ECoFFeS Manual

Release 1.0

Jiawei Huang, Yong Wang and Dongsheng Cao

January 3, 2017

CONTENTS

1. Installation

1.1 System Requirements

64-Bit ECoFFeS				
Operating Systems	Processors	Disk Space	RAM	Graphics
Windows 10 Windows 8.1 Windows 8 Windows 7 Service Pack 1 Windows Server 2012 Windows Server 2008 R2 Service Pack 1	Any Intel or AMD x86-64 processor 4 cores is recommended	5 GB is required	4 GB is required	No specific graphics card is required. Hardware accelerated graphics card supporting OpenGL 3.3 with 1 GB GPU memory is recommended.

Note: Microsoft Office needs to be installed beforehand.

1.2 Download and Install

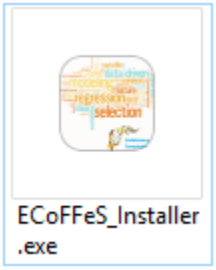
Step 1: Download

Download the files (**ECoFFeS_Installer.part01.rar ~ ECoFFeS_Installer.part10.rar)** from the following website and then unzip the files.

https://github.com/Jiaweihuang/ECoFFeS.

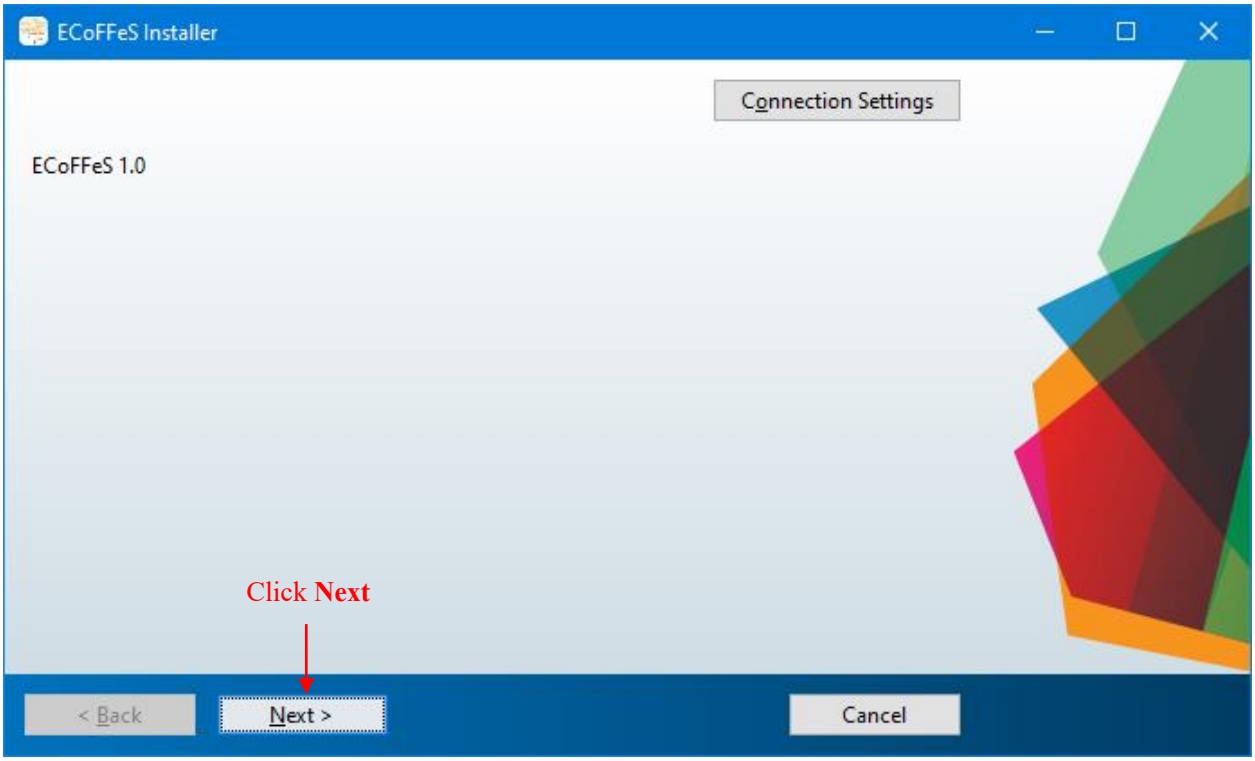
Step 2: Start the Installer

Double-click the **ECoFFeS_Installer.exe** to begin the installation.

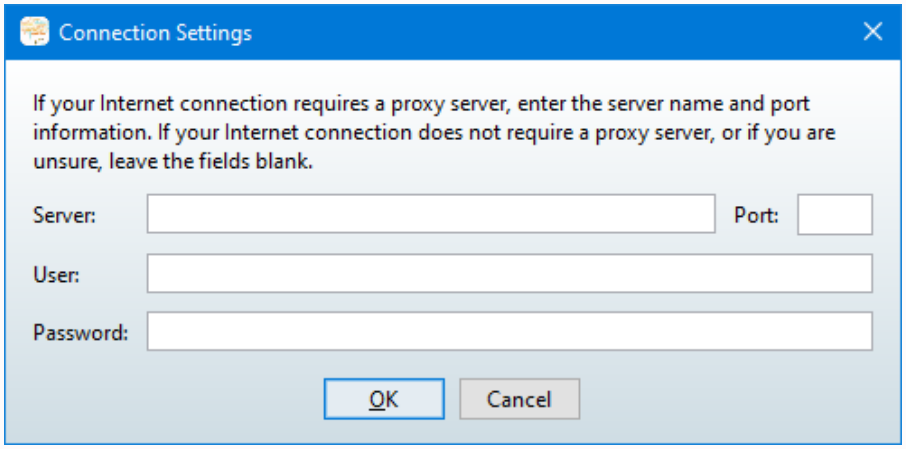


Step 3: ECoFFeS Installer

Click **Next**.



If your Internet connection requires a proxy server, click **Connection Settings**. Enter the server name, port, and password in the Connection Settings page.

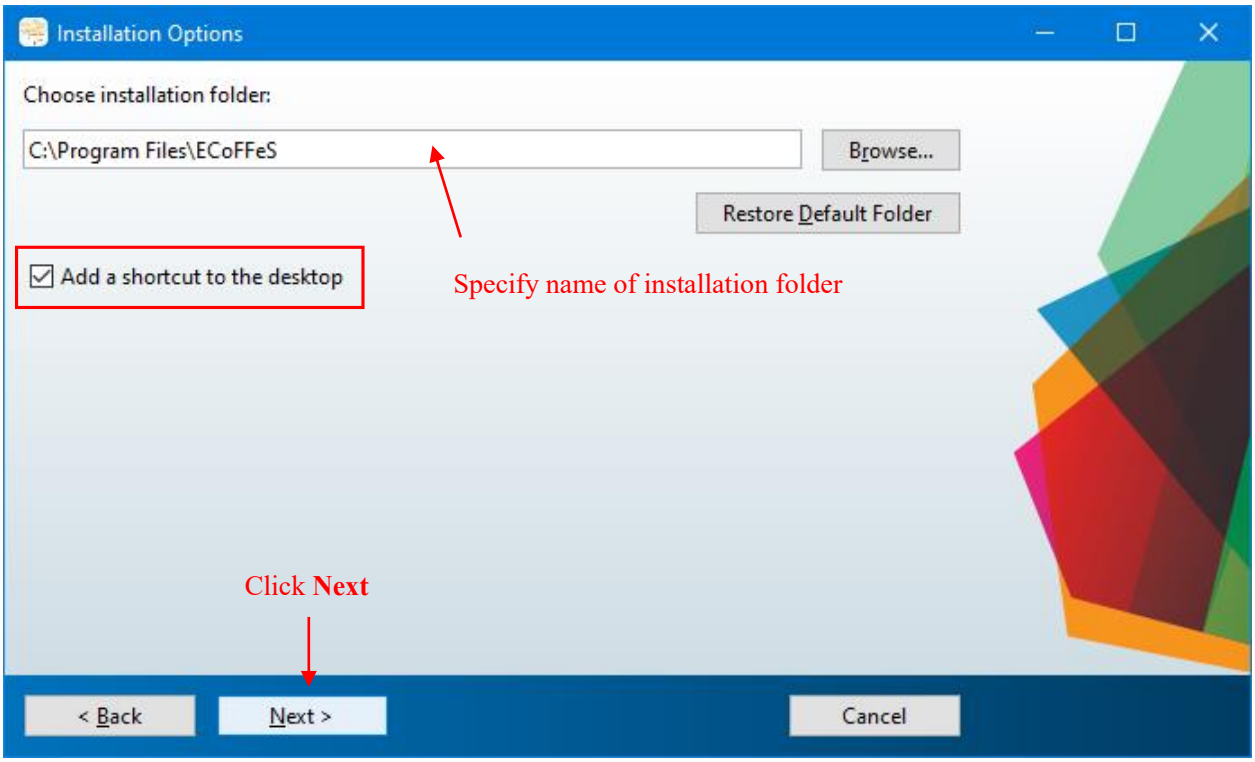


Note: On Windows systems, the installer uses your system proxy settings, by default. If your proxy server requires you to log in, the installer prompts you for your login information.

Step 4: Specify the Installation Folder

Specify the name of the folder where you want to install ECoFFeS products. Accept the default installation folder, or click **Browse** to select a different one. If the folder does not exist, the installer creates it.

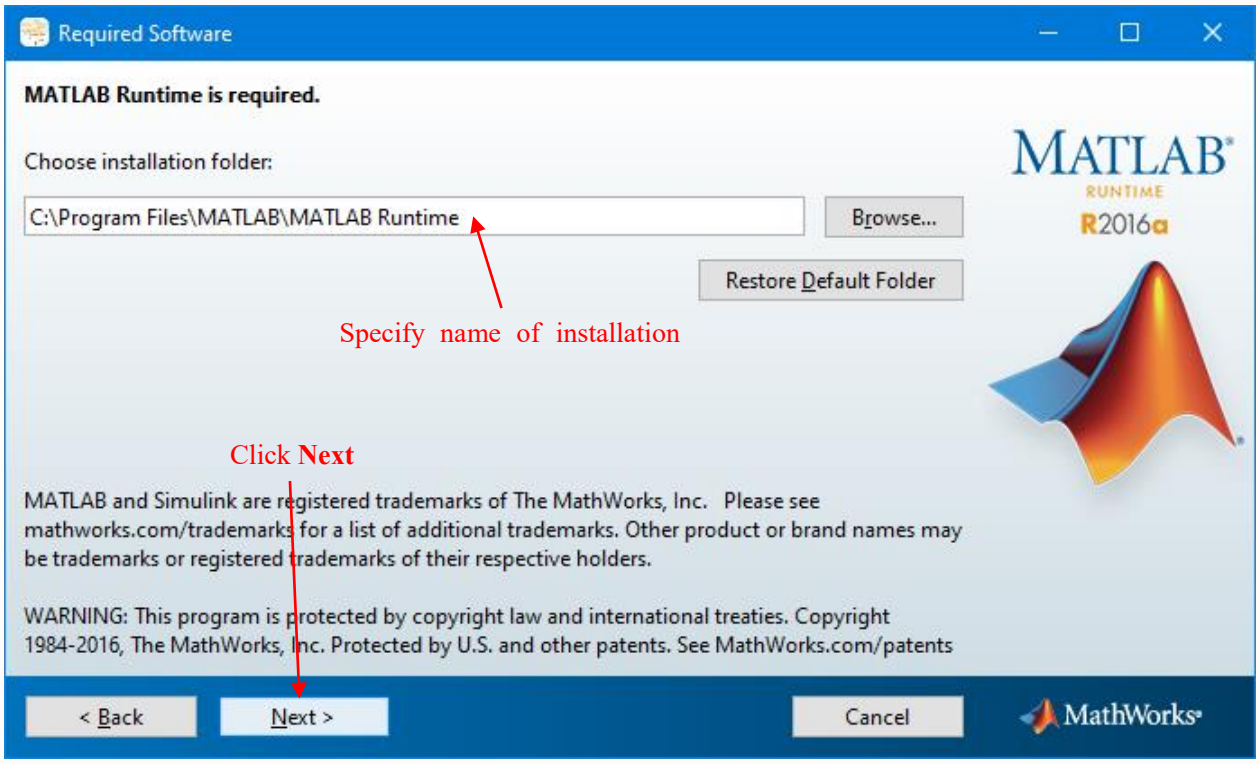
When specifying a folder name, you can use any alphanumeric character and some special characters, such as underscores. If you make a mistake while entering a folder name and want to start over, click **Restore Default Folder**. On Windows, the Installation Options dialog box lets you choose whether to put shortcuts for starting ECoFFeS on the desktop. After making your selection, click **Next**.



Step 5: Required Software

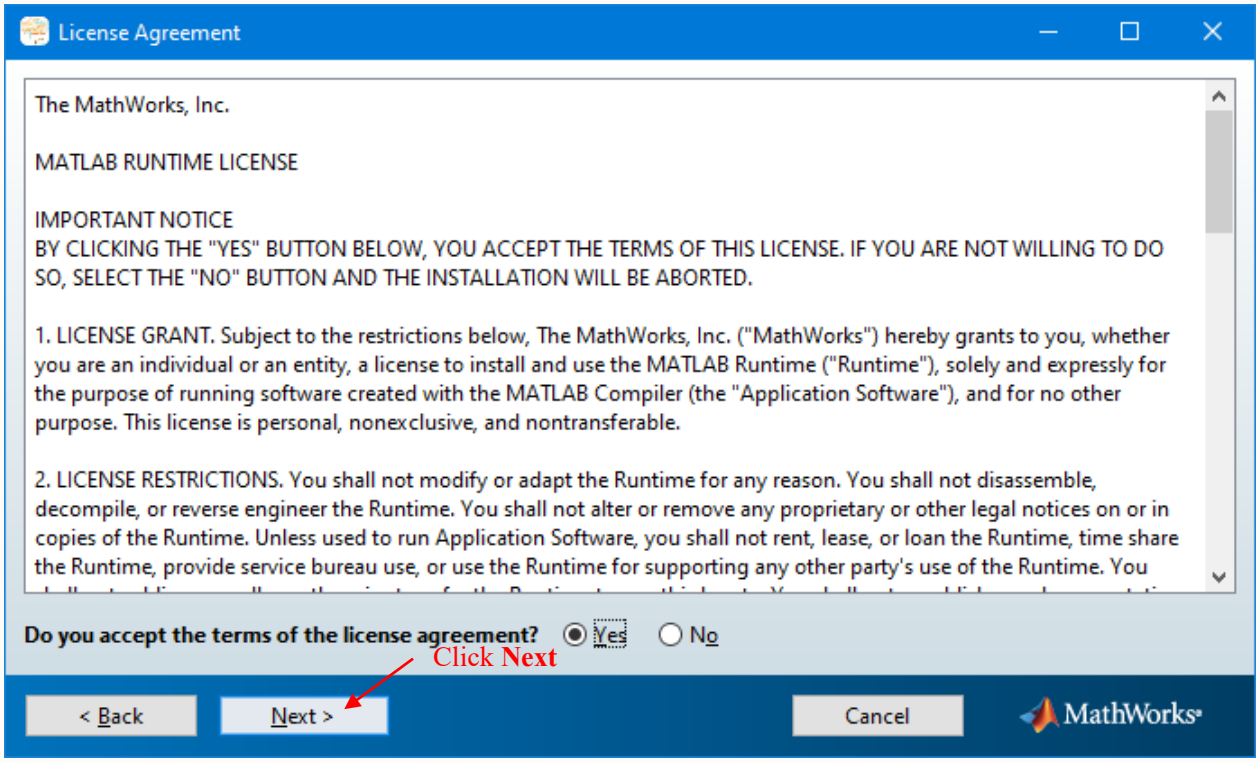
The MATLAB Runtime is a standalone set of shared libraries, MATLAB code, and other files that enable the execution of MATLAB programs on computers without an installed version of MATLAB. Specify the name of the folder where you want to install MATLAB Runtime. Accept the default installation folder, or click **Browse** to select a different one. If the folder does not exist, the installer creates it.

When specifying a folder name, you can use any alphanumeric character and some special characters, such as underscores. If you make a mistake while entering a folder name and want to start over, click **Restore Default Folder**. After making your selection, click **Next** to proceed with the installation.



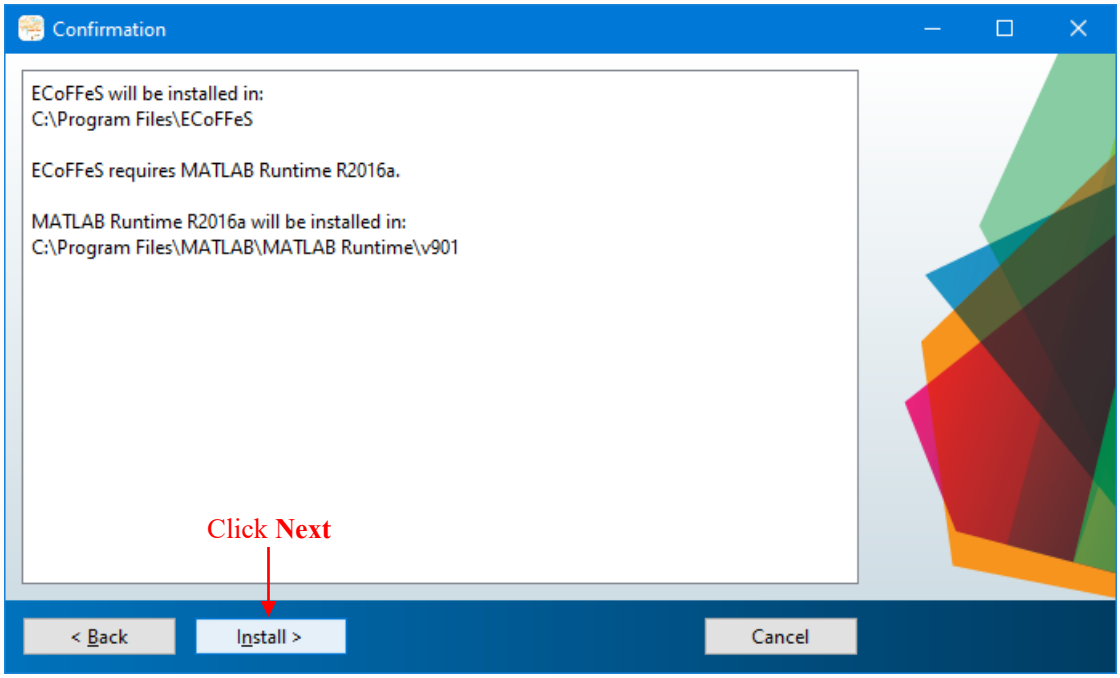
Step 6: License Agreement

Review the software license agreement and, if you agree with the terms, select **Yes** and click **Next**.



