
		R_F^2	0.759	0.786	0.800	0.816	0.830	0.857	0.884
		R_P^2	0.761	0.792	0.802	0.822	0.834	0.860	0.887
	Test	R_C^2	0.744	0.776	0.794	0.808	0.817	0.844	0.872
		R_F^2	0.741	0.768	0.793	0.798	0.814	0.840	0.870
		R_P^2	0.744	0.776	0.794	0.805	0.817	0.844	0.872

the computational time is also going up with the candidate knot number increasing. Under the same candidate knot number settings, the R^2 values for the CM method, the HCM method, and the PHCM method are almost the same.

Let T_C denote the computational time for CM method, T_H for HCM method and T_P for PHCM method. The compuational time ratio of three methods are defined as follows:

$$\text{Computation time ratio of CM} = \frac{T_C}{T_C} = 1 \tag{17}$$

$$\text{Computation time ratio of HCM} = \frac{T_H}{T_C} \tag{18}$$

$$\text{Computation time ratio of PHCM} = \frac{T_P}{T_C}. \tag{19}$$

The computational time of CM method is the benchmark. If the computational time is less than 1, it implies that the methods uses less computational time than CM method.

Figures 9, 10 and 11 summarize the computational time ratios of three methods on datasets D_6, D_1 and D_5 under different candidate knot number settings. The input variable $\boldsymbol{x}$ for D_6 is 2 dimensional, $\boldsymbol{x}$ for D_1 is 7 dimensional and $\boldsymbol{x}$ for D_5 is 20 dimensional. Under most cases, the HCM and PHCM methods used less computational time than CM method. The ratio of HCM over different candidate knot numbers remained relatively stable compared to the ratio of PHCM, around 0.60 to 0.70, which indicates only 60% to 70% of the computational time of CM method was used in HCM method. The ratio of PHCM on D_1 dropped dramatically with increasing candidate knot numbers from 0.80 to 0.25, which indicates that the PHCM method used about 80% of the computational time of CM method when the candidate knot number was 10, and 25% of the CM computational time when the candidate knot number was 1000. The ratio for PHCM is going down when the candidate knot number increases, which means PHCM is more computationally efficient when dealing with a large size dataset. The figures also show that the proposed methods HCM and PHCM can perform very well with different levels of noise.

5.2. Exploration of different datasets

In this section, we tested our methods on six different datasets with the candidate knot number setting being 1000. The CM method, the HCM method, and the PHCM method are used on these six datasets.

Table 2 summarizes the R^2 values of three methods on six different datasets under four levels of noises. Under the same settings, the final achieved R^2 are almost the same. It also shows there is little difference in R^2 between the training set and the testing set, so no overfitting is again not a problem..

Figure 12 is the comparison of the computational time ratios of three methods on six different datasets. Comparing datasets with different dimensions, we saw that for datasets with 7, 10, 10, and 20 dimensions, the PHCM method has dramatically lower computational time ratio than HCM method. For datasets with 2 dimensions, the differences in the ratio are not as dramatic as those for