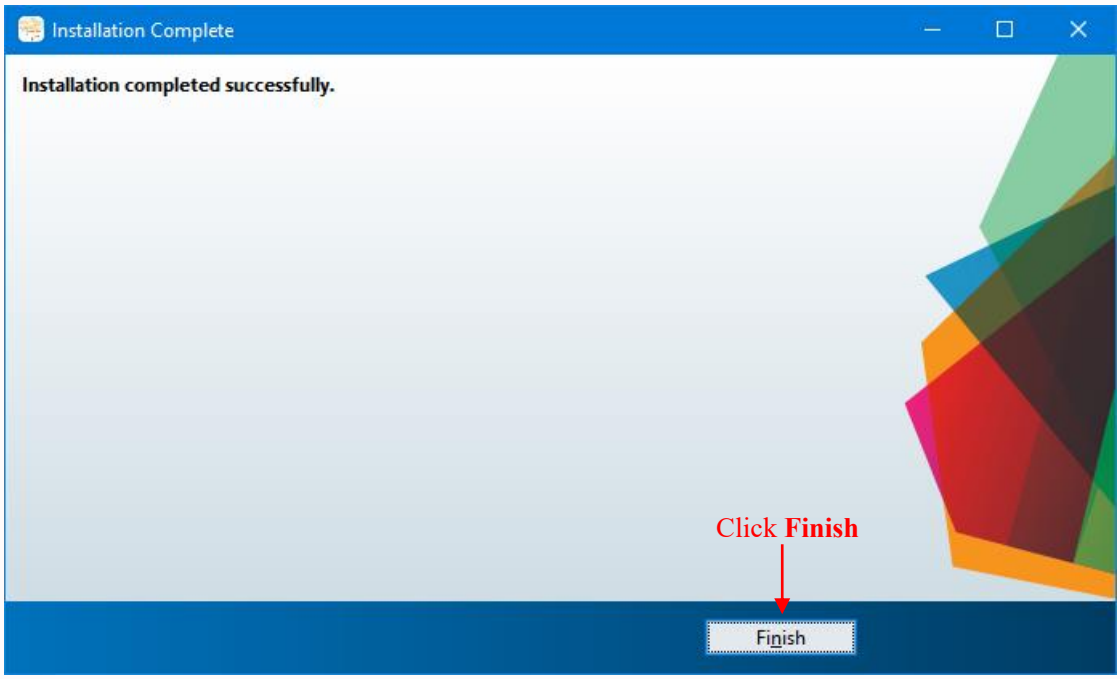
Step 7: Confirm Your Choices

Before it begins installing the software on your hard disk, the installer displays a summary of your installation choices. To change a setting, click **Back**. To proceed with the installation, click **Install**.



Step 8: Complete the Installation

When the installation successfully completes, Click **Finish** to complete the installation.



1.3 Further Development of ECoFFeS

If you are interested in the further development of ECoFFeS, you can download the source code from the following website.
https://github.com/Jiaweihuang/ECoFFeS.

Note: ECoFFeS is developed based on MATLAB (Windows/Linux/Mac), release >= R2016a

2. Introduction

Recent decade has witnessed significant progress in the development of feature selection for many bioinformatics and cheminformatics applications, such as sequence analysis, microarray analysis, mass spectra (MS) analysis, single nucleotide polymorphism (SNP) analysis, and quantitative structure-activity relationship (QSAR) analysis [1][2]. Compared with other dimensionality reduction techniques, feature selection merely selects a feature subset and does not alter the original representation of the features [1]. Thus, the selected feature subset preserves the semantics of the features while offering the advantage of interpretability. For example, in QSAR analysis, the selected feature subset improves the interpretability of relationship between descriptors and biological activities [3].

However, feature selection remains a challenging task due to the fact that it is an NP-hard problem, in which the total number of possible feature subsets is 2^n-1 (n is the number of features). To deal with this issue, many search methods have been proposed, such as complete search, greedy search, and heuristic search [20]. Nevertheless, most of them tend to suffer from stagnation in a local optimum. Evolutionary computation, as a kind of powerful global search method inspired by nature [4], has attracted a high level of interest from the feature selection research community [20]. For the purpose of further boosting evolutionary computation for feature selection, we develop a user-friendly and standalone software named ECoFFeS (Evolutionary Computation For Feature Selection). To the best of our knowledge, it is the first software to integrate a set of evolutionary algorithms (including two modified evolutionary algorithms proposed by the authors) with various evaluation combinations for feature selection. Among these evolutionary algorithms, four are single-objective evolutionary algorithms and two are multi-objective evolutionary algorithms. In addition, ECoFFeS supports parallel execution which can significantly reduce the total analysis time.

3. Graphical User Interface

To cope with feature selection problems in the fields of bioinformatics and cheminformatics, we design a standalone software called ECoFFeS, which provides a user-friendly and easy-to-use graphical user interface.

3.1 The Main Interface

Figure 3.1 shows the main interface of ECoFFeS. It includes three parts:

1) **Subset Discovery**: 'SOEAs (Single-objective Evolutionary Algorithms)' or 'MOEAs (Multi-objective Evolutionary Algorithms)'.

2) **Subset Evaluation**: 'Regression Model & Metric' or 'Classification Model & Metric'.

3) **Start**: Start the secondary interface.

Figure 3.1 The main interface of ECoFFeS. Through combining 'Subset Discovery' with 'Subset Evaluation', a corresponding secondary interface, namely, 'SOEAs_Regression', 'MOEAs_Regression', 'SOEAs_Classification' or 'MOEAs_Classification' can be produced.

3.2 The Secondary Interface

3.2.1 SOEAs_Regression

In main interface, 'Subset Discovery' selects **'SOEAs'**, 'Subset Evaluation' selects **'Regression Model & Metric'**, click **Start**, then the secondary interface **'SOEAs_Regression'** can be produced. Figure 3.2 shows the secondary interface: **'SOEAs_Regression'**. It includes six panels:

1) **Data**: show the data that has been imported.
2) **Results**: show the results of the calculation.
3) **Figure**: show the charts of the calculation.
4) **State**: show the current operating status.
5) **Settings**: parameter settings (Import Data, Parallelization, Save Figure, SOEAs, SOEAs_parameter, Models, Model_parameter, Metric, Metric_parameter, Popsize, Iteration, Runs). For a specific feature selection analysis, datasets can be imported as XLS or XLSX files. The resultant data can be exported as XLS or XLSX files, and the resultant figures can be exported as JPG, PNG or FIG files.
6) **Command**: command control (Play, Stop, menu).

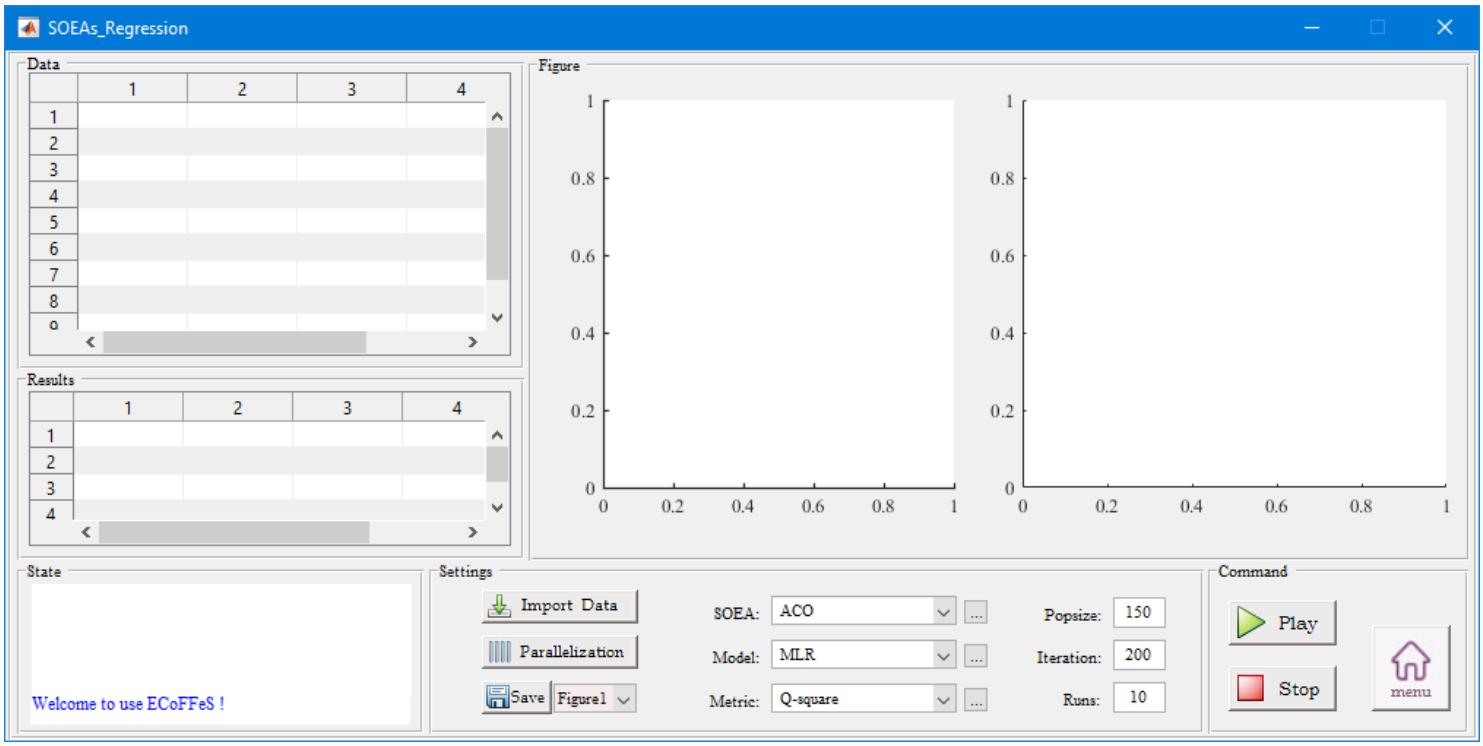


Figure 3.2 The secondary interface: 'SOEAs_Regression'

3.2.2 MOEAs_Regression

In main interface, 'Subset Discovery' selects **'MOEAs'**, 'Subset Evaluation' selects **'Regression Model & Metric'**, click **Start**, then the secondary interface **'MOEAs_Regression'** can be produced. Figure 3.3 shows the secondary interface: **'MOEAs_Regression'**. It includes six panels:

1) **Data**: show the data that has been imported.
2) **Results**: show the results of the calculation.
3) **Figure**: show the charts of the calculation.
4) **State**: show the current operating status.
5) **Settings**: parameter settings (Import Data, Parallelization, Save Figure, MOEAs, MOEAs_parameter, Models, Model_parameter, Metric, Metric_parameter, Popsize, Iteration, Runs). For a specific feature selection analysis, datasets can be imported as XLS or XLSX files. The resultant data can be exported as XLS or XLSX files, and the resultant figures can be exported as JPG, PNG or FIG files.
6) **Command**: command control (Play, Stop, menu).

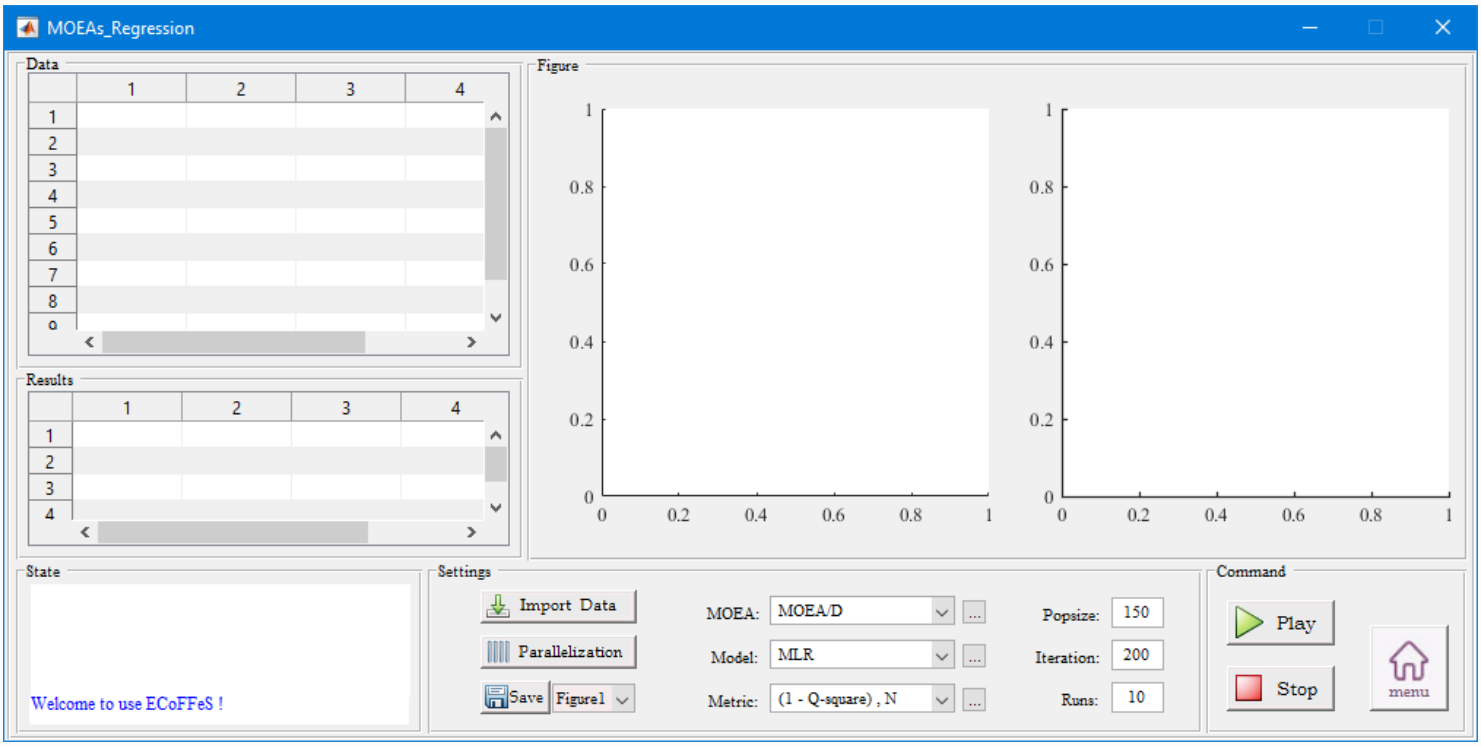


Figure 3.3 The secondary interface: MOEAs_Regression

3.2.3 SOEAs_Classification

In main interface, 'Subset Discovery' selects **'SOEAs'**, 'Subset Evaluation' selects **'Classification Model & Metric'**, click **Start**, then the secondary interface **'SOEAs_Classification'** can be produced. Figure 3.4 shows the secondary interface: **'SOEAs_Classification'**. It includes six panels:

1) **Data**: show the data that has been imported.
2) **Results**: show the results of the calculation.
3) **Figure**: show the charts of the calculation.
4) **State**: show the current operating status.
5) **Settings**: parameter settings (Import Data, Parallelization, Save Figure, SOEAs, SOEAs_parameter, Models, Model_parameter, Metric, Metric_parameter, Popsize, Iteration, Runs). For a specific feature selection analysis, datasets can be imported as XLS or XLSX files. The resultant data can be exported as XLS or XLSX files, and the resultant figures can be exported as JPG, PNG or FIG files.
6) **Command**: command control (Play, Stop, menu).

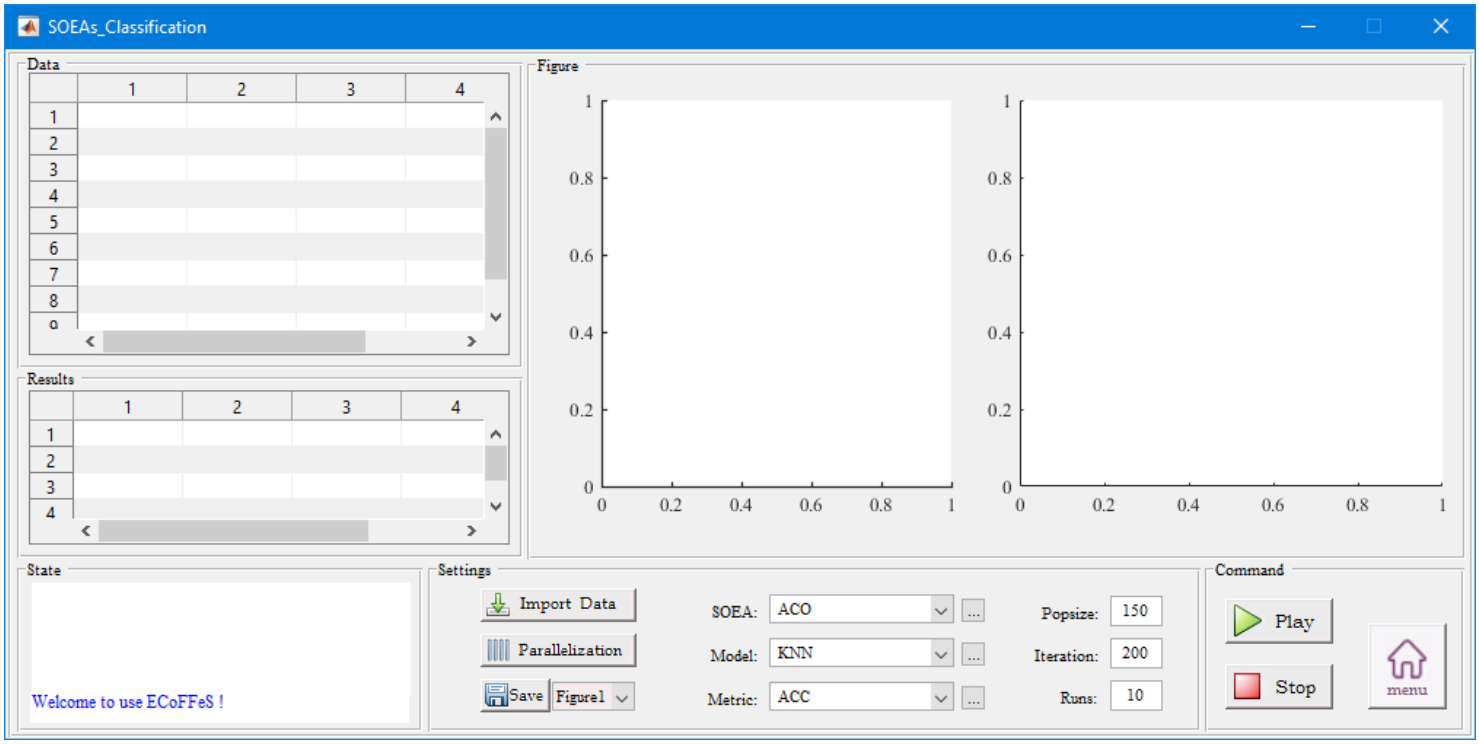


Figure 3.4 The secondary interface: SOEAs_Classification

3.2.4 MOEAs_Classification

In main interface, 'Subset Discovery' selects **'MOEAs'**, 'Subset Evaluation' selects **'Classification Model & Metric'**, click **Start**, then the secondary interface **'MOEAs_Classification'** can be produced. Figure 3.5 shows the secondary interface: **'MOEAs_Classification'**. It includes six panels:

1) **Data**: show the data that has been imported.
2) **Results**: show the results of the calculation.
3) **Figure**: show the charts of the calculation.
4) **State**: show the current operating status.
5) **Settings**: parameter settings (Import Data, Parallelization, Save Figure, MOEAs, MOEAs_parameter, Models, Model_parameter, Metric, Metric_parameter, Popsize, Iteration, Runs). For a specific feature selection analysis, datasets can be imported as XLS or XLSX files. The resultant data can be exported as XLS or XLSX files, and the resultant figures can be exported as JPG, PNG or FIG files.
6) **Command**: command control (Play, Stop, menu).

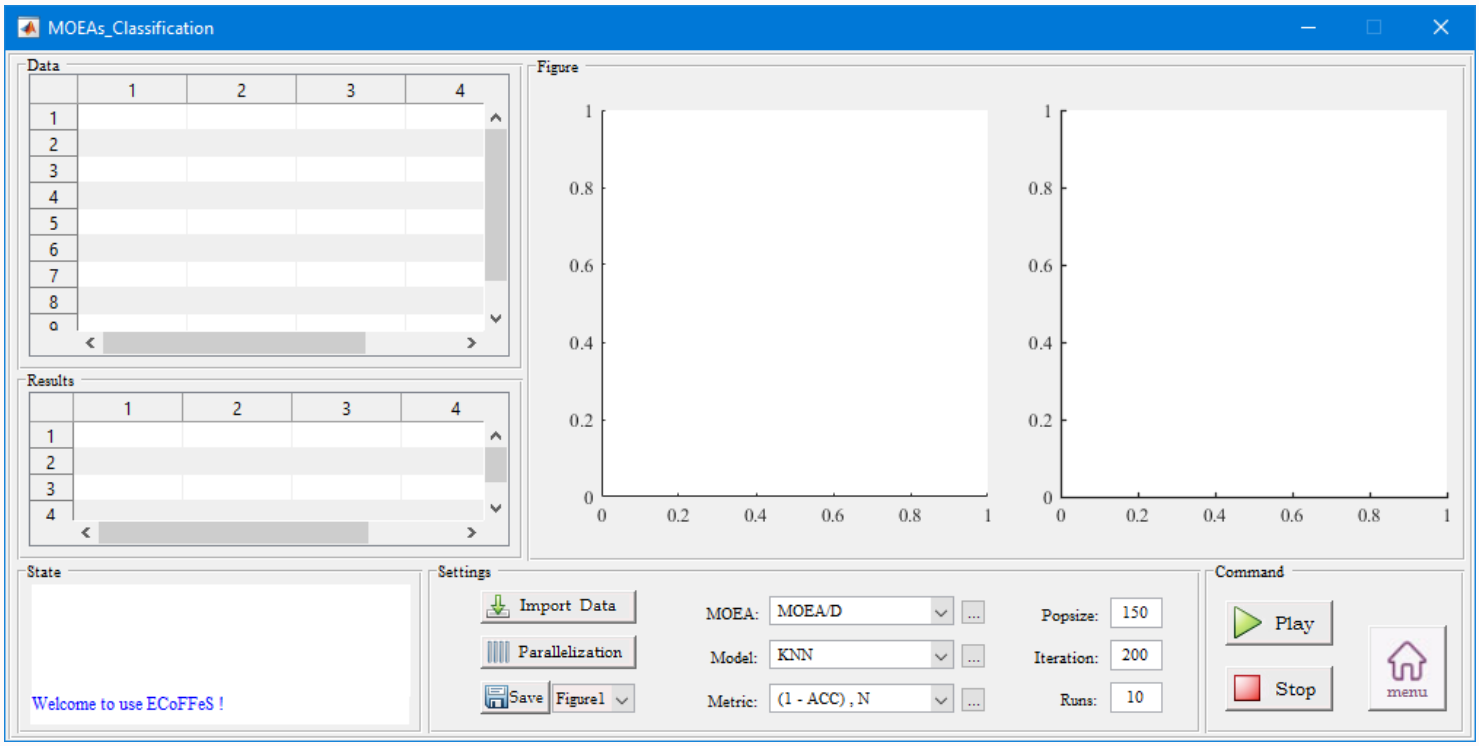


Figure 3.5 The secondary interface: MOEAs_Classification

4. Internal Structure of ECoFFeS

ECoFFeS integrates a set of evolutionary algorithms (including two modified evolutionary algorithms proposed by the authors) with various evaluation combinations for feature selection. In addition, ECoFFeS supports parallel execution which can significantly reduce the total analysis time.

4.1 The Framework

Figure 4.1 shows a general feature selection process and all the five components. Among them, Subset Discovery is a search procedure to generate candidate feature subsets, and Subset Evaluation is a process to assess the candidate feature subsets produced by the Subset Discovery. Detailed discussions about Figure 4.1 can be seen in [6-8].

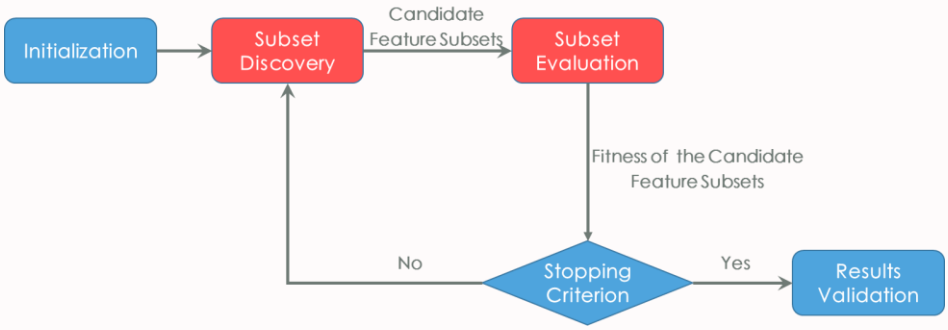


Figure 4.1 General feature selection process

Figure 4.2 shows the internal structure of ECoFFeS, which consists of four main parts. It is evident that each part, which contains two key components, i.e., 'Subset Discovery' and 'Subset Evaluation', corresponds to a secondary interface.

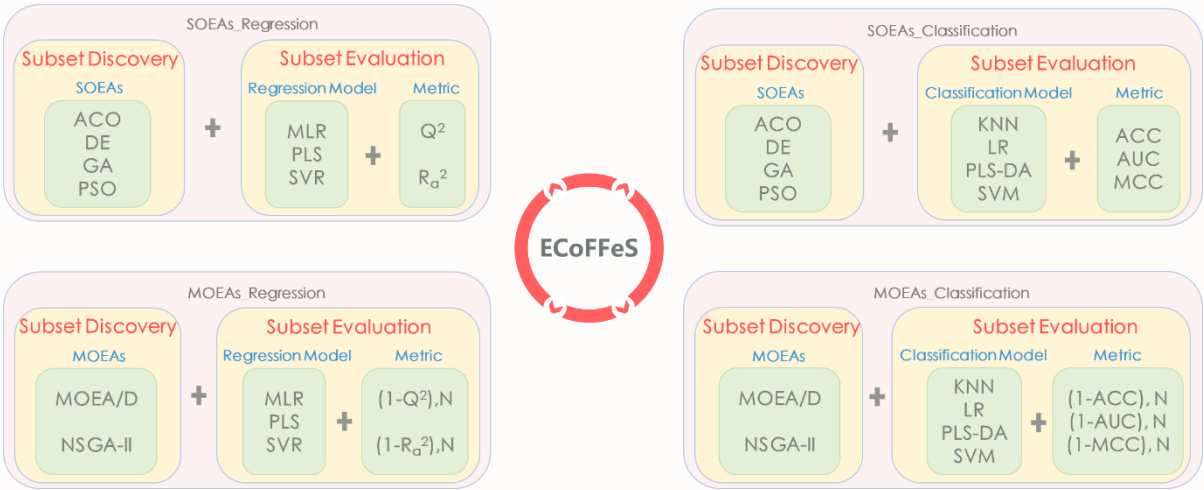


Figure 4.2 the internal structure of ECoFFeS

4.2 Subset Discovery: Evolutionary Computation

Currently, there is no feasible way to select the optimal feature subset due to the fact that it is an NP-hard problem, in which the total number of possible feature subsets is 2^n-1 (n is the number of features). This "combinatorial explosion" for an exhaustive search leads to a computational load that grows exponentially as the total number of features increases [2][5]. In practical terms, this becomes impossible even for the most powerful computers if there are more than 30 features to be searched. To deal with this issue, a variety of search techniques have been applied to feature selection, such as complete search, greedy search, heuristic search and random search [6][11-14]. However, most existing feature selection methods still suffer from stagnation in local optima or high computational cost [15][16]. Therefore, an efficient global search technique is needed to better solve feature selection problems. Evolutionary computation (EC) techniques have recently received much attention from the feature selection research community as they are well-known for their global search ability [20]. Compared with traditional search methods, EC techniques do not need domain knowledge and do not make any assumptions about the search space, such as whether it is linearly or non-linearly separable, and differentiable. Another significant advantage of EC techniques is that their population based mechanism can produce multiple solutions in a single run. This is particularly suitable for multi-objective feature selection in order to find a set of non-dominated solutions with the trade-off between the number of features and the performance metric [20]. Figure 4.3 shows the categories of EC for feature subset discovery.

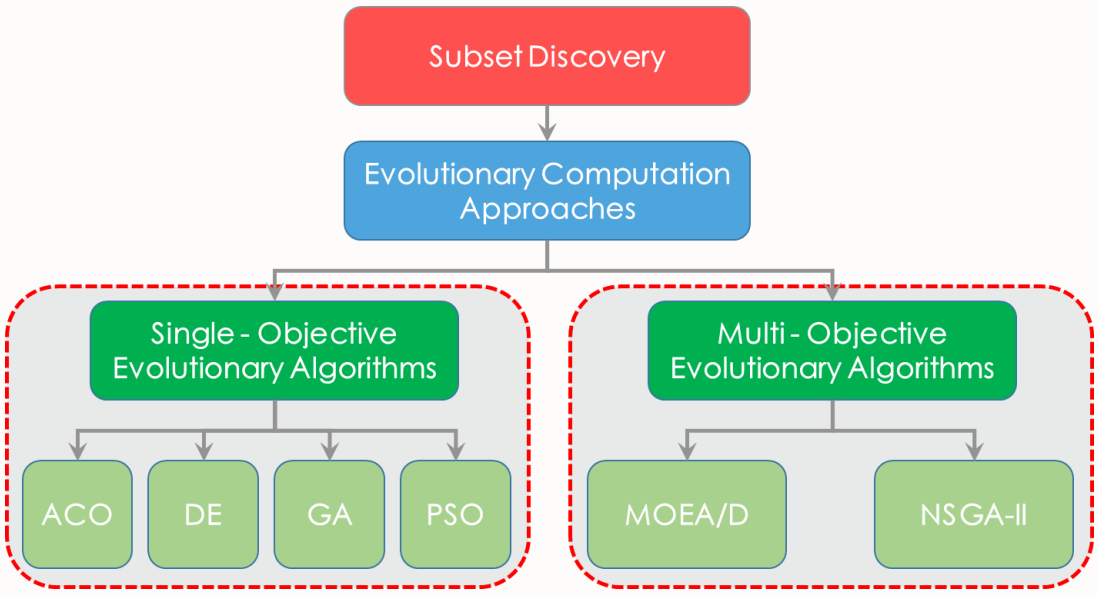


Figure 4.3 The categories of EC for feature subset search

4.2.1 Single-Objective Evolutionary Algorithms

4.2.1.1 Ant Colony Optimization (ACO)

Ant colony optimization (ACO) algorithm is biologically inspired from the behavior of colonies of real ants, and in particular how they forage for food. Since the idea of ACO was proposed by Colorni et al in 1991 [21], it has been successfully applied to solve various discrete combinatorial optimization problems. Generally, a moving ant lays some pheromone (in varying quantities) on the ground, thus marking the path by a trail of this substance. While an isolated ant moves essentially at random, an ant encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behavior that emerges is a form of autocatalytic behavior where the more the ants following a trail, the more attractive that trail becomes for being followed. The process is thus characterized by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path [22][23].

To some extent, feature selection problem is similar to the ants find the routes between nest and food [24][25]. As depicted in Figure 4.4, the feature $\mathrm{F_i}$ $(i = 1,2,\dots,n)$ correspond to two routes, the light green route r_{i1} and the blank route r_{i0}. When an ant k $(k = 1,2,\dots,m)$ came to a crossroads, the only choice is route r_{i1} or route r_{i0}. If ant k chooses route r_{i1}, it means the feature $\mathrm{F_i}$ is selected. Otherwise, ant k chooses route r_{i0}, and it indicates the feature $\mathrm{F_i}$ is not selected. After the tour from the nest to the food, n traversed routes are obtained and it is easy to derive the selected feature subset $\mathrm{FS_k}$ $(k = 1,2,\dots,m)$. For instance, suppose $n = 3$ and 3 routes traversed by ant k are $\{r_{11}, r_{20}, r_{31}\}$, then the selected feature subset is $\mathrm{FS_k} = \{\mathrm{F_1}, \mathrm{F_3}\}$. According to feature subset $\mathrm{FS_k}$, it is easy to obtain the feature subset evaluation value f_k $(k = 1,2,\dots,m)$. Thus, the pheromone laid on n routes that traversed by ant k is updated by its evaluation value f_k. As time goes on, m ants repeated traversed from the nest to the food and the pheromone of each route r_{ij} gradually remain stable. Finally, all m ants choose n routes with high pheromone and the feature subset FS is acquired based on these n routes.

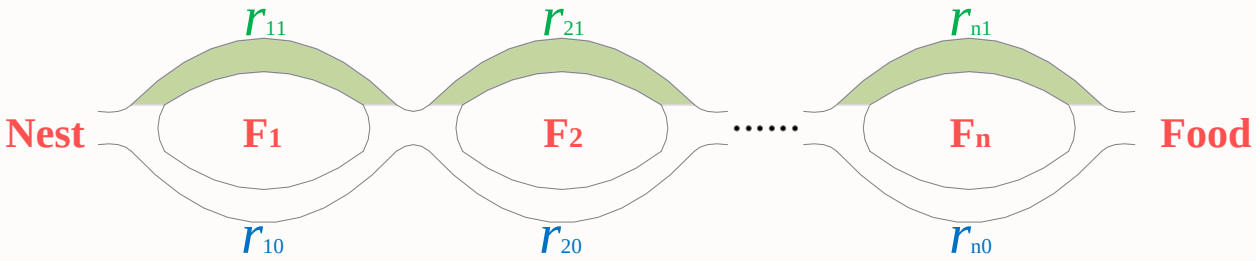


Figure 4.4 The routes graph between nest and food

In the light of the graph representation and the above description, a modified ACO algorithm is proposed. The main characteristic of the modified ACO algorithm is that, at each iteration, the pheromone values are not only updated by all the m ants, but also updated by an additional ant called best-so-far ant. This leads to the exploration of ants around the optimal solution in next iterations. The pheromone $\tau_{ij}(t)$, associated with route r_{ij}, is updated as follows:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) + e\cdot\Delta\tau_{ij}^{best} \tag{4.1}$$

where ρ is the evaporation rate, The parameter ρ is used to avoid unlimited accumulation of the pheromone trails and it enables the algorithm to 'forget' bad decisions previously taken. m is the number of ants, $\Delta\tau_{ij}^{k}(t)$ is the quantity of pheromone laid on route r_{ij} by ant k at iteration t, e is the weight, and $\Delta\tau_{ij}^{best}$ is additional pheromone deposited by the best-so-far ant:

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{f_k(t)}{N_{ij}(t)}, & \text{if ant } k \text{ used } r_{ij} \text{ in its tour} \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

$$\Delta\tau_{ij}^{best} = \begin{cases} 1, & \text{if } r_{ij} \text{ belongs to the best tour} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $f_k(t)$ is the fitness of ant k at iteration t, $N_{ij}(t)$ is the number of ants that traverse through route r_{ij}.

In the construction of a solution, ants select the following routes to be visited through a stochastic mechanism. For ant k, the probability of choosing route r_{ij} is given by:

$$p_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}\cdot[\eta_{ij}]^{\beta}}{\sum_{k\in N_i^k}[\tau_{ik}(t)]^{\alpha}\cdot[\eta_{ik}]^{\beta}}, & \text{if } j\epsilon N_i^k \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

where N_i^k is the feasible routes of ant k, $N_i^k = \{0,1\}$, α and β are two parameters which determine the relative influence of the pheromone and the heuristic information η_{ij}. The role of the parameters α and β is the following. If $\alpha = 0$, the feature $\mathrm{F_i}$ is more likely not to be selected: this corresponds to a classic stochastic greedy algorithm. If $\beta = 0$, only pheromone amplification is at work, that is, only pheromone is used, without any heuristic bias. This generally leads to rather poor results and, in particular, for values of $\alpha > 0$ it leads to the rapid emergence of a stagnation situation, that is, a situation in which all the ants follow the same path and construct the same tour, which, in general, is strongly suboptimal. And η_{ij} is defined as:

$$\eta_{ij} = pp_{ij} \tag{4.5}$$

where pp_{ij} is the prior probability of choose route r_{ij}. In our experiment we assume that the probability of not select the feature $\mathrm{F_i}$ is bigger than the probability of select, so they were set as $pp_{i0} = 0.55,\ pp_{i1} = 0.45\ \ (i = 1,2,\ldots,n)$. The following is the pseudocode of ACO-FS.

Algorithm: ACO-FS

1: $t = 1$; /* t denotes the iteration number */

2: Initialize the pheromone $\tau_{ij}(t)$ and parameters;

3: Compute the transition probability $p_{ij}(t)$ according to Eq. 4.4;

4: Generate m ants and place them on nest;

5: For each ant k $(k = 1,2, \dots, m)$, construct a feature subsect $\mathrm{FS_k}$ $(k = 1,2, \dots, m)$ using n routes that traversed by ant k;

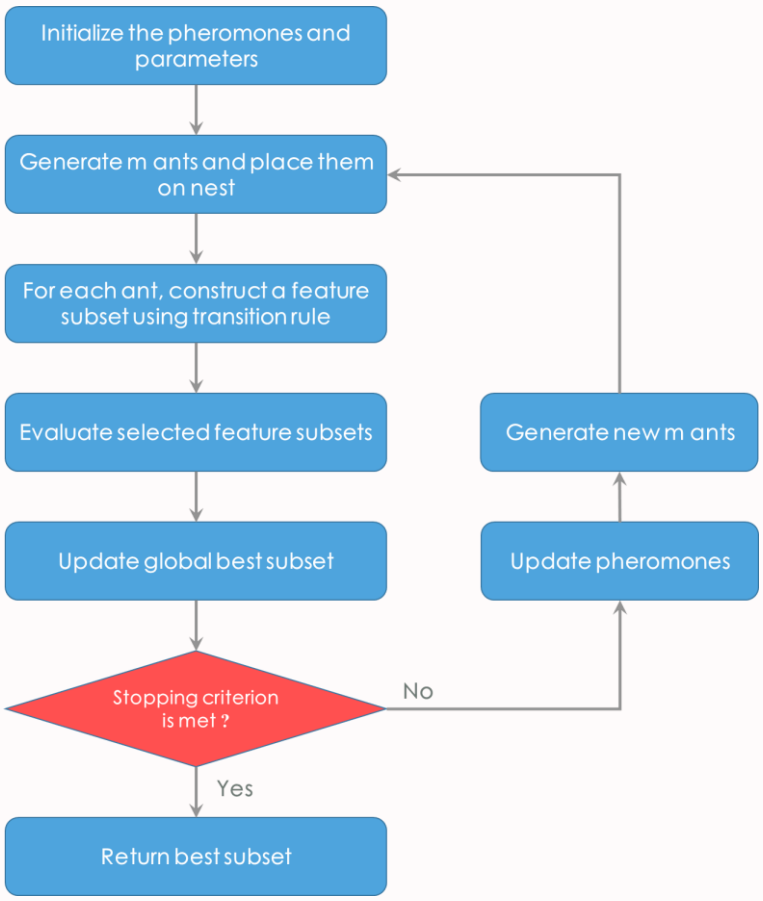
6: Evaluate each feature subset $\mathrm{FS_k}$ and use $f_k(t)$ as the fitness value;

7: Update global best feature subset $\mathrm{FS_{best}}$ and f_{best};

8: Update the pheromone $\tau_{ij}(t+1)$ in accordance with Eq. 4.1;

9: $t = t + 1$;

10: **Stopping Criterion:** If the maximum number of iteration numbers is reached, then stop and output the best feature subset $\mathrm{FS_{best}}$ and f_{best}; otherwise go to step 4.