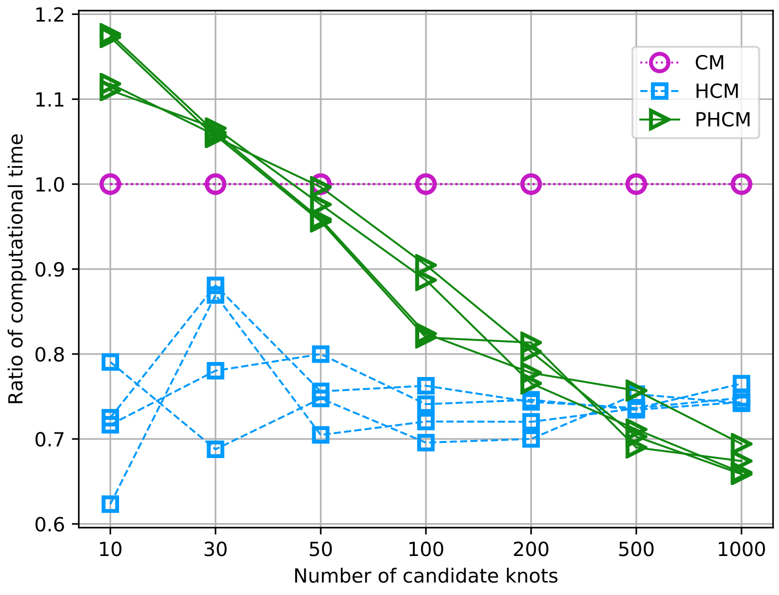


Figure 9: Computational time ratios of three methods on D_6 under different knot number settings with four noise levels: $\boldsymbol{x}$ is 2 dimensional

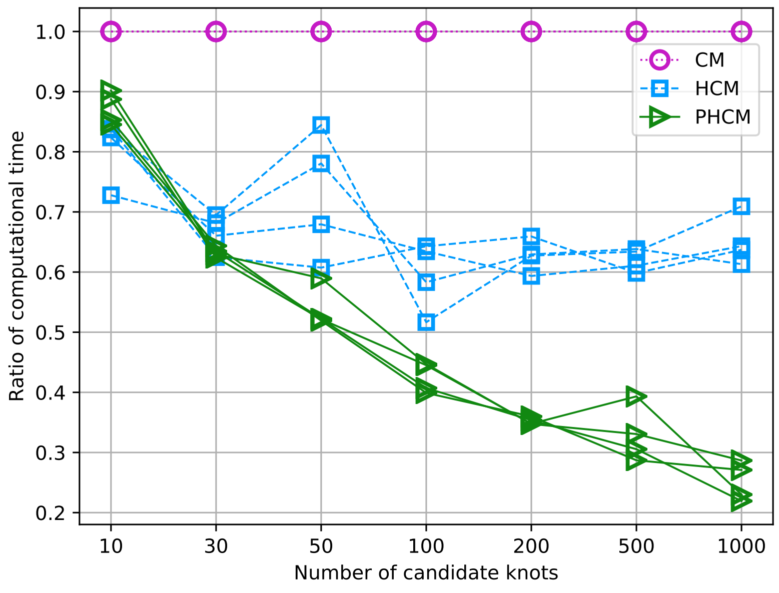


Figure 10: Computational time ratio of three methods on D_1 under different knot number settings with four noise levels: $\boldsymbol{x}$ is 7 dimensional

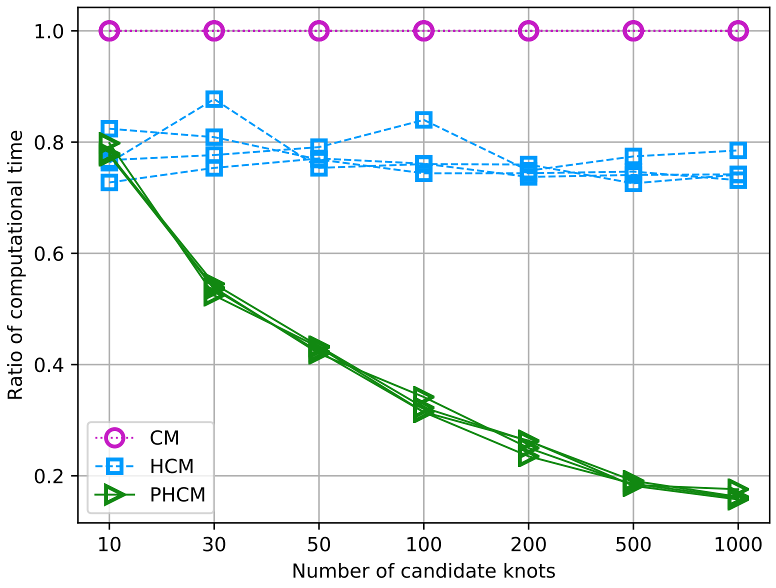


Figure 11: Computational time ratios of three methods on D_5 under different knot number settings with four noise levels: $\boldsymbol{x}$ is 20 dimensional