4.2.1.2 Differential Evolution (DE): a modified DE

Differential evolution (DE) was proposed by Storn and Price in 1995 [26][27], is one of the most popular evolutionary algorithm (EA) paradigms in the community of evolutionary computation. Like other EA paradigms, DE is a population-based optimization method, which contains a lot of solutions. In DE, each solution in the population is called a target vector. DE includes three main operators, i.e., mutation, crossover, and selection. In the classical DE, for each target vector, a mutant vector is generated by making use of the mutation operator. Afterward, the crossover operator is implemented on the target vector and the mutant vector, and thus, a trial vector is obtained. Finally, the target vector is compared with the trial vector, and the better one will be selected for the next population. The mutation operator and the crossover operator together are called the trial vector generation strategy, since they are utilized to generate the trial vector. DE also contains three important control parameters, i.e., the population size, the scaling factor in the mutation operator, and the crossover control parameter in the crossover operator [28][29].

As a discrete optimization problem, feature selection problem can't be solved by DE directed. Thus, a binary approach is used to enable the DE correctly perform optimization in binary spaces, whilst ensuring that the basic fundamentals of DE are not compromised. That is to say, the binary approach uses the floating-point DE individuals to determine a probability for each component. These probabilities are then used to generate a binary-digits solution from the floating-point vector. This binary-digits is used by the fitness function to determine its quality. The resulting fitness is then associated with the floating-point representation of the individual [30].

In the light of the above description, a modified DE algorithm (Binary Feedback Differential Evolution, namely BFDE) is proposed. Let P_G be a population at generation G, which consists of N individuals: $\vec{x}_{1,G}, \dots, \vec{x}_{N,G}$. For each individual $\vec{x}_{i,G} = (x_{i,1,G}, \dots, x_{i,n,G})$, $x_{i,j,G}$ is a floating-point number. Then, the corresponding binary-digits solution $\overrightarrow{xb}_{i,G} = (xb_{i,1,G}, \dots, xb_{i,n,G})$ is calculated using

$$xb_{i,j,G} = \begin{cases} 1, & if \ U(0,1) < S(x_{i,j,G}) \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

where $U(0,1)$ is a single uniformly distributed random number in the interval (0,1) and S is the sigmoid function

$$S(x) = \frac{1}{1+e^{-x}} \tag{4.7}$$

Finally, population Pb_G is generated, which consists of N binary-digits individuals $\overrightarrow{xb}_{1,G}, \dots, \overrightarrow{xb}_{N,G}$. And the brief description of the transformation process from P_G to Pb_G is depicted in Figure 4.5.

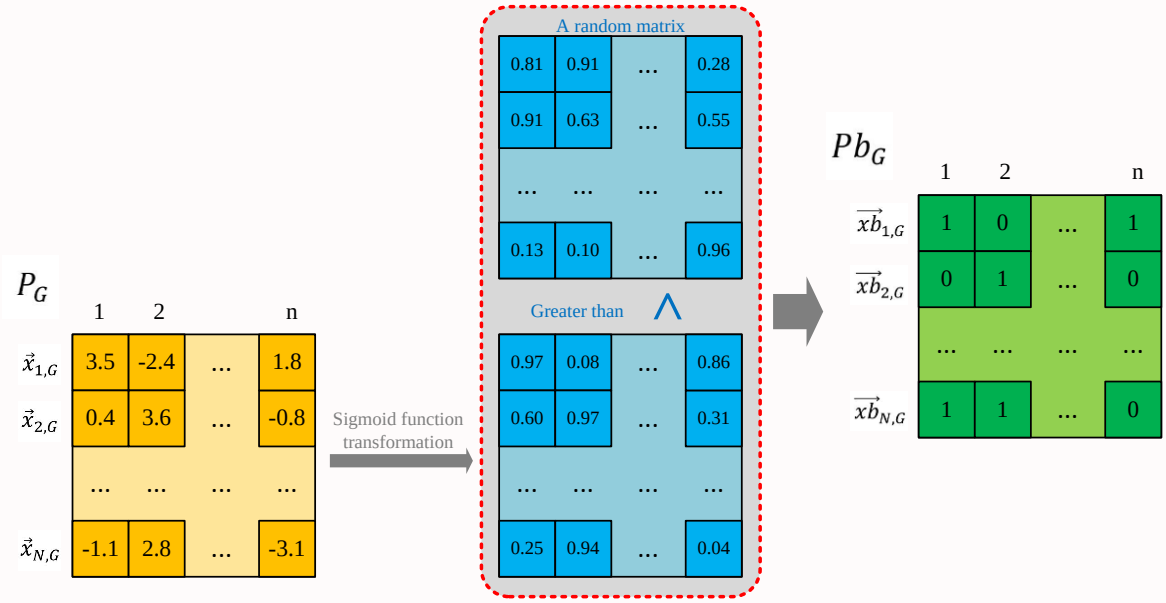


Figure 4.5 The brief description of the transformation process

For each individual $\vec{x}_{i,G}$ (also called a target vector), the following DE/current-to-best/1 mutation operator is applied:

$$\vec{v}_{i,G} = \vec{x}_{i,G} + F \cdot (\vec{x}_{best,G} - \vec{x}_{i,G}) + F \cdot (\vec{x}_{r1,G} - \vec{x}_{r2,G}) \tag{4.8}$$

where $r1$, $r2$, $r3$, $r4$ and $r5$ are mutually different integers chosen from $[1, N]$ and also different from i, $\vec{x}_{best,G}$ is the best target vector at generation G, F is the scaling factor, and $\vec{v}_{i,G}$ is the mutant vector.

In order to limits the ultimate probability that bit $xb_{i,j,G}$ will take on 0 or 1. A simple and popular repair operator works as follows: if the jth element $v_{i,j,G}$ of the mutant vector $\vec{v}_{i,G} = (v_{i,1,G}, \dots, v_{i,n,G})$ is out of the search region $[l_{min}, l_{max}]$, then $v_{i,j,G}$ is reset as follow:

$$v_{i,j,G} = \begin{cases} l_{min}, & if \quad v_{i,j,G} < l_{min} \\ l_{max}, & if \quad v_{i,j,G} > l_{max} \end{cases} \tag{4.9}$$

After the repair, the target vector $\vec{x}_{i,G}$ and its mutant vector $\vec{v}_{i,G}$ will undergo binomial crossover

$$u_{i,j,G} = \begin{cases} v_{i,j,G}, & if \quad rand_j \leq CR \quad or \quad j = j_{rand} \\ x_{i,j,G}, & otherwise \end{cases} \tag{4.10}$$

where $i = 1, \dots, N$, $j = 1, \dots, D$, $rand_j$ is a uniformly distributed random number between 0 and 1 and regenerated for each j, j_{rand} is an integer randomly chosen from $[1, D]$, CR is the crossover control parameter, and $u_{i,j,G}$ is the jth element of the trial vector $u_{i,G}$. Figure 4.6 shows the schematic of binomial crossover.

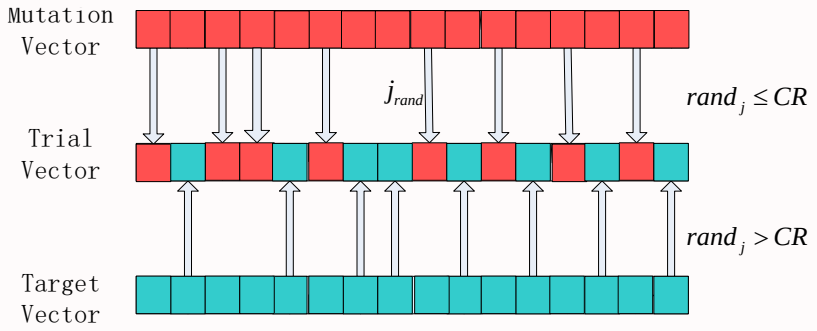


Figure 4.6 The schematic of binomial crossover

Subsequently, the selection is performed between the target vector $\vec{x}_{i,G}$ and its trial vector $\vec{u}_{i,G}$, and the better one will survive into the next population

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G}, & if \ \ f(\overrightarrow{ub}_{i,G}) \geq f(\overrightarrow{xb}_{i,G}) \\ \vec{x}_{i,G}, & otherwise \end{cases} \tag{4.11}$$

where $\overrightarrow{ub}_{i,G}$ is the binary-digits individual of $\vec{u}_{i,G}$, $\overrightarrow{xb}_{i,G}$ is the binary-digits individual of $\vec{x}_{i,G}$ and f is the fitness function.

In addition, more information of the binary-digits population Pb_{G+1} is incorporated to the population P_{G+1}:

$$\vec{x}_{i,G+1} = \vec{x}_{i,G+1} - (1 - \overrightarrow{xb}_{best,G+1}) \tag{4.12}$$

where $\overrightarrow{xb}_{best,G}$ is the binary-digits individual of best individual $\vec{x}_{best,G+1}$ at generation $G+1$. Figure 4.7 shows the framework of BFDE-FS. And the following is the pseudocode of BFDE-FS.

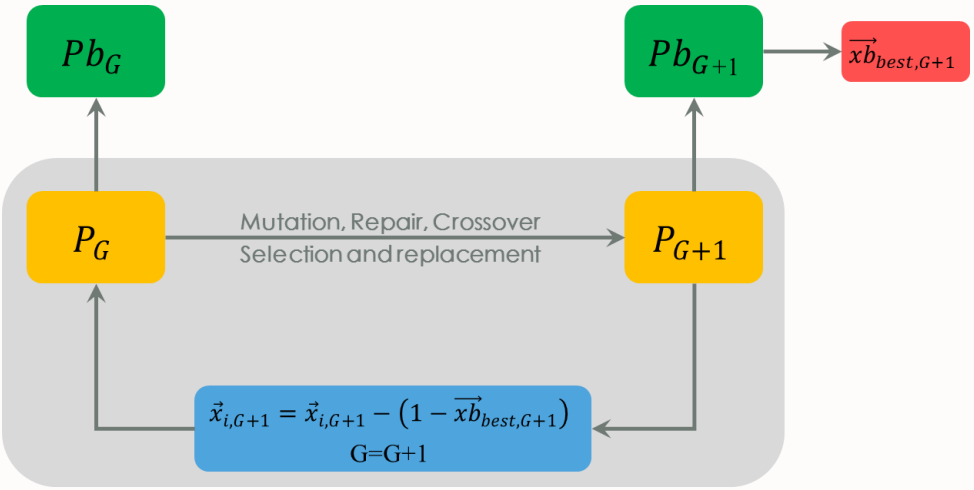


Figure 4.7 The framework of BFDE-FS

Algorithm: BFDE-FS

1: $G = 1$; /* G denotes the generation number */
2: Randomly generate an initial population P_G and obtain the binary-digits population Pb_G by the binary approach;
3: Evaluate each individual in Pb_G and use $f(\overrightarrow{xb_{i,G}})$ as the fitness value of $\vec{x}_{i,G}$.
4: $P_{G+1} = \emptyset$;
5: **For** each individual $\vec{x}_{i,G}$ (also called a target vector) in P_G
6: Generate a mutant vector $\vec{v}_{i,G}$ by using the DE/current-to-best/1 mutation operator; /*Mutation*/
7: If $\vec{v}_{i,G}$ is not feasible (i.e., not in the search space), use a repair operator to make $\vec{v}_{i,G}$ feasible; /*Repair*/
8: Mix $\vec{x}_{i,G}$ and $\vec{v}_{i,G}$ to generate a trial vector $\vec{u}_{i,G}$ by using the binomial crossover operator; /*Crossover*/
9: If $f(\overrightarrow{ub_{i,G}}) \leq f(\overrightarrow{xb_{i,G}})$, set $\vec{x}_{i,G+1} = \vec{u}_{i,G}$; otherwise, set $\vec{x}_{i,G+1} = \vec{x}_{i,G}$, then store $\vec{x}_{i,G+1}$ into P_{G+1}; /*Selection and replacement*/
10: **End For**
11: Incorporate the information of the binary-digits population Pb_G to P_{G+1};
12: $G = \mathrm{G} + 1$;
13: **Stopping Criterion:** If the maximum number of generation numbers is reached, then stop and output the best individual in P_G and Pb_G; otherwise go to step 4.

4.2.1.3 Genetic Algorithm (GA)

Genetic algorithm (GA) was developed by John Holland in the 1960s and first published in 1975 [31][32]. As a population based heuristic method, GA is inspired by the process of natural evolution. In GAs, a solution is represented as a chromosome. Each chromosome in the population is associated with a Fitness Values. A pre-defined number of iterations of evolution follows the initial population generation. In each iteration, some pairs of chromosomes are selected by a biased random selection approach. Chromosomes with the higher fitness values are more likely selected from the population. A crossover approach is implemented on each pair of selected chromosomes to generate some new chromosomes. Some other chromosomes are also selected from the population, followed by a mutation procedure that also generates some other new chromosomes. In each iteration of the GA, the fitness values of all chromosomes in the population are evaluated, and the best chromosome is recorded. After a large number of iterations, the best chromosome in the population is translated as the selected solution [33][34].

According to the characteristics of feature selection problem, a string with n binary digits is used as chromosome representation. For each binary digit, value 1 means that the feature is selected and value 0 has the opposite meaning. In the first step of the proposed algorithm, randomly generated the initial population $Pb_G = \{\overrightarrow{xb}_{i,G} = (xb_{i,1,G}, \dots, xb_{i,n,G}), i = 1, \dots, N\}$ where N is the population size, $\overrightarrow{xb}_{i,G}$ is a binary string with n dimension. And the fitness of the chromosome $\overrightarrow{xb}_{i,G}$ is determined by evaluating the models using a training set whose responses are represented using only the selected feature subset.

Afterwards, the chromosome selection for the next generation is done on the basis of fitness. The selection mechanism should ensure that fitter chromosomes have a higher probability of survival. In proportional roulette wheel selection, chromosomes are selected with a probability that is directly proportional to their fitness values i.e. a chromosome's selection corresponds to a portion of a roulette wheel [35]. Obviously, those with the largest fitness (i.e. largest segment sizes) have more probability of being chosen. The fittest chromosome occupies the largest segment, whereas the least fit have correspondingly smaller segment within the roulette wheel. The circumference of the roulette wheel is the sum of all fitness values of the chromosomes. The proportional roulette wheel mechanism and the algorithm procedure are depicted in Figure 4.8. In the figure, when the wheel is spun, the wheel will finally stop and the pointer attached to it will point on one of the segment, most probably on one of the widest ones. However, all segments have a chance, with a probability that is proportional to its width. By repeating this each time, the better chromosomes will be chosen more often than the poorer ones, thus fulfilling the requirements of survival of the fittest. Let $f_1, f_2, \dots, f_n$ be fitness values of chromosomes $\overrightarrow{xb}_{1,G}, \overrightarrow{xb}_{2,G}, \dots, \overrightarrow{xb}_{N,G}$. Then the selection probability, p_i for chromosome i is

defined as,

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j} \tag{4.13}$$

The basic advantage of proportional roulette wheel selection is that it discards none of the individuals in the population and gives a chance to all of them to be selected. Therefore, diversity in the population is preserved.

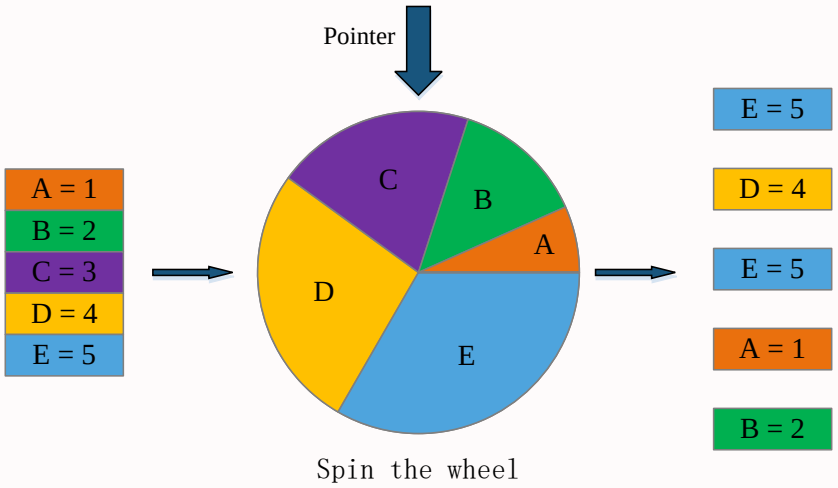


Figure 4.8 The proportional roulette wheel selection

Subsequently, the crossover operator splits up the "parent" chromosomes and recombines them. It is also one of the genetic operators in which genes of two chromosomes are exchanged and the genotypes of two selected parents are merged to yield two new offspring [36]. A two-point crossover operator uses two randomly chosen crossover points $Cpoint_1$ and $Cpoint_2$, where $1 \leq Cpoint_1 < Cpoint_2 \leq n$. Then, generate an uniformly distributed random number $rand$ in the interval $[0,1]$. If $rand < pc$, chromosomes exchange the segment that falls between these two points. Finally, two new offspring are created and put back into the chromosomes pool. The operations are exemplified in Figure 4.9.

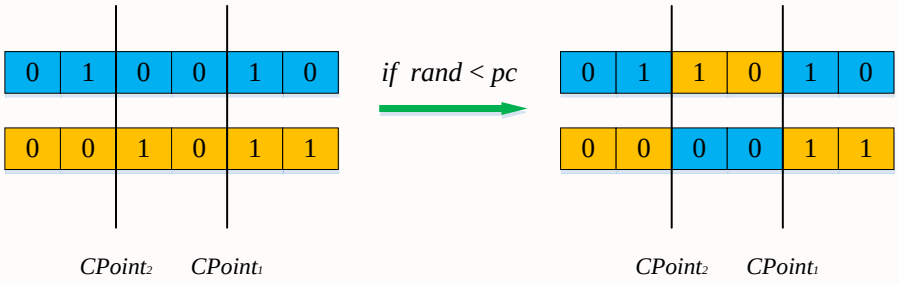


Figure 4.9 Two-point crossover operator

The mutation simply adds "genes" to the chromosomes. Mutation introduces new genetic structures in the population by randomly modifying some of the genes, such that the search algorithm can escape from the local optimum and avoid the genetic algorithm from converging too fast. In other

words, mutation operation gives genetic algorithm an opportunity to search for new and more feasible chromosomes in new areas of the solution spaces [36]. The probability of each chromosome to perform mutation is pm. For each chromosome $\overrightarrow{xb_{i,G}} = (xb_{i,1,G}, \dots, xb_{i,n,G})$, generate n uniformly distributed random numbers $rand_1, \dots, rand_n$ in the interval $[0,1]$. If $rand_i < pm$, $xb_{i,j,G} = \overline{xb_{i,J,G}}$. An example of the mutation procedure is shown in Figure 4.10. The following is the pseudocode of GA-FS.

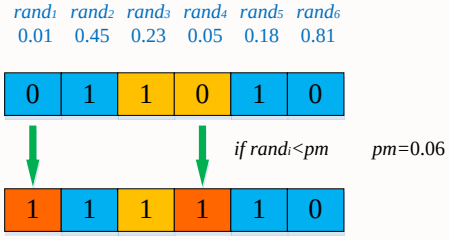


Figure 4.10 Mutation operator

Algorithm: GA-FS

1: $G = 1$; /* G denotes the generation number */

2: Randomly generate an initial population Pb_G with N chromosomes $\overrightarrow{xb_{1,G}}, \overrightarrow{xb_{2,G}}, \dots, \overrightarrow{xb_{N,G}}$;

3: Evaluate each chromosome in the population Pb_G and find the best chromosome $\overrightarrow{xb_{best,G}}$.

4: $Pb_{G+1} = \emptyset$;

5: **Selection:** Generate a parent population $Sb_G = \{\overrightarrow{sb_{1,G}}, \overrightarrow{sb_{2,G}}, \dots, \overrightarrow{sb_{N-2,G}}\}$ from population Pb_G by proportional roulette wheel selection operator;

6: **Crossover:** Generate a new population $Cb_G = \{\overrightarrow{cb_{1,G}}, \overrightarrow{cb_{2,G}}, \dots, \overrightarrow{cb_{N-2,G}}\}$ from population Sb_G by 2-point crossover operator;

7: **Mutation:** Generate a new population $Mb_G = \{\overrightarrow{mb_{1,G}}, \overrightarrow{mb_{2,G}}, \dots, \overrightarrow{mb_{N-2,G}}\}$ from population Cb_G by mutation operator;

8: Combine populations Mb_G and Pb_G to form an offspring population $Pb_{G+1} = \{\overrightarrow{xb_{1,G+1}}, \overrightarrow{xb_{2,G+1}}, \dots, \overrightarrow{xb_{N,G+1}}\} = \{\overrightarrow{mb_{1,G}}, \overrightarrow{mb_{2,G}}, \dots, \overrightarrow{mb_{N-2,G}}, \overrightarrow{xb_{best,G}}, \overrightarrow{xb_{best,G}}\}$

9: $G = G + 1$;

10: **Stopping Criterion:** If the maximum number of generation numbers is reached, then stop and output the best individual in Pb_G; otherwise go to step 3.

4.2.1.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) algorithm was first introduced by Kennedy and Eberhart as an optimization technique for continuous problems in 1995 [37-39]. It is a kind of evolutionary computation technique motivated by the behavior of organisms such as fish schooling and bird flocking. Because of its simple and easy implementation, PSO has been widely used in a variety of optimization problems. In 1997, Kennedy and Eberhart adapted the standard continuous PSO algorithm to binary spaces [40]. However, this variant of the continuous PSO, termed the binary PSO (BPSO), has not been studied much.

In the PSO algorithm, there are two update functions: the velocity update function and the position update function [41]. Originally, the position was updated by combining its current position and velocity, but in BPSO, the position is updated by reflecting only the current velocity; Generally, the sigmoid function has been used to update the position in BPSO. Due to this characteristic, it seems that the velocity domain is a search space though an actual binary search space already exists.

For the feature selection problem, a modified PSO algorithm is proposed. In PSO, a population, also called a swarm, of candidate solutions are encoded as particles in the search space. PSO starts with the random initialization of a population of particles. Particles move in the search space to search for the optimal solution by updating the position of each particle based on the experience of its own and its neighbor particles [42]. During movement, the current position of particle i is represented by a vector $\overrightarrow{xb_{i,G}} = (x_{i,1,G}, \dots, x_{i,j,G}, \dots, x_{i,n,G})$, where n is the dimensionality of the search space. The velocity of particle i is represented as $\vec{v}_{i,G} = (v_{i,1,G}, \dots, v_{i,j,G}, \dots, v_{i,n,G})$. The best previous position of particle i is recorded as the personal best called $\overrightarrow{pbest_{i,G}}$ and the best position obtained by the swarm so far is the global best called $\overrightarrow{gbest_G}$. PSO searches for the optimal solution by updating the position and the velocity of each particle according to the following equations:

$$v_{i,j,G+1} = \omega v_{i,j,G} + c_1 r_1 (pbest_{i,j,G} - x_{i,j,G}) + c_2 r_2 (gbest_{j,G} - x_{i,j,G}) \tag{4.14}$$

$$x_{i,j,G+1} = \begin{cases} 0, & if \ \ rand \geq S(x_{i,j,G} + v_{i,j,G+1}) \\ 1, & if \ \ rand < S(x_{i,j,G} + v_{i,j,G+1}) \end{cases} \tag{4.15}$$

$$S(x_{i,j,G} + v_{i,j,G+1}) = \frac{1}{1+e^{-(x_{i,j,G}+v_{i,j,G+1})}} \tag{4.16}$$

where G denotes the Gth iteration in the evolutionary process. j denotes the jth dimension in the search space. ω is inertia weight, which is employed to control the impact of the previous velocities on the current velocity. c_1 and c_2 are acceleration constants. r_1 and r_2 are random values uniformly distributed in $[0,1]$. $(x_{i,j,G} + v_{i,j,G+1})$ is limited by a predefined interval $[-l_{max}, l_{max}]$.

$S(\cdot)$ is sigmoid transformation described in Figure 4.11.

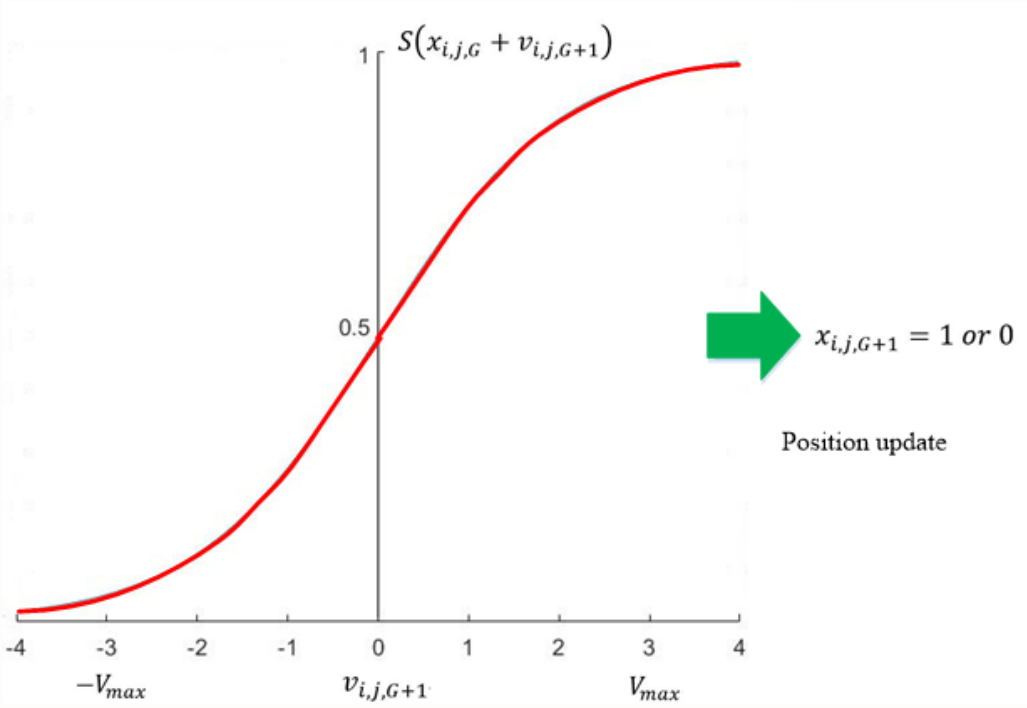


Figure 4.11 Sigmoid transformation

In the discrete version l_{max} is retained, that is, $|x_{i,j,G} + v_{i,j,G+1}| < l_{max}$, but as can be seen, this simply limits the ultimate probability that bit $x_{i,j,G+1}$ will take on a zero or one value. For instance, if $l_{max} = 4.0$, then probabilities will be limited to $S(v_{i,j,G+1})$, between 0.982 and 0.018. The result of this is that new vectors will still be tried, even after each bit has attained its best position. Specifying a higher l_{max}, e.g., 10.0, makes new vectors less likely. Thus part of the function of l_{max} in the discrete particle swarm is to set a limit to further exploration after the population has converged; in a sense, it could be said to control the ultimate mutation rate or temperature of the bit vector. Note also that, while high l_{max}, in the continuous-valued version increases the range explored by a particle, the opposite occurs in the binary version; smaller l_{max}, allows a higher mutation rate [40]. The following is the framework of PSO-FS.

Algorithm: PSO-FS

1: $G = 1$; /* G denotes the generation number */

2: Randomly generate an population Pb_G with N particles $\overrightarrow{xb}_{1,G}, \ldots, \overrightarrow{xb}_{N,G}$. For each particle $\overrightarrow{xb}_{i,G}$, initial the velocity $\vec{v}_{i,G}$ and the personal best position

$pbest_{i,G}$.

3: Evaluate each particle in Pb_G and obtain the global best position $gbest_G$.

4: $P_{G+1} = \emptyset$;

5: **For** each particle $\overrightarrow{xb}_{i,G}$ in Pb_G

6: Generate the updated velocity $\vec{v}_{i,G+1}$ by using Eq. 4.14; /*Velocity update*/

7: Generate the updated position $\overrightarrow{xb}_{i,G+1}$ by using Eq. 4.15; /*Position update*/

8: If $\vec{v}_{i,G+1}$ is not feasible, use a repair operator to make $\vec{v}_{i,G+1}$ feasible; /*Repair*/

9: Evaluate the updated position $\overrightarrow{xb}_{i,G+1}$ by the fitness function. According to the fitness value $f(\overrightarrow{xb}_{i,G+1})$, obtain the updated personal best position $\overrightarrow{pbest_{i,G+1}}$ and the updated global best position $\overrightarrow{gbest_{G+1}}$. /* $\overrightarrow{pbest}$ and $\overrightarrow{gbest}$ update*/

10: **End For**

11: $G = \mathrm{G} + 1$;

12: **Stopping Criterion:** If the maximum number of generation numbers is reached, then stop and output the best individual in Pb_G; otherwise go to step 4.

4.2.2 Multi-Objective Evolutionary Algorithms

Feature selection involves two main objectives, which are to minimize the evaluation metric and minimize the number of features. They are often conflicting objectives. Therefore, feature selection can be treated as a multi-objective problem to find a set of trade-off solutions between these two objectives. The research on this direction has gained much attention only in recent years, where EC techniques contribute the most since EC techniques using a population based approach are particularly suitable for multi-objective optimization [20][43].

4.2.2.1 Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D)

A multi-objective optimization problem (MOP) can be mathematically formulated as

$$\begin{cases} \text{minimize} \quad \mathrm{F}(\vec{x}) = \left(f_1(\vec{x}), \ldots, f_m(\vec{x})\right)^T \\ \text{s.t.} \quad \vec{x} \in \Omega \end{cases} \tag{4.17}$$

where Ω is the decision space and $\vec{x} \in \Omega$ is a decision vector. $\mathrm{F}(\vec{x})$ consists of m objective functions $f_i: |\Omega \to R,\ i = 1, \ldots, m$, where R^m is the objective space.

The objectives in Eq. (4.17) often conflict with each other. Improvement of one objective may lead to deterioration of another. Thus, a single solution, which can optimize all objectives simultaneously, does not exist. Instead, the best trade-off solutions, called the Pareto optimal solutions, are important to a decision maker (DM). The Pareto optimality concept, which was first proposed by Edgeworth and Pareto [44], is formally defined as follows [45][49].

- ✓ Definition 1. A vector $\vec{u} = (u_1, \ldots, u_m)^T$ is said to dominate another vector $\vec{v} = (v_1, \ldots, v_m)^T$, denoted as $\vec{u} \prec \vec{v}$, if $\forall i \in \{1, \ldots, m\},\ u_i \leq v_i$ and $\vec{u} \neq \vec{v}$.
- ✓ Definition 2. A feasible solution $\overrightarrow{x^*} \in \Omega$ of problem (1) is called a Pareto optimal solution, if $\nexists \vec{y} \in \Omega$ such that $F(\vec{y}) \prec F\left(\overrightarrow{x^*}\right)$. The set of all the Pareto optimal solutions is called the Pareto set (PS), denoted as $\mathrm{PS} = \{\vec{x} \in \Omega | \nexists \vec{y} \in \Omega, F(\vec{y}) \prec F(\vec{\mathrm{x}})\}$.

The image of the PS in the objective space is called the Pareto front (PF), $\mathrm{PF} = \{F(\vec{\mathrm{x}}) | \vec{x} \in \mathrm{PS}\}$.

For the feature selection, it is needed to provide users the trade-off between the two objectives (performance metric and number of features). To achieve this, it is thought to use evolutionary multi-objective algorithms to evolve a set of Pareto front solutions (feature subsets), which allows decision-makers to choose a preferred solution according to their own requirements. As a representative method of the decomposition-based approaches [46], MOEA/D has been widely applied. MOEA/D explicitly decomposes the multi-objective problem (MOP) into scalar optimization subproblems. It solves these subproblems simultaneously by evolving a population of solutions. At each generation,

the population is composed of the best solution found so far (i.e. since the start of the run of the algorithm) for each subproblem. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors. The optimal solutions to two neighboring subproblems should be very similar. Each subproblem (i.e., scalar aggregation function) is optimized in MOEA/D by using information only from its neighboring subproblems [47][48].

Due to MOEA/D requires a decomposition approach for converting approximation of the PF of MOP into a number of single objective optimization problems. In principle, any decomposition approach can serve for this purpose. Three different decomposition approaches have been used widely. In this paper, we use the Tchebycheff approach [49]. A single objective optimization subproblem in this approach is

$$\begin{cases} mininmize \;\; g(x|\lambda, z^*) = \max\limits_{1\le i\le m}\{\lambda_i|f_i(x) - z_i^*|\} \\ subject \;\; to \;\; x \in \Omega \end{cases} \tag{4.18}$$

where $\lambda = (\lambda_1, \dots, \lambda_m)$ is a weight vector, i.e., $\lambda_i \geq 0$ for all $i = 1, \dots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. $z^* = (z_1^*, \dots, z_m^*)$ is the reference point, i.e., $z^* = min\{f_i(x), x \in \Omega\}$ for each $i = 1, \dots, m$.

It is well known that, under mild conditions, for each Pareto optimal point there exists a weight vector λ such that it is the optimal solution of Eq. (4.18) and each optimal solution of Eq. (4.18) is a Pareto optimal solution of Eq. (4.17). Let $\lambda^1, \dots, \lambda^N$ be a set of weight vectors. Correspondingly, we have N single objective optimization subproblems where the ith subproblem is Eq. (4.18) with $\lambda = \lambda^i$. If N is reasonably large and $\lambda^1, \dots, \lambda^N$ are properly selected, then the optimal solutions to these subproblems will provide a good approximation to the PF or PS of Eq. (4.17).

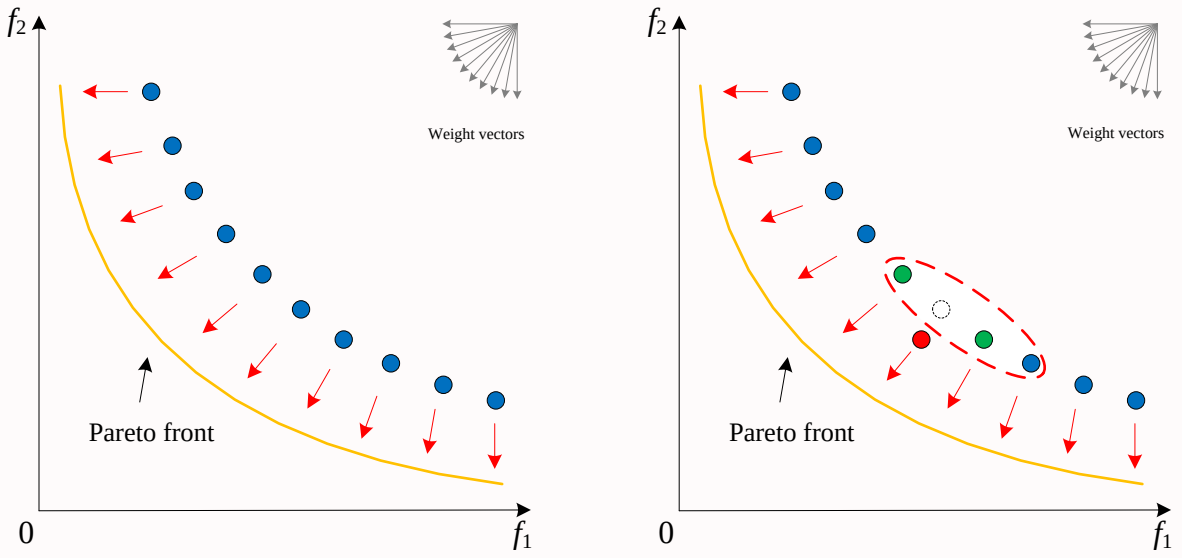


Figure 4.12 A simple graphical representation of MOEA/D

MOEA/D attempts to optimize these single objective optimization subproblems simultaneously instead of solving MOP directly. In MOEA/D, the T closest weight vectors in $\{\lambda^1, \dots, \lambda^N\}$ to a

weight vector λ^i constitute the neighborhood of λ^i. The neighborhood of the ith subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. Since $g(x|\lambda, z^*)$ is continuous of λ, the optimal solutions of neighboring subproblems should be close in the decision space. MOEA/D exploits the neighborhood relationship among the subproblems for making its search effectively and efficiently. A simple graphical representation of MOEA/D is depicted in Figure 4.12.

As a discrete optimization problem, feature selection problem can't be solved by MOEA/D directed. To overcome these shortcomings, this paper proposes a new implementation of MOEA/D, called MOEA/D for dealing with discrete MOPs. MOEA/D-BFDE uses a binary feedback differential evolution (DE refer to 4.2.1.2) operator for producing new solutions. The following is the pseudocode of MOEA/D-BFDE.