Table 2: R^2 comparison on six different datasets: training vs testing

Noise	$f(\mathbf{x})$		D_1	D_2	D_3	D_4	D_5	D_6
	N_k		1000	1000	1000	1000	1000	1000
0%	Train	R_O^2	0.941	0.999	0.999	0.999	0.944	0.971
		R_H^2	0.938	0.999	0.999	0.999	0.945	0.971
		R_P^2	0.940	0.999	0.999	0.999	0.943	0.971
	Test	R_O^2	0.913	0.994	0.999	0.998	0.940	0.946
		R_H^2	0.910	0.994	0.999	0.998	0.936	0.946
		R_P^2	0.914	0.999	0.999	0.998	0.932	0.946
5%	Train	R_O^2	0.909	0.995	0.993	0.996	0.935	0.952
		R_H^2	0.902	0.995	0.993	0.996	0.935	0.953
		R_P^2	0.906	0.995	0.993	0.996	0.940	0.952
	Test	R_O^2	0.886	0.991	0.992	0.995	0.940	0.929
		R_H^2	0.877	0.991	0.999	0.995	0.940	0.927
		R_P^2	0.883	0.998	0.999	0.995	0.938	0.929
10%	Train	R_O^2	0.891	0.985	0.973	0.991	0.933	0.906
		R_H^2	0.875	0.985	0.973	0.991	0.933	0.906
		R_P^2	0.890	0.985	0.973	0.991	0.933	0.906
	Test	R_O^2	0.873	0.989	0.953	0.997	0.918	0.897
		R_H^2	0.856	0.989	0.953	0.997	0.918	0.897
		R_P^2	0.870	0.989	0.949	0.997	0.930	0.890
20%	Train	R_O^2	0.887	0.948	0.898	0.965	0.904	0.794
		R_H^2	0.884	0.948	0.898	0.965	0.904	0.794
		R_P^2	0.887	0.948	0.898	0.965	0.903	0.794
	Test	R_O^2	0.872	0.955	0.895	0.950	0.909	0.778
		R_H^2	0.870	0.955	0.896	0.950	0.908	0.778
		R_P^2	0.872	0.951	0.897	0.950	0.910	0.778

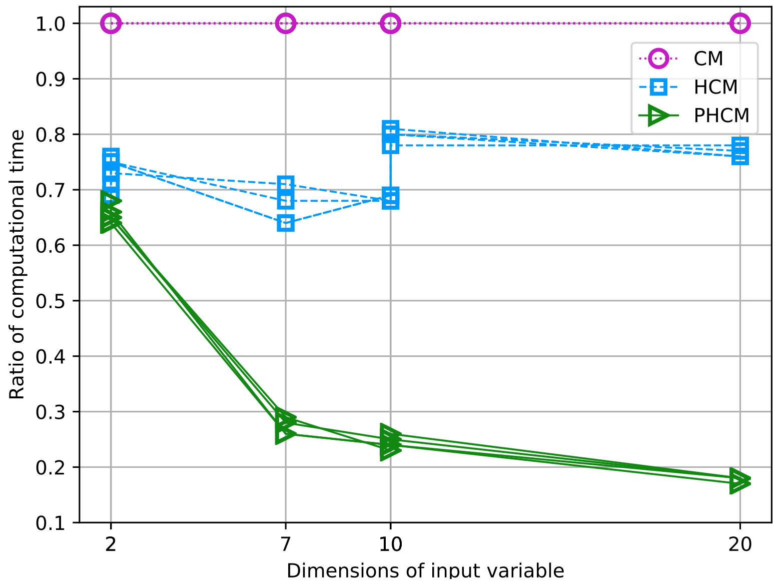


Figure 12: Computational time ratios of three methods on six datasets with different dimensions under four noise levels

high dimensional datasets. The phenomenon indicates that PHCM should be the preferred method for high-dimensional data.

6. Conclusion

In this paper, we proposed two new methods for MARS knot positioning, the hill climbing method (HCM) and the hill climbing method using key knots. The HCM and PHCM achieved a reduction in computational time compared to CM, while maintaining similar quality of fit. The PHCM achieved the most significant savings with over 80% reduction in computational time for the higher-dimensional data sets. By using different datasets with different noise levels, we show that PHCM and HCM are robust dealing with noisy datasets.

Acknowledgement

This research is partially funded by National Science Foundation grant CMMI-1434401.

Appendix A. Other results tables and charts

Table A.3 summarizes the training results on dataset D_1 under different candidate knot number settings, 10, 30, 50, 100, 200, 500 and 1000. Define N_k to be the number of candidate knots. Let N_C be the total number of knots in which Δe was calculated using the CM, N_H be that using the HCM, and N_P be that using the PHCM, where fewer calculated Δe values usually result in a lower computational time. Let R_P be the ratio of N_P to N_C, and R_H be the ratio of N_H to N_C, where a lower ratio indicates lower computational effort. Let R_P^2, R_H^2, and R_C^2 be the coefficients of determination for the PHCM, the HCM, and the CM, respectively, where a higher R^2 indicated a better fit to the data. Let T be the computational time in seconds of the MARS algorithm with T_P for the PHCM, T_H for the HCM, and T_C for the CM.

We also tested our methods on six different datasets with the candidate knot number setting being 1000. The CM method, the HCM method, and the PPHCM method are used on these six datasets.

Table A.4 summarizes the training results on six different datasets with the candidate knot number setting being 1000.

Table A.3: Comparison of three methods on dataset D_1 over different candidate knot numbers: training results