Algorithm: MOEA/D-BFDE

1: $G = 1$; /* G denotes the generation number */

2: Randomly generate an initial population $P_G = \{\vec{x}_{1,G}, \dots, \vec{x}_{N,G}\}$ and obtain the binary-digits population $Pb_G = \{\overrightarrow{xb}_{1,G}, \dots, \overrightarrow{xb}_{N,G}\}$ by the binary approach;

3: Evaluate each individual in Pb_G and use $FV^i = F(\overrightarrow{xb}_{i,G})$ as the fitness value of $\vec{x}_{i,G}$.

4: Initialize $z = (z_1, z_2, \dots, z_m)^T$ by setting $z_j = \min_{1 \le i \le N} f_j(\overrightarrow{xb}_{i,G})$

5: $P_{G+1} = \emptyset$;

6: **For** each individual $\vec{x}_{i,G}$ (also called a target vector) in P_G

7: **Selection of Mating/Update Range:** Set $P = \begin{cases} B(i) & if\ rand < \delta \\ \{1, \dots, N\} & otherwise \end{cases}$

8: **Reproduction:** Generate a mutant vector $\vec{v}_{i,G}$ by using the DE/rand/1 mutation operator and incorporate binary-digits information to $\vec{v}_{i,G} = \vec{v}_{i,G} - (1 - \overrightarrow{xb}_{i,G})$.

9: **Repair:** If $\vec{v}_{i,G}$ is not feasible (i.e., not in the search space), use a repair operator to make $\vec{v}_{i,G}$ feasible;

10: Mix $\vec{x}_{i,G}$ and $\vec{v}_{i,G}$ to generate a trial vector $\vec{u}_{i,G}$ by using the binomial crossover operator;

11: **Update of z:** For each $j = 1, \dots, m$, if $z_j > f_j(\overrightarrow{ub}_{i,G})$, then set $z_j =$

$f_j(\overrightarrow{ub}_{i,G})$;

12: **Update of Solutions:** Set $c = 0$ and then do the following:

13: If $c = n_r$ or P is empty, go to Step 16. Otherwise, randomly pick an index j from P.

14: If $\mathrm{g}(\vec{u}_{i,G}|\lambda^j, z) \leq \mathrm{g}(\vec{x}_{j,G}|\lambda^j, z)$, then set $\vec{x}_{j,G} = \vec{u}_{j,G}$, $FV^j = \mathrm{F}(\vec{u}_{j,G})$ and $c = c + 1$.

15: Remove j from P and go to Step 13.

16: **End For**

17: $G = G + 1$

18: **Stopping Criterion:** If the maximum number of generation numbers is reached then stop and output the population Pb_G and $\{F(\overrightarrow{xb}_{1,G}), \ldots, F(\overrightarrow{xb}_{N,G})\}$; otherwise go to step 3.

4.2.2.2 Non-dominated Sorting Genetic Algorithm-II (NSGA-II)

For the feature selection problem, the representation used here is the same as described in 4.2.1.3 Genetic Algorithm, which is a n-bit string, where n is the total number of features in the dataset. As a representative method of the Pareto domination-based approaches, NSGA-II has been widely used [45][50]. In the following, we present the nondominated sorting GA approach, which uses a fast nondominated sorting procedure, an elitist-preserving approach, and a parameterless niching operator.

First, for each solution we calculate two entities: 1) domination count n_p, the number of solutions which dominate the solution p, and 2) S_p, a set of solutions that the solution p dominates. All solutions in the first nondominated front will have their domination count as zero. Now, for each solution p with $n_p = 0$, we visit each member (q) of its set S_p and reduce its domination count by one. In doing so, if for any member q the domination count becomes zero, we put it in a separate list Q. These members belong to the second nondominated front. Now, the above procedure is continued with each member of Q and the third front is identified. This process continues until all fronts are identified. Figure 4.13 shows the process of sorting according to different nondomination levels.

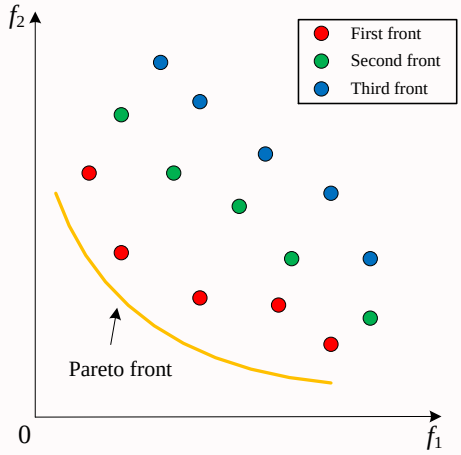


Figure 4.13 Nondominated sorting procedure

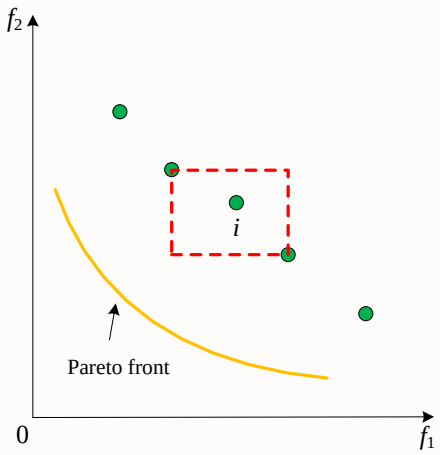


Figure 4.14 Crowding-distance calculation

Along with convergence to the Pareto-optimal set, it is also desired that an EA maintains a good spread of solutions in the obtained set of solutions. To get an estimate of the density of solutions surrounding a particular solution in the population, we calculate the average distance of two points on either side of this point along each of the objectives. This quantity $i_{distance}$ serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices (call this the crowding distance). In Figure 4.14, the crowding distance of the ith solution in its front (marked with solid circles) is the average side length of the cuboid (shown with a dashed box).

The crowded-comparison operator $(\prec_n)$ guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto-optimal front. Assume that every individual i in

the population has two attributes: 1) Nondomination rank (i_{rank}); 2)Crowding distance $(i_{distance})$. We now define a partial order $\prec_n$ as

$$i \prec_n j \quad if \ (i_{rank} < j_{rank}) \ or \ ((i_{rank} = j_{rank}) \ and \ (i_{distance} > j_{distance}))$$

That is, between two solutions with differing nondomination ranks, we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a lesser crowed region.

For completeness, a brief description of the NSGA-II algorithm is presented in Figure 4.15. Suppose the parent population at tth generation is P_t and its size is N, while the offspring population created from P_t is Q_t having N members. The first step is to choose the best N members from the combined parent and offspring population $R_t = P_t \cap Q_t$ (of size 2N), thus allowing us to preserve elite members of the parent population. To achieve this, first the combined population R_t is sorted according to different nondomination levels (F1, F2, and so on). Then, each nondomination level is selected one at a time to construct a new population S_t, starting from F1, until the size of S_t is equal to N or for the first time exceeds N. Let us say the last level included is the lth level. Thus, all solutions from level $(l+1)$ onward are rejected from the combined population R_t. In most situations, the last accepted level (lth level) is only accepted partially. In such a case, only those solutions that will maximize the diversity of the lth front are chosen. In NSGA-II, this is achieved through a computationally efficient, yet approximate, niche-preservation operator that computes the crowding distance for every last level member as the summation of objective-wise normalized distance between two neighboring solutions. Thereafter, the solutions that have larger crowding distance values are chosen [51][52].

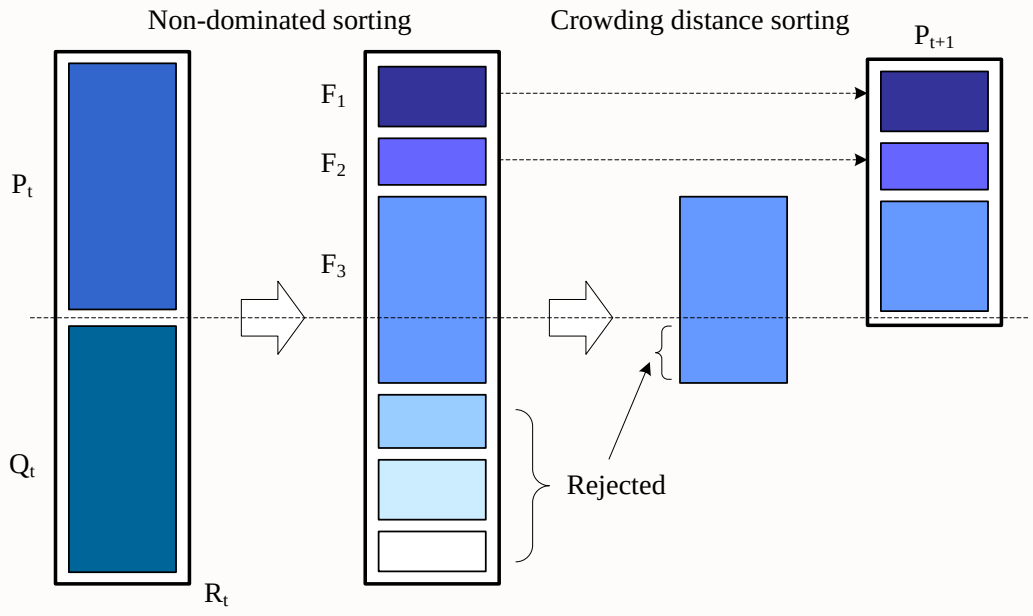


Figure 4.15 NSGA-II procedure

The following is the main loop of NSGA-II.

Main Loop
1: $R_t = P_t \cup Q_t$
2: F = fast-non-dominated-sort(R_t)
3: $P_{t+1} = \emptyset$ and $i = 1$
4: until $\lvert P_{t+1}\rvert + \lvert F_i\rvert \leq N$
5: crowding-distance-assignment(F_i)
6: $P_{t+1} = P_{t+1} \cup F_i$
7: $i = i + 1$
8: Sort($F_i, \prec_n$)
9: $P_{t+1} = P_{t+1} \cup F_i[1:(N - \lvert P_{t+1}\rvert)]$
10: Q_{t+1}=make-new-pop(P_{t+1})
11: $t = t + 1$

4.2.3 Parallel Execution

With rapid development of the information age and the emergence of “big data”, feature selection problem gradually becomes to be computationally and data-intensive problem. Therefore, parallel strategy is a useful approach which makes full use of the processing power of multicore desktops, and leads to a significant reduction in the processing time. Given all that, the parallel mechanism is applied to accelerate the evolutionary algorithms [53][54].

4.3 Subset Evaluation: Metrics

Feature subsets produced by the search procedure will be examined by an evaluation function to determine their goodness. The evaluation function plays an important role in a feature selection algorithm, because it helps guide the algorithm to search for the optimal feature subset. Considering that validation metrics are the ultimate criteria to judge the quality of the validated models, we apply it as the evaluation function. For better understanding, we have divided validation metrics into two categories: (a) validation metrics for regression-based models, and (b) validation metrics for classification-based models. Figure 4.16 shows the overview of the Subset Evaluation: Metrics.

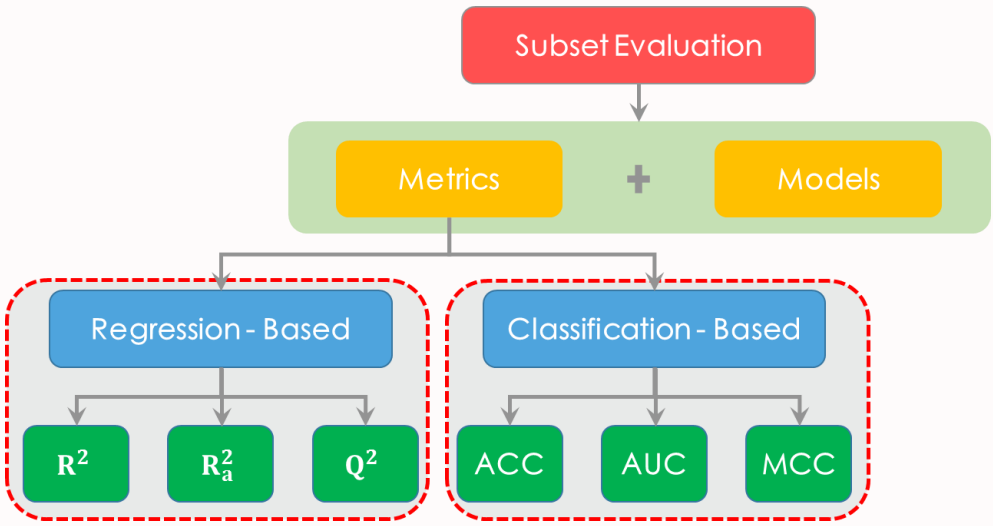


Figure 4.16 The overview of the Subset Evaluation: Metrics

4.3.1 Metrics for Regression-Based Models

4.3.1.1 Determination coefficient (R^2)

In order to judge the fitting ability of a model, we can consider the average of the observed Y values ($\bar{Y}_{obs}$) as the reference, such that the model performance should be more than $\bar{Y}_{obs}$. For a good model, the residual values (or the sum of squared residuals) should be small, while the deviation of most of the individual observed Y values from $\bar{Y}_{obs}$ is expected to be high. Thus, the ration $\frac{\sum(Y_{obs}-Y_{calc})^2}{\sum(Y_{obs}-\bar{Y}_{obs})^2}$ should have a low value for a good model. We can define the determination coefficient (R^2) in the following manner:

$$R^2 = 1 - \frac{\sum(Y_{obs}-Y_{calc})^2}{\sum(Y_{obs}-\bar{Y}_{obs})^2} \tag{4.19}$$

For the ideal model, the sum of squared residuals being 0, the value of R^2 is 1. As the value of R^2 deviates from 1, the fitting quality of the model deteriorates. The square root of R^2 is the

multiple correlation coefficient (R).

4.3.1.2 Adjusted R^2 (R_a^2)

If we examine the expression of the determination coefficient, we can see that it only compares the calculated Y values with the experimental ones, without considering the number of features in the model. If one goes on increasing the number of features in the model for a fixed number of observations, R^2 values will always increase, but this will lead to a decrease in the degree of freedom and low statistical reliability. Thus, a high value of R^2 is not necessarily an indication of a good statistical model that fits the available data. If, for example, one uses 100 features in a model for 100 observations, the resultant model will show $R^2 = 1$, but it will not have any reliability, as this will be a perfectly fitted model (a solved system) rather than a statistical model. For a reliable model, the number of observations and number of features should bear a ration of at least 5:1. Thus, to better reflect the explained variance (the fraction of the data variance explained by the model), a better measure is adjusted R^2, which is defined in the following manner:

$$R_a^2 = \frac{(m-1)\times R^2 - n}{m-1-n} \tag{4.20}$$

In Eq. (4.20), n is the number of predictor variables used in model development. For a model with a given number of observations (m), as the number of predictor variables increases, the value of R^2 increases, while the adjusted R^2 value is penalized due to an increase in the number of predictor variables.

4.3.1.3 Cross-validated Q^2

The cross-validation technique mainly involves internal validation [56][57], where an example of m samples is partitioned into calibration (i.e., training) and validation (i.e., test) subsets. The calibration subset is used to construct a model, while the validation subset is used to test how well the model predicts the new data-that is, the data points not used in the calibration procedure. To judge the quality and goodness-of-fit of the model, internal validation is ideal. As a metric for internal validation, the key steps for the calculation of leave-one-out cross-validation (LOO-CV) are described in Figure 4.17.

To achieve leave-one-out cross-validation (LOO-CV), the training data set is primarily modified by removing one sample from the set. The model is then rebuilt based on the remaining samples of the training set using the feature combination originally selected, and the value of the deleted sample is measured based on the resulting equation. This cycle is repeated until all the samples of the training set have been deleted once, and the predicted value data obtained for all the training set samples are used for the calculation of various internal validation parameters. Finally, model predictive accuracy is judged using the predicted residual sum of squares (PRESS) and cross-validated Q^2 [58] for the model. PRESS is the sum of squared differences between experimental and LOO predicted data. Eq.

(4.21) and Eq. (4.22) give the expressions for PRESS and Q^2, respectively:

$$\text{PRESS} = \sum\left(Y_{obs} - Y_{pred}\right)^2 \tag{4.21}$$

$$Q^2 = 1 - \frac{\sum\left(Y_{obs(train)} - Y_{pred(train)}\right)^2}{\sum\left(Y_{obs(train)} - \bar{Y}_{train}\right)^2} = 1 - \frac{\text{PRESS}}{\sum\left(Y_{obs(train)} - \bar{Y}_{train}\right)^2} \tag{4.22}$$

where Y_{obs} and Y_{pred} correspond to the observed and LOO predicted values, while $Y_{obs(train)}$ is the observed value, $Y_{pred(train)}$ is the predicted value of the training set based on the LOO technique.

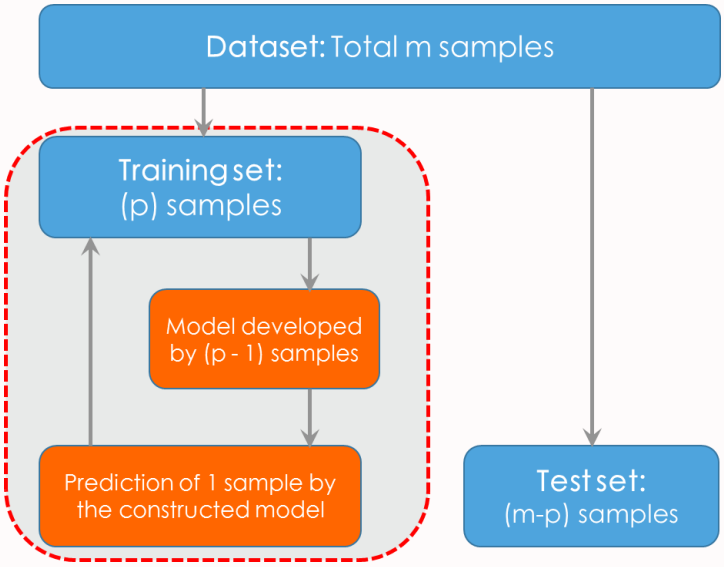


Figure 4.17 Key steps for the calculation of LOO-CV

Note: In n-fold cross-validation, a sample of n observations is randomly partitioned into n-folds (partitions) and the folds are near-equal size. Subsequently, a single fold of the n folds is retained as the test set for testing the model, and the remaining (n - 1) folds are used as the training set. The cross-validation process is then repeated n times, with each of the n folds used only once as the test set. The n results from the n experiments are then averaged to produce a single estimate of the model performance. The advantage of such a method is that all instances are used for both training and testing, and each instance is used for testing only once. Generally, a larger n will produce an estimate with smaller bias because of the higher proportion of instances in the training set, but potentially higher variance (on top of being computationally expensive) [55]. Leave-one-out cross-validation (LOO-CV) is an extreme case of n-fold cross-validation, which uses a single observation from the dataset as the test set, and the remaining observations as the training set. This is the same as a n-fold cross-validation with n being equal to the total number of observations in the dataset. Note that n-fold cross-validation is usually used when the number of observations in the entire dataset is small.

4.3.2 Metrics for Classification-Based Models

Validation metrics can assess the performance of the classification-based models in terms of accurate qualitative prediction of the dependent variable [59]. The validation for classification model is usually executed for two-class problems, where the samples are categorized either as positive or as negative.