Noise	N_k	10	30	50	100	200	500	1000
0%	N_C	17,197	53,675	93,610	181,336	370,894	1,086,174	2,090,062
	N_H	12,521	33,630	58,586	119,062	242,323	656,259	1,339,374
	N_P	13,032	31,134	46,381	69,180	130,477	320,969	590,287
	R_H	0.73	0.63	0.63	0.66	0.65	0.60	0.64
	R_P	0.76	0.58	0.50	0.38	0.35	0.30	0.28
	R^2_C	0.769	0.809	0.860	0.871	0.884	0.920	0.941
	R^2_H	0.781	0.799	0.846	0.860	0.882	0.912	0.938
	R^2_P	0.769	0.805	0.860	0.870	0.884	0.920	0.940
	R^2_{aC}	0.744	0.790	0.845	0.853	0.873	0.913	0.936
	R^2_{aH}	0.757	0.778	0.830	0.843	0.872	0.905	0.932
	R^2_{aP}	0.744	0.785	0.845	0.842	0.873	0.913	0.929
	T_C	3.40	7.94	13.27	24.36	46.60	146.38	259.29
	T_H	2.85	4.96	8.06	15.66	30.72	87.61	165.16
	T_P	2.90	5.11	6.89	9.72	16.77	42.01	70.25
5%	N_C	17,282	56,015	86,755	192,700	348,157	869,325	1,988,368
	N_H	12,610	35,746	56,893	125,606	218,216	535,707	1,266,933
	N_P	13,110	31,609	42,310	82,831	121,907	257,309	518,964
	R_H	0.73	0.64	0.66	0.65	0.63	0.62	0.64
	R_P	0.76	0.56	0.49	0.43	0.35	0.30	0.26
	R^2_C	0.769	0.825	0.834	0.854	0.880	0.895	0.909
	R^2_H	0.760	0.814	0.830	0.842	0.862	0.875	0.902
	R^2_P	0.769	0.816	0.832	0.852	0.880	0.895	0.906
	R^2_{aC}	0.743	0.807	0.818	0.835	0.868	0.875	0.900
	R^2_{aH}	0.740	0.794	0.811	0.831	0.846	0.865	0.881
	R^2_{aP}	0.743	0.801	0.815	0.833	0.865	0.874	0.897
	T_C	3.46	8.51	11.54	26.94	45.66	102.34	259.35
	T_H	2.85	5.62	7.84	17.09	27.10	62.49	166.78
	T_P	3.07	5.31	6.02	11.99	15.88	40.25	59.66
10%	N_C	17,222	52,221	85,520	188,521	381,650	879,351	1,336,330
	N_H	13,084	33,790	53,963	113,931	253,330	585,591	909,954
	N_P	12,965	29,753	42,885	78,657	136,553	275,360	341,559
	R_H	0.76	0.65	0.63	0.60	0.66	0.67	0.68
	R_P	0.75	0.57	0.50	0.42	0.36	0.31	0.26
	R^2_C	0.769	0.816	0.831	0.843	0.853	0.869	0.891
	R^2_H	0.766	0.813	0.825	0.836	0.847	0.861	0.875
	R^2_P	0.769	0.804	0.831	0.838	0.853	0.869	0.890
	R^2_{aC}	0.744	0.796	0.814	0.826	0.838	0.857	0.880
	R^2_{aH}	0.741	0.793	0.811	0.821	0.833	0.852	0.861
	R^2_{aP}	0.744	0.784	0.814	0.823	0.838	0.857	0.878
	T_C	3.49	7.72	11.85	26.70	51.18	110.18	253.85
	T_H	2.54	5.26	9.25	15.57	32.21	70.33	155.65
	T_P	2.95	4.90	6.19	10.87	18.10	33.66	55.67
20%	N_C	17,762	50,280	84,420	172,248	354,203	799,428	1,755,070
	N_H	12,856	34,745	61,390	94,171	237,130	497,342	1,246,389
	N_P	13,976	29,537	43,982	74,961	128,663	254,856	514,569
	R_H	0.72	0.69	0.73	0.55	0.67	0.62	0.71
	R_P	0.79	0.59	0.26	0.44	0.36	0.32	0.29
	R^2_C	0.761	0.792	0.802	0.825	0.834	0.860	0.887
	R^2_H	0.759	0.786	0.800	0.816	0.830	0.857	0.884
	R^2_P	0.761	0.792	0.802	0.822	0.834	0.860	0.887
	R^2_{aC}	0.735	0.771	0.790	0.805	0.818	0.848	0.877
	R^2_{aH}	0.733	0.770	0.789	0.792	0.820	0.844	0.874
	R^2_{aP}	0.735	0.771	0.787	0.801	0.821	0.851	0.877
	T_C	3.56	7.70	10.16	23.13	47.44	83.5	199.02
	T_H	2.98	5.35	8.58	11.95	29.76	52.94	141.16
	T_P	3.21	4.85	5.99	10.34	16.48	27.61	57.05

Table A.4: Result comparison of three methods on six different datasets: training results