4.3.2.1 Sensitivity, specificity and accuracy

The samples classified employing the classification-based model can be divided into four categories based on a comparison between the predicted and observed values: (1) true positives (TPs), the positive samples that have been correctly predicted as positive based on the developed models; (2) false positives (FPs), which include the negative samples that have been erroneously classified as positive; (3) false negatives (FNs), which comprise the positive samples wrongly classified as negative; and (4) true negatives (TNs), which account for the negative samples that have been accurately predicted as negative by the models under validation [60]. Based on this classification and the number of test-set samples, a two-by-two confusion matrix [61], also referred to as the contingency table (confusion matrix), may be constructed that corresponds to the dispositions of the samples under consideration. This is a matrix with two rows and two columns that reports the number of samples belonging to each of the four classes. To evaluate the classifier model performance and classification capability, a number of statistical tests have been employed by several researchers:

$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{4.23}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \tag{4.24}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \tag{4.25}$$

4.3.2.2 Matthews correlation coefficient

The Matthews correlation coefficient (MCC) [62] is utilized as a measure of the quality of binary classifications. It considers true and false positives and negatives and is generally regarded as a balanced measure that can be used even if the classes have different sizes. The MCC is simply a correlation coefficient between the observed and predicted binary classifications, and it returns a value between -1 and 1. A coefficient of 1 indicates a perfect prediction, 0 an average random prediction, and -1 an inverse prediction. The MCC can be calculated directly from the confusion matrix using the following formula:

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{4.26}$$

In Eq. (4.26), TP is the number of true positives, TN the number of true negatives, FP the number of false positives, and FN the number of false negatives. If any of the four sums in the denominator is zero, the denominator can be arbitrarily set to 1; this results in a Matthews correlation coefficient of zero, which can be shown to be the correct limiting value. This metric in particular addresses the issue of improper explanation of a confusion matrix and the cases where the data set sizes are higher.

4.3.2.3 AUC-ROC

Area under curve-receiver operating characteristics (AUC-ROC) is equivalent to a simple average of the ranks of the positive samples; the good performance of early recognitions is offset quickly by late recognitions [63]. Let m be the number of positive samples and M be the total number of samples; in that case, AUC-ROC is approximately normally distributed, with mean $\mu = 1/2 + 1/2(M - m)$ and variance. $\sigma^2 = M + 1/12m(M - m)$. AUC-ROC, as defined in Eq. (4.27), is linearly related to the rank sum of positive samples, which is also called the *Mann-Whitney U* test.

$$\mathrm{AUC}-\mathrm{ROC} = 1 - \frac{\sum_{i=1}^{m} r_i}{m\times(M-m)} + \frac{m+1}{2\times(M-m)} \tag{4.27}$$

In this expression, r_i is the rank of the ith positive sample.

4.4 Subset Evaluation: Models

Models are developed using one or more statistical model building tools, which may be broadly categorized into regression- and classification-based approaches. Figure 4.18 shows the overview of the Subset Evaluation: Models.

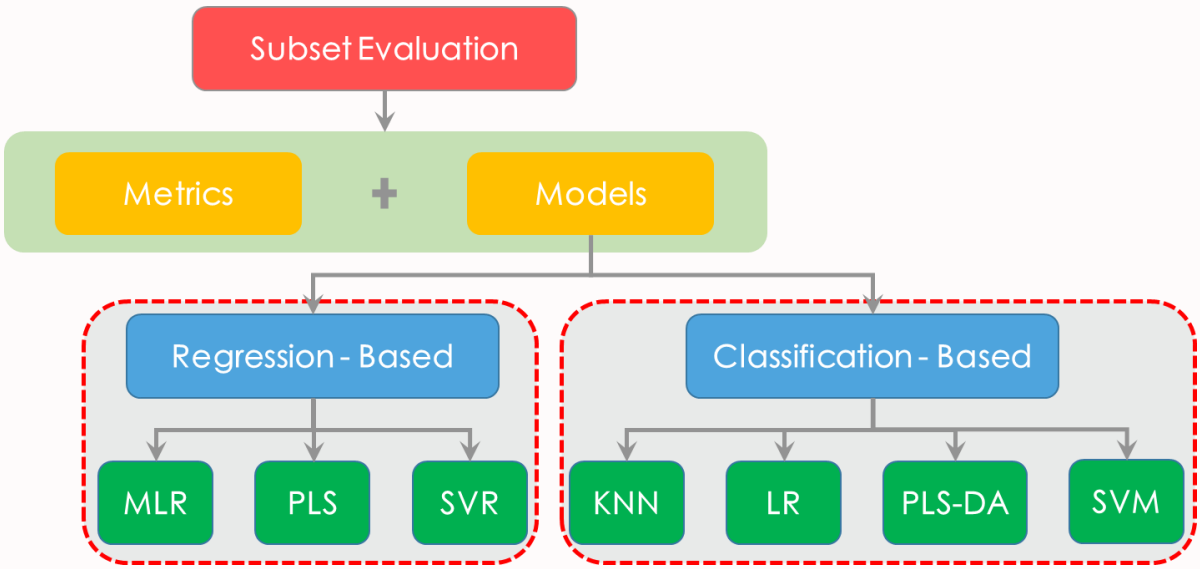


Figure 4.18 The overview of the Subset Evaluation: Models

4.4.1 Regression-Based Approaches

Regression-based approaches are used when the response variable is quantitative. The generated regression model can compute quantitative response data from the model. And Table 4.1 is a representative table for regression.

Table 4.1 A representative table for regression

Sample No.	Response (Y)	Feature 1 (X_1)	Feature 2 (X_2)	Feature 3 (X_3)	...	Feature n (X_n)
1	$(Y)_1$	$(X_1)_1$	$(X_2)_1$	$(X_3)_1$	...	$(X_n)_1$
2	$(Y)_2$	$(X_1)_2$	$(X_2)_2$	$(X_3)_2$	...	$(X_n)_2$
3	$(Y)_3$	$(X_1)_3$	$(X_2)_3$	$(X_3)_3$	...	$(X_n)_3$
4	$(Y)_4$	$(X_1)_4$	$(X_2)_4$	$(X_3)_4$	...	$(X_n)_4$
...	...	...	...	...	...	...
m	$(Y)_m$	$(X_1)_m$	$(X_2)_m$	$(X_3)_m$	...	$(X_n)_m$

4.4.1.1 Multiple linear regression (MLR)

Multiple linear regression (MLR) [64] is one of the most popular methods due to its simplicity in

operation, reproducibility, and ability to allow easy interpretation of the features used. This is a regression approach of the dependent variable on more than one feature. The generalized expression of an MLR equation is as follows:

$$Y = a_0 + a_1 \times X_1 + a_2 \times X_2 + a_3 \times X_3 + \cdots + a_n \times X_n \quad (3.28)$$

In Eq. (3.28), Y is the response or dependent variable; $X_1, X_2, \ldots, X_n$ are features (independent variables) present in the model with the corresponding regression coefficients $a_1, a_2, \ldots, a_n$, respectively; and a_0 is the constant term of the model. The interpretation of contribution of individual features $X_1, X_2, \ldots, X_n$ is straightforward depending on the corresponding coefficient value and its algebraic sign.

4.4.1.2 Partial least squares (PLS)

While handling a large number of inter-correlated and noisy features for a limited number of data points, Partial least squares (PLS) [65] is a better choice over MLR. PLS, being a generalization of MLR, tries to extract the latent variables (LV), which are functions of the original variables, accounting for as much of the underlying factor variation as possible while modeling the responses. Before the analysis, the X- and Y-variables are often transformed to make their distributions fairly symmetrical. The response variables are usually logarithmically transformed and the X variables should be scaled appropriately. The linear PLS finds a few new variables (latent variables), which are linear combinations of the original variables. When the number of LVs is equal to the number of variables, the PLS model becomes same as the MLR model. A strict test of the predictive significance of each PLS component is necessary, and then stopping addition of new components when components start to be non-significant. Cross-validation (CV) is a practical and reliable way to test this predictive significance. A PLS equation can be expressed in the same form as in MLR; thus contributions of individual features to the response can be easily found out.

4.4.1.3 Support vector regression (SVR)

Support vector machine (SVM) is a popular machine learning tool for classification and regression, first identified by Vladimir Vapnik and his colleagues in 1992 [66]. A detailed description of the theory of SVM can be referred in several excellent books and tutorials [67][68]. SVMs are originally developed for classification problems; they can also be extended to solve nonlinear regression problems by the introduction of ε-insensitive loss function. And support vector regression (SVR) has shown promising capability for solving a number of regression problems. In SVR, the input x is first mapped into a higher dimensional feature space by the use of a kernel function, and then a linear model is constructed in this feature space. The kernel functions often used in SVR include linear, polynomial, radial basis function, and sigmoid function. The quality of estimation is measured by a

loss function known as ε-insensitive loss function. The advantages of SVRs include that they can be used to avoid the difficulties of using linear functions in the high-dimensional feature space, and the optimization problem is transformed into dual convex quadratic programs.

The generalization performance of SVR depends on a good setting of parameters: C, ε and the kernel type and corresponding kernel parameters. The selection of the kernel function and corresponding parameters is very important because they define the distribution of the training set samples in the high dimensional feature space. Parameter C is a regularization constant which determines the trade-off between the model complexity and the degree to which deviations larger than ε are tolerated in an optimization formulation.

4.4.2 Classification-Based Approaches

Classification-based approaches are used when the response variable is Label (like positive-negative). And Table 4.2 is a representative table for Classification.

Table 4.2 A representative table for classification

Sample No.	Label (Y)	Feature 1 (X_1)	Feature 2 (X_2)	Feature 3 (X_3)	…	Feature n (X_n)
1	$(Y)_1$	$(X_1)_1$	$(X_2)_1$	$(X_3)_1$	…	$(X_n)_1$
2	$(Y)_2$	$(X_1)_2$	$(X_2)_2$	$(X_3)_2$	…	$(X_n)_2$
3	$(Y)_3$	$(X_1)_3$	$(X_2)_3$	$(X_3)_3$	…	$(X_n)_3$
4	$(Y)_4$	$(X_1)_4$	$(X_2)_4$	$(X_3)_4$	…	$(X_n)_4$
…	…	…	…	…	…	…
m	$(Y)_m$	$(X_1)_m$	$(X_2)_m$	$(X_3)_m$	…	$(X_n)_m$

4.4.2.1 K Nearest Neighbor (KNN)

The K Nearest Neighbor (KNN) [69] is a type of instance-based learning algorithm. When using KNN for classification, it calculates the distances between an instance in the test set and every instance in the training set. KNN assigns the test instance to the class that is the most common amongst its k nearest neighbors, where k is a positive integer, typically small. If $k = 1$, the test instance is simply assigned to the class of the single nearest neighbor. Euclidean distance, Manhattean distance, Minkowski distance and other distance measures can be used to measure the distance between the test instance and the training instances in KNN [70]. In KNN, there is no explicit training process or it is very minimal. In other words, KNN does not use the training data points to do any generalization. The training data in KNN is needed during the testing process, which is in contrast to other techniques like SVM, where the training set and all non-support vectors (hyperplanes) can be safely discarded. KNN does not make any assumptions on the underlying data distribution. In real-

world applications, most of the datasets do not obey the typical theoretical assumptions (e.g. Gaussian mixtures, linearly separable or independent features), which are needed in certain learning algorithm [70]. Therefore, KNN is a simple learning algorithm, but works well in practice. However, for a large training set, KNN requires large memory and is very time-consuming to make a decision [71].

4.4.2.2 Logistic Regression (LR)

Logistic regression [72] is a statistical classification model that measures the relationship between a categorical-dependent variable (having only two categories) and one or more independent variables, which are usually (but not necessarily) continuous, by using probability scores as the predicted values of the dependent variable. Logistic regression does not assume a linear relationship between the dependent and independent variables. The independent variables need neither be normally distributed, nor linearly related, nor of equal variance within each group. The form of the logistic regression equation is

$$\text{logit}[p(x)] = log\left[\frac{p(x)}{1-p(x)}\right] = a + b_1x_1 + b_2x_2 + b_3x_3 + \cdots \quad (3.29)$$

where $\text{logit}[p(x)]$ is the log (to base e) of the likelihood ratio that the dependent variable is 1, and p can only range from 0 to 1. In Eq. (3.29), a is the model's intercept, and $X_1, \ldots, X_k$ are molecular descriptors with their corresponding regression coefficients $b_1, \ldots, b_k$ (for molecular descriptors 1 through k). For an unknown sample, LR calculates the probability that the sample belongs to a certain target Label. LR estimates the probability of the sample being a positive sample. If the calculated $\text{logit}[p(x)]$ is greater than 0.5, then it is more probable that the sample is positive. Similar to MLR, the regression coefficients in LR can describe the influence of a feature on the outcome of the prediction. When the coefficient has a large value, it shows that the feature strongly affects the probability of the outcome, whereas a zero value coefficient shows that the feature has no influence on the outcome probability. Likewise, the sign of the coefficients affects the probability as well; that is, a positive coefficient increases the probability of an outcome while a negative coefficient will result in the opposite.

4.4.2.3 Partial Least Squares Discriminant Analysis (PLS-DA)

Partial Least Squares Discriminant Analysis (PLS-DA) is a linear classification method that combines the properties of partial least squares regression with the discrimination power of a classification technique. PLS-DA is based on the PLS regression algorithm which searches for latent variables with a maximum covariance with the Y-variables [73]. When dealing with PLS-DA, the class vector (containing the membership of samples to the G classes) is transformed into a dummy matrix Y, with n rows (samples) and G columns (the class information). Each entry y_{ig} of Y represents the membership of the ith sample to the gth class expressed with a binary code (1 or 0). Therefore, the

n-dimensional class vector is transformed into a binary Y matrix constituted by n rows and G columns.

The PLS regression model is then calibrated on the Y matrix in the usual way [74]. Thus, PLS-DA returns estimated values (y_{ig}^{calc}) for each ith sample and for each gth class. The estimated class values will not have either 1 or 0 values perfectly; however, if y_{ig}^{calc} is closer to zero, then the ith sample does not likely belong to the gth class, while a value closer to one would indicate the opposite. To make a class assignment, the probability that a sample belongs to a specific class can be calculated on the basis of the estimated class values [75][76]. Therefore, a probability is calculated for each class and classification of samples is carried out by choosing the class that has the highest probability. Under this approach, samples are always classified in one of the classes. On the other hand, a threshold can be defined for each class: if y_{ig}^{calc} is greater than the threshold defined for the gth class, then the ith sample is assigned to the gth class, otherwise not.

4.4.2.4 Support Vector Machine (SVM)

Support vector machines (SVMs) are a popular machine learning method. They are based on the concept of decision planes that define decision boundaries. The main idea of SVMs is to use a kernel function to map the input data to a higher-dimensional space, where the instances are linearly separable. In the high-dimensional space, SVMs construct a hyperplane or a set of hyperplanes, which are used to create decision boundaries for classification [77]. SVMs are inherently two-class classifiers. Each hyperplane is expected to separate between a set of instances having two classes. Instances are classified based on what side of these hyperplanes they fall on. SVMs aim to maximize the distances between the hyperplanes and the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier [77].

SVMs were primarily designed for binary classification. Different methods have been developed to use SVMs for multiple (C) class classification [78]. A common way is to build C "one-versus-rest" classifiers (commonly referred to as "one-versus-all" classification) [79], and to choose the classifier that classifies the instances with the greatest margin. Another way is to build a set of "one-versus-one" (binary) classifiers [80], and to choose the class that is selected by the most classifiers. A particular advantage of SVMs over other learning algorithms is that they are based on sound mathematics theory and can be analyzed theoretically using concepts from the computational learning theory [81]. From a practical point of view, the most serious disadvantage of SVMs is the high algorithmic complexity and extensive memory requirements in large-scale tasks [82].

4.5 Performance Analysis and Comparison

In ECoFFeS, we present two novel algorithms (modified DE and modified MOEA/D, namely BFDE and MOEA/D-BFDE, proposed by the authors) to choose the optimal feature subset. In order to verify the validity of the algorithms, some contrast experiments were conducted in this section.

4.5.1 Benchmark Datasets

In this study, three quantitative structure-activity/property relationship datasets are used for demonstrating the effectiveness of our proposed algorithms (BFDE and MOEA/D-BFDE). The first is artemisinin dataset, which consists of 211 artemisinin analogues [83]. Due to the fact that this dataset has many enantiomeric pairs of activities, the element in each enantiomeric pair with smaller logarithm of the relative activity is used as the output variable (referred as log RA), and the other element is removed. Therefore, the artemisinin dataset used in this paper has 178 compounds. As pointed out in [84], several structural diverse compounds have the same log RA (i.e., -4.0), which makes the model development more difficult. For each compound, two-dimensional (2D) descriptors are calculated using ChemoPy software package [85], which is developed by our group. Note that before further descriptor selection, two descriptor preselection steps are performed to eliminate some uninformative descriptors: (1) remove the descriptors, the variance of which is near zero or zero and (2) if the correlation of two descriptors is larger than 0.95, then remove one of them. Finally, 89 molecular descriptors are obtained for representing compounds in the artemisinin dataset, and these molecular descriptors are used as inputs for QSAR/QAPR model development. These molecular descriptors include 18 constitutional descriptors, 32 topological structural descriptors, 27 electrotopological state (E-state) descriptors, 5 molecular property descriptors, 4 kappa descriptors, and 3 connectivity descriptors.

The second is benzodiazepine receptors (BZR) dataset. In the BZR dataset, benzodiazepines are a class of psychoactive drugs, which are used to treat anxiety, insomnia, and a range of other circumstances conditions. At the same time, benzodiazepines exhibit sedative, hypnotic, anti-anxiety, anticonvulsant, and muscle relaxant properties, and act via the BZR, which have been extensively researched in QSAR/QSPR [86]. The BZR dataset used in our study is presented in [87]. It contains 163 compounds and 75 2.5D descriptors consisting of S_sCH3, S_dssC, CHI-0, and so on.

The third is selwood dataset [88], which has become a benchmark to evaluate the performance of different methods and has been well-studied in QSAR/QSPR [89]. It consists of 29 compounds, 53 descriptors, and a set of corresponding antifilarial antimycin activities expressed as -log(IC50). The molecular descriptors in the selwood dataset include partial atomic charges for atoms 1–10

(ATCH1-ATCH10), dipole vector (DIPV_X, DIPV_Y, and DIPV_Z), dipole moment (DIPMOM), and so on.

The previous three datasets, together with calculated molecular descriptors [90], could be found in the following website (https://github.com/Jiaweihuang/ECoFFeS).

4.5.2 Analysis and Comparison

4.5.2.1 Binary Feedback Differential Evolution (BFDE)

The proposed BFDE includes two main features: (1) the binary approach and (2) incorporate the information of the binary-digits population to the numerical population. In order to assess the effectiveness of BFDE, six different methods are employed for comparison on the three datasets. These methods include the following: (1) PLS; (2) PSO-PLS [40]; (3) BFPSO-PLS, which is formed by combing the PSO-PLS with the proposed strategy (binary feedback strategy); (4) WS-PSO-PLS [90]; (5) BDE-PLS [30], which is formed by combining the PLS just with the binary approach; (6) BFDE-PLS, the difference between it and BDE-PLS is that BFDE-PLS not only use the binary approach, but also incorporate the information of the binary-digits population to the numerical population.