Noise		D_1	D_2	D_3	D_4	D_5	D_6
	N_k	1000	1000	1000	1000	1000	1000
0%	N_C	2,090,062	11,051,248	10,871,840	376,966	12,382,743	353,038
	N_H	1,339,374	7,581,651	8,527,864	275,526	9,603,147	265,990
	N_P	590,287	2,806,414	2,773,136	240,948	2,290,161	231,107
	R_H	0.64	0.69	0.78	0.73	0.78	0.75
	R_P	0.28	0.25	0.26	0.64	0.18	0.65
	R^2_C	0.941	0.999	0.999	0.999	0.944	0.971
	R^2_H	0.938	0.999	0.999	0.999	0.945	0.971
	R^2_P	0.940	0.999	0.999	0.999	0.943	0.971
	R^2_{aC}	0.936	0.999	0.999	0.999	0.938	0.970
	R^2_{aH}	0.932	0.999	0.999	0.999	0.940	0.970
	R^2_{aP}	0.929	0.999	0.999	0.999	0.938	0.970
	T_C	259.29	1149.26	960.37	30.36	1340.91	28.20
	T_H	165.16	755.43	758.20	22.48	1052.71	21.10
	T_P	70.25	273.41	224.10	19.48	235.69	19.00
5%	N_C	1,988,368	10,710,274	7,174,956	319,140	12,515,344	361,014
	N_H	1,266,933	7,404,639	5,800,284	220,277	9,540,538	269,215
	N_P	518,964	2,621,238	1,712,336	203,052	2,208,984	236,997
	R_H	0.64	0.69	0.81	0.69	0.76	0.75
	R_P	0.26	0.24	0.24	0.64	0.18	0.66
	R^2_C	0.909	0.995	0.993	0.996	0.935	0.952
	R^2_H	0.902	0.995	0.993	0.996	0.935	0.953
	R^2_P	0.906	0.995	0.993	0.996	0.940	0.952
	R^2_{aC}	0.900	0.995	0.992	0.996	0.929	0.949
	R^2_{aH}	0.881	0.995	0.992	0.996	0.929	0.950
	R^2_{aP}	0.897	0.995	0.992	0.996	0.933	0.949
	T_C	259.35	1096.25	648.71	25.68	1406.60	29.33
	T_H	166.78	749.59	574.37	17.52	1028.94	21.80
	T_P	59.66	254.78	146.67	16.47	225.57	20.36
10%	N_C	1,336,330	9,364,412	5,505,934	377,963	12,363,800	325,122
	N_H	909,954	6,365,090	4,421,255	268,752	9,462,023	242,482
	N_P	341,559	2,264,578	1,360,088	240,343	2,202,486	208,351
	R_H	0.68	0.68	0.80	0.71	0.77	0.75
	R_P	0.26	0.24	0.25	0.64	0.18	0.64
	R^2_C	0.891	0.985	0.973	0.991	0.933	0.906
	R^2_H	0.875	0.985	0.973	0.991	0.933	0.906
	R^2_P	0.890	0.985	0.973	0.991	0.933	0.906
	R^2_{aC}	0.880	0.984	0.971	0.990	0.927	0.900
	R^2_{aH}	0.861	0.984	0.971	0.990	0.927	0.900
	R^2_{aP}	0.878	0.984	0.970	0.990	0.926	0.900
	T_C	253.85	1051.80	525.21	31.62	1368.79	25.20
	T_H	155.65	676.18	436.86	22.11	1015.08	19.28
	T_P	55.67	230.08	120.25	19.78	222.06	16.64
20%	N_C	1,755,070	6,510,958	4,491,985	411,861	11,454,536	331,104
	N_H	1,246,389	4,458,339	3,600,931	313,356	8,692,736	242,237
	N_P	514,569	1,477,733	1,061,862	280,717	1,987,142	213,803
	R_H	0.71	0.68	0.80	0.76	0.76	0.73
	R_P	0.29	0.23	0.24	0.68	0.17	0.65
	R^2_C	0.887	0.948	0.898	0.965	0.904	0.794
	R^2_H	0.884	0.948	0.898	0.965	0.904	0.794
	R^2_P	0.887	0.948	0.898	0.965	0.903	0.794
	R^2_{aC}	0.877	0.942	0.887	0.963	0.894	0.781
	R^2_{aH}	0.874	0.942	0.887	0.963	0.894	0.781
	R^2_{aP}	0.877	0.942	0.887	0.963	0.893	0.781
	T_C	199.02	772.69	418.99	33.97	1243.55	26.18
	T_H	141.16	526.57	340.56	25.89	922.75	19.42
	T_P	57.05	158.24	92.90	23.50	195.84	17.24

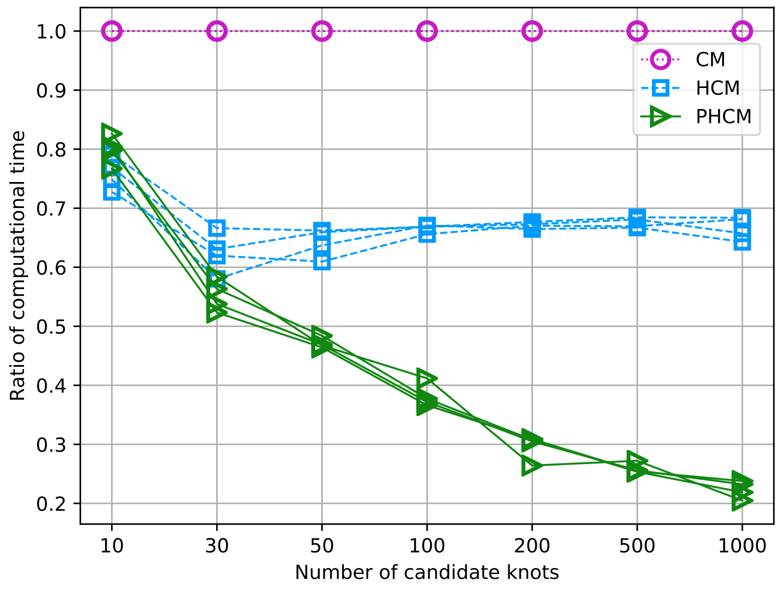


Figure A.13: Computational time ratios of three methods on D_2 under different knot number settings with four noise levels: $\boldsymbol{x}$ is 10 dimensional

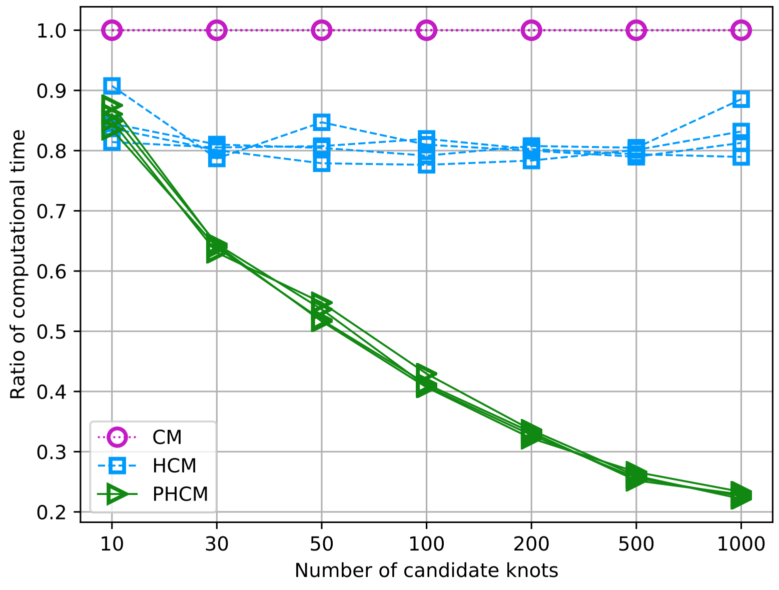


Figure A.14: Computational time ratios of three methods on D_3 under different knot number settings with four noise levels: $\boldsymbol{x}$ is 10 dimensional

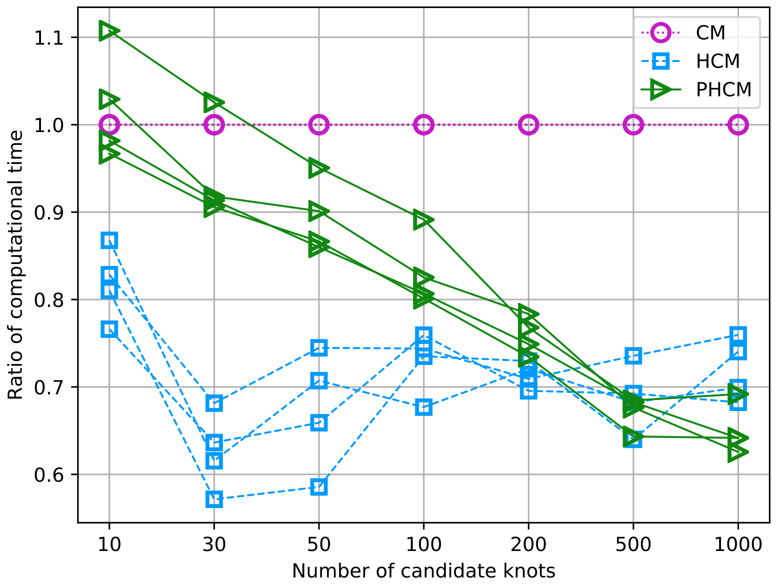


Figure A.15: Computational time ratios of three methods on D_3 under different knot number settings with four noise levels: $\boldsymbol{x}$ is 2 dimensional

References

[1] J. H. Friedman, Multivariate adaptive regression splines, The annals of statistics (1991) 1–67.

[2] J. Leathwick, D. Rowe, J. Richardson, J. Elith, T. Hastie, Using multivariate adaptive regression splines to predict the distributions of new zealand's freshwater diadromous fish, Freshwater Biology 50 (2005) 2034–2052.

[3] J. Leathwick, J. Elith, T. Hastie, Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions, Ecological modelling 199 (2006) 188–196.

[4] T.-S. Lee, C.-C. Chiu, Y.-C. Chou, C.-J. Lu, Mining the customer credit using classification and regression tree and multivariate adaptive regression splines, Computational Statistics & Data Analysis 50 (2006) 1113–1130.

[5] J. Deichmann, A. Eshghi, D. Haughton, S. Sayek, N. Teebagy, Application of multiple adaptive regression splines (mars) in direct response modeling, Journal of Interactive Marketing 16 (2002) 15–27.

[6] Z. Yang, V. C. Chen, M. E. Chang, M. L. Sattler, A. Wen, A decision-making framework for ozone pollution control, Operations Research 57 (2009) 484–498.

[7] C. Conoscenti, V. Agnesi, M. Cama, N. A. Caraballo-Arias, E. Rotigliano, Assessment of gully erosion susceptibility using multivariate adaptive regression splines and accounting for terrain connectivity, Land degradation & development 29 (2018) 724–736.

[8] S. S. Roy, R. Roy, V. E. Balas, Estimating heating load in buildings using multivariate adaptive regression splines, extreme learning machine, a hybrid model of mars and elm, Renewable and Sustainable Energy Reviews 82 (2018) 4256–4268.

[9] S. Heddam, O. Kisi, Modelling daily dissolved oxygen concentration using least square support vector machine, multivariate adaptive regression splines and m5 model tree, Journal of hydrology 559 (2018) 499–509.

[10] V. C. Chen, D. Ruppert, C. A. Shoemaker, Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming, Operations Research 47 (1999) 38–53.

[11] E. K. Koc, C. Iyigun, Restructuring forward step of mars algorithm using a new knot selection procedure based on a mapping approach, Journal of Global Optimization 60 (2014) 79–102.

[12] S. Miyata, X. Shen, Free-knot splines and adaptive knot selection, Journal of the Japan Statistical Society 35 (2005) 303–324.

[13] P. Bratley, B. Fox, Implementing sobols quasirandom sequence generator (algorithm 659), ACM Transactions on Mathematical Software 29 (2003) 49–57.

[14] F. Liu, Z. Wang, A novel adaptive genetic algorithm for wine farm layout optimization, in: Power Symposium (NAPS), 2017 North American, IEEE, 2017, pp. 1–6.

[15] M. Laguna, R. Martí, Experimental testing of advanced scatter search designs for global optimization of multimodal functions, Journal of Global Optimization 33 (2005) 235–255.

[16] D. H. Johnson, Signal-to-noise ratio, Scholarpedia 1 (2006) 2088.

[17] K. Mahdavi, M. Harman, R. M. Hierons, A multiple hill climbing approach to software module clustering, in: Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference on, IEEE, 2003, pp. 315–324.

[18] X. Ju, Y. Duan, C. Ma, H. Ju, Predicting service execution time towards a runtime monitoring approach, in: Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on, IEEE, 2013, pp. 221–224.