Table 4.3 The parameter values used in different methods

Methods	Parameters			
	Population size	Number of generations	Number of fitness evaluations	Number of runs
PLS	50	600	30000	30
WS-PSO-PLS	50	600	30000	30
PSO-PLS	150	200	30000	30
BFPSO-PLS	150	200	30000	30
BDE-PLS	150	200	30000	30
BFDE-PLS	150	200	30000	30

By comparing the performance of the involved methods, three performance metrics have been chosen: Q^2, the root mean square error from fivefold cross-validation (denoted as RMSECV) and the number of the selected descriptors (denoted as N). Here, the reason that we choose five-fold cross validation is as follows: Firstly, the five-fold cross-validation and the standard leave-one-out cross-validation are similar owing to that they both belong to the class of k-fold cross-validation. Secondly, in our previous trial and error experiments, we found that five-fold cross-validation has higher computational efficiency than that of the standard leave-one-out cross-validation. Thirdly, maybe you

think that the fivefold cross-validation is not as stable as the standard leave-one-out cross-validation. Note that all the involved methods have been implemented 30 times to obtain the statistical results. Thus, the experimental results are credible and steady. All the data were firstly auto-scaled to have zero mean and unit variance before modeling. Note that in PLS, all the descriptors were directly used for the model development. The parameter settings for all the involved methods have been listed in Table 4.3 and the experimental results have been presented in Table 4.4.

Table 4.4 Experimental results on the three datasets

Datasets	Methods	Mean Q^2 ± Standard deviation	Mean RMSECV ± Standard deviation	Mean N ± Standard deviation
Artemisinin	PLS	0.6003	0.9912	89
	PSO-PLS	0.7098 ± 0.0090	0.8446 ± 0.0131	16.0333 ± 3.6811
	BFPSO-PLS	0.7436 ± 0.0085	0.7938 ± 0.0131	22.0000 ± 3.4641
	WS-PSO-PLS	0.7588 ± 0.0090	0.7699 ± 0.0144	30.8333 ± 3.6774
	BDE-PLS	0.7481 ± 0.0117	0.7867 ± 0.01845	38.5000 ± 3.4114
	BFDE-PLS	**0.7594 ± 0.0072**	**0.7690 ± 0.0115**	**23.6000 ± 2.9196**
BZR	PLS	0.4007	0.8501	75
	PSO-PLS	0.5260 ± 0.0099	0.7560 ± 0.0079	15.9667 ± 3.8100
	BFPSO-PLS	0.5698 ± 0.0085	0.7203 ± 0.0072	20.2000 ± 2.3253
	WS-PSO-PLS	0.5652 ± 0.0058	0.7241 ± 0.0048	28.3667 ± 2.4138
	BDE-PLS	0.5735 ± 0.0133	0.7171 ± 0.0111	32.0667 ± 3.8321
	BFDE-PLS	**0.5863 ± 0.0087**	**0.7063 ± 0.0074**	**21.5000 ± 2.3599**
Selwood	PLS	0.2407	0.6461	53
	PSO-PLS	0.8943 ± 0.0167	0.2405 ± 0.0178	9.6333 ± 1.0334
	BFPSO-PLS	0.9149 ± 0.0077	0.2161 ± 0.0095	11.9333 ± 1.2015
	WS-PSO-PLS	**0.9242 ± 0.0152**	**0.2031 ± 0.0212**	**16.2333 ± 3.5202**
	BDE-PLS	0.8976 ± 0.0113	0.2369 ± 0.0129	18.4667 ± 3.5305
	BFDE-PLS	0.9206 ± 0.0067	0.2087 ± 0.0087	12.0000 ± 1.4384

The first observation from Table 4.4 is that when using all the descriptors in PLS, the mean Q^2 is 0.6003, 0.4007, and 0.2407 and the mean RMSECV is 0.9912, 0.8501, and 0.6461 for the artemisinin, BZR, and selwood datasets, respectively. In contrast, the average number of the selected descriptors in BDE-PLS is drastically decreased. However, under this condition, the mean Q^2 is 0.7481, 0.5735, and 0.8976 and the mean RMSECV is 0.7867, 0.7171, and 0.2369 for the artemisinin, BZR, and selwood datasets, respectively. The aforementioned results suggest that BDE-PLS with less number

of descriptors is significantly better than PLS, which verifies the necessity to perform descriptor selection before the QSAR model development.

From Table 4.4, it can be seen that BFDE-PLS performs better than BDE-PLS in terms of all the performance metrics on the three datasets. For example, with respect to the artemisinin dataset, the mean Q^2 is 0.7594 versus 0.7481, the mean RMSECV is 0.7690 versus 0.7867, and the mean N is 23.6 versus 38.5. As pointed out previously, BFDE-PLS not only use the binary approach, but also incorporate the information of the binary-digits population to the numerical population. Therefore, the feedback strategy can be adopted to enhance the performance of BDE-PLS based on the experimental results. In order to further illustrate the effectiveness of the feedback strategy, contrast test is conducted between BPSO-PLS and BFPSO-PLS. On the basis of the results in Table 4.4, our conclusion is certified again.

As shown in Table 4.4, except the mean Q^2 and the mean RMSECV are slightly worse for the selwood dataset, BFDE-PLS is better than WS-PSO-PLS in term of the mean Q^2, the mean RMSECV and the mean N on the three datasets. As a result, we can conclude that BFDE-PLS benefits from the binary approach and the feedback strategy during the evolution.

Figure 4.19, Figure 4.20 and Figure 4.21 provide the boxplots of Q^2 for the involved methods on the artemisinin, BZR and selwood datasets, respectively. As shown in these figures, BFDE-PLS achieves the best overall performance among the involved methods, which further validates that the feedback strategy during the whole evolution process is effective for the performance improvement. In each box, the horizontal line inside the box represents the median, the edges of the box are the 25th and 75th percentile, the whiskers extending to the most extreme data points are the maximum and minimum, the dot inside the box is the mean, and red "+" represents outliers.

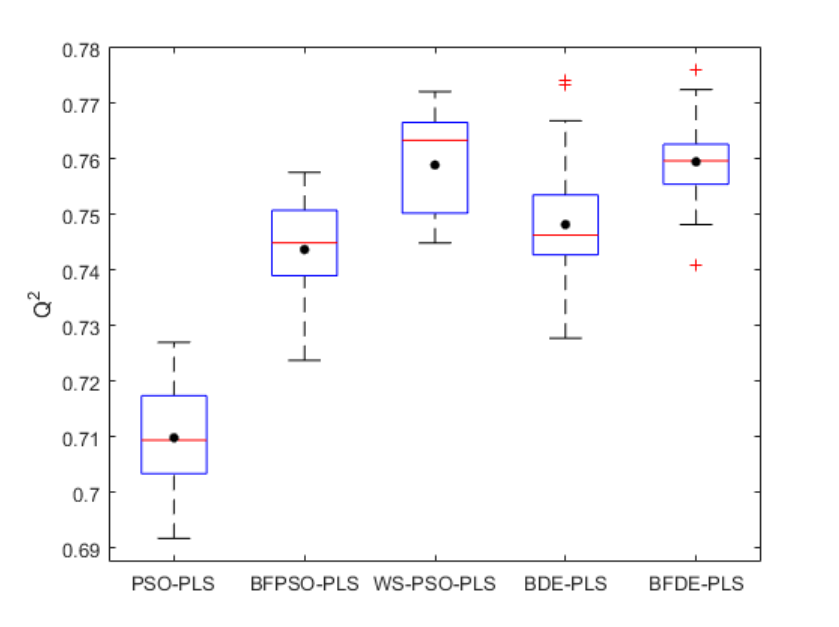


Figure 4.19 Boxplot of Q^2 for different methods on the Artemisinin dataset

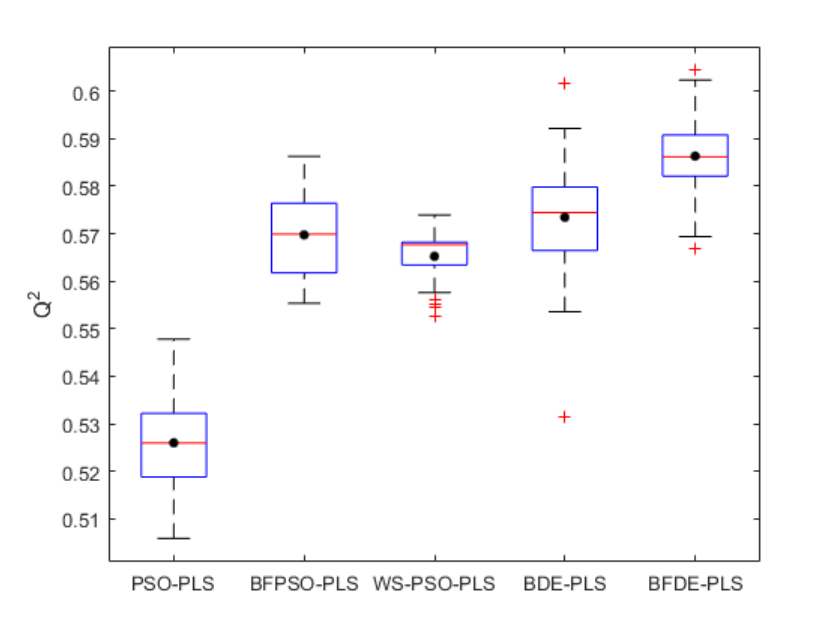


Figure 4.20 Boxplot of Q^2 for different methods on the BZR dataset

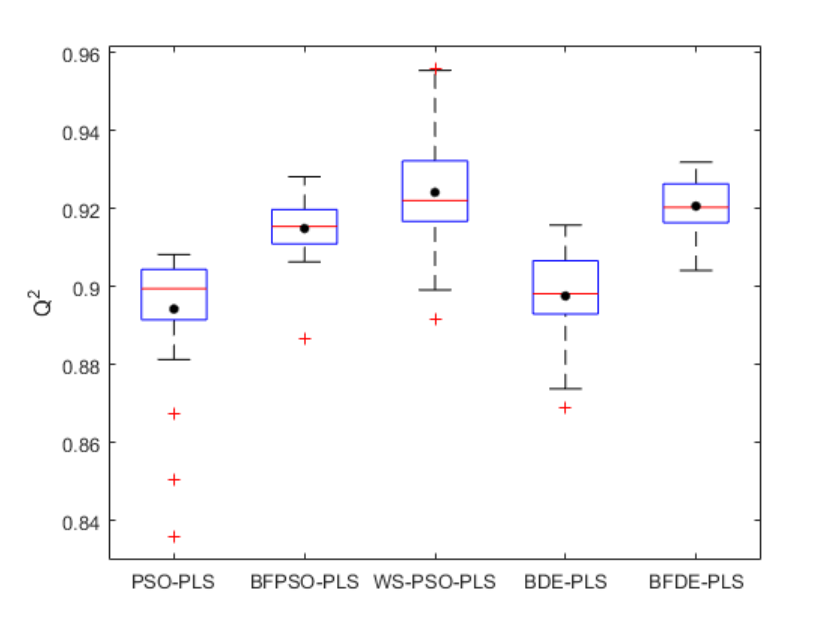


Figure 4.21 Boxplot of Q^2 for different methods on the Selwood dataset

4.5.2.2 MOEA/D-BFDE

The proposed MOEA/D-BFDE includes four main features: (1) a decomposition approach; (2) exploit the neighborhood relationship among the subproblem; (3) a binary approach; (4) incorporate the information of the binary-digits population to the numerical population. By analyzing the performance of the proposed method, three performance metrics have been chosen: Max Q^2, Min

RMSECV and Max N (the max number of the selected descriptors). The parameter settings for all the involved datasets have been listed in Table 4.5 and the experimental results have been presented in Table 4.6, Table 4.7 and Table 4.8.

Table 4.5 The parameter values used in different datasets

Datasets	Methods	Parameters		
		Population size	Number of generations	Number of runs
Artemisinin	MOEA/D-BFDE-MLR	150	400	10
	NSGA-II- MLR	150	400	10
BZR	MOEA/D-BFDE- MLR	150	400	10
	NSGA-II- MLR	150	400	10
Selwood	MOEA/D-BFDE- MLR	150	400	10
	NSGA-II- MLR	150	400	10

The first observation from Table 4.6, Table 4.7 and Table 4.8 is that, in the BZR and Selwood datasets, MOEA/D-BFDE-PLS performs better than NSGA-II- MLR, but the performances are slightly worse than NSGA-II-MLR in the Artemisinin dataset. For example, with respect to the BZR dataset, the max Q^2 is 0.5824 versus 0.5636 and min RMSECV is 0.7097 versus 0.7225. As a result, we can conclude that MOEA/D-BFDE- MLR outperforms NSGA-II-PLS on the whole.

Table 4.6 Experimental results on Artemisinin dataset

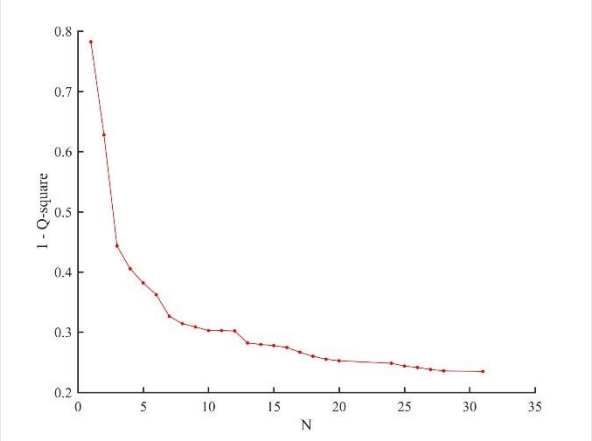
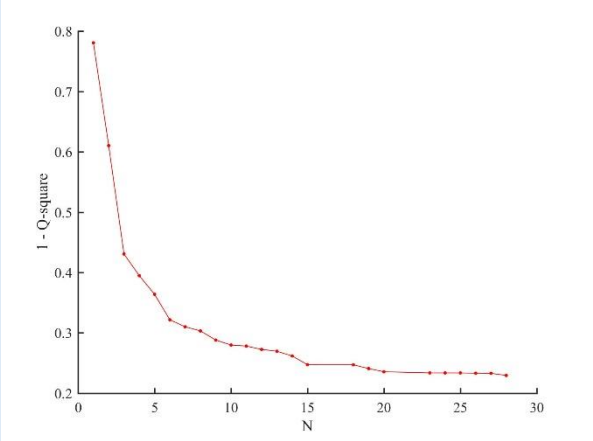
Artemisinin Dataset					
MOEA/D-BFDE-MLR			NSGA-II-MLR		
Max Q^2	Min RMSECV	Max N	Max Q^2	Min RMSECV	Max N
0.7652	0.7597	31	0.7703	0.7514	28
					

Table 4.7 Experimental results on BZR dataset

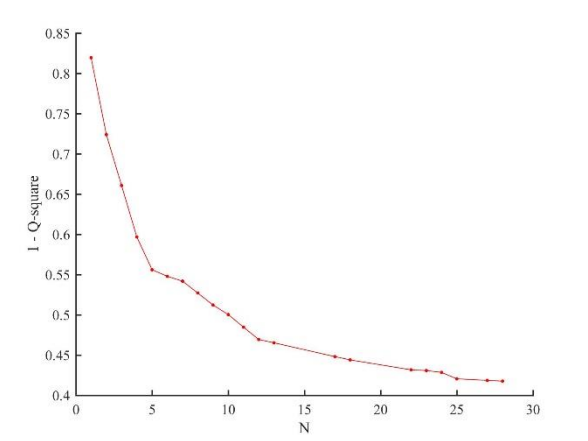
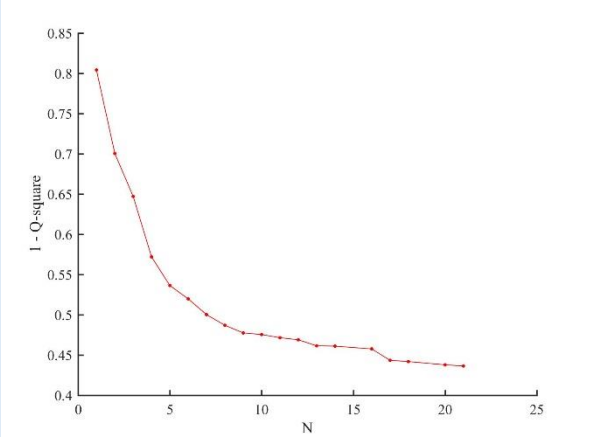
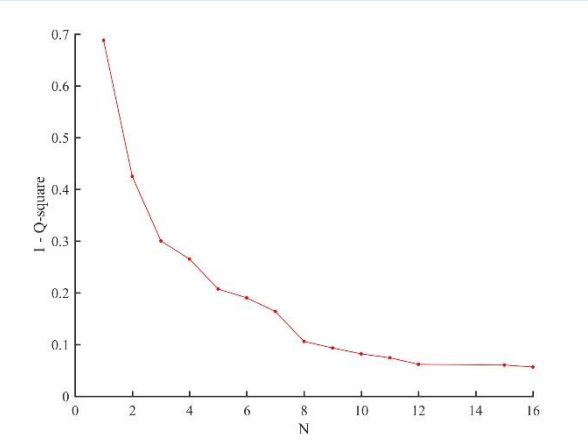
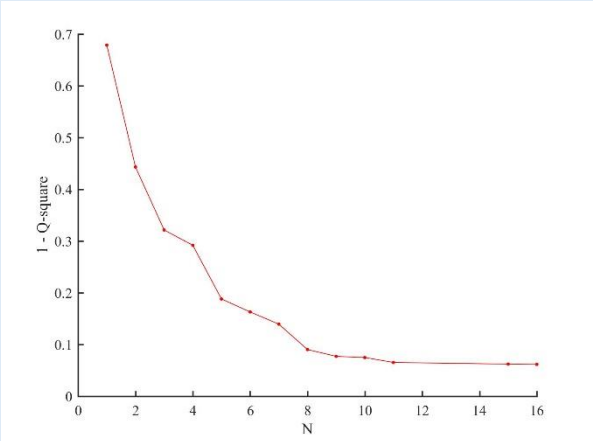
BZR Dataset					
MOEA/D-BFDE-MLR			NSGA-II-MLR		
Max Q^2	Min RMSECV	Max N	Max Q^2	Min RMSECV	Max N
0.5824	0.7097	28	0.5636	0.7255	21
					

Table 4.8 Experimental results on Selwood dataset

Selwood Dataset					
MOEA/D-BFDE-MLR			NSGA-II-MLR		
Max Q^2	Min RMSECV	Max N	Max Q^2	Min RMSECV	Max N
0.9434	0.1764	16	0.9383	0.1841	16
					

5. Applications of ECoFFeS

5.1 ADMET Evaluation in Drug Discovery

Absorption, distribution, metabolism, excretion, and toxicity (ADMET) properties of drug candidates play key roles in drug discovery [91][92]. A schematic diagram of drug discovery, development, and clinical assessment is shown in Figure 4.1. Pharmacodynamics (i.e., activity) is obviously the first object of study, but the new paradigm of drug R&D now dictates that ADMET screening must be initiated rapidly. Activity (PD) and ADMET (PK) screening and evaluation thus run in parallel throughout the preclinical phases.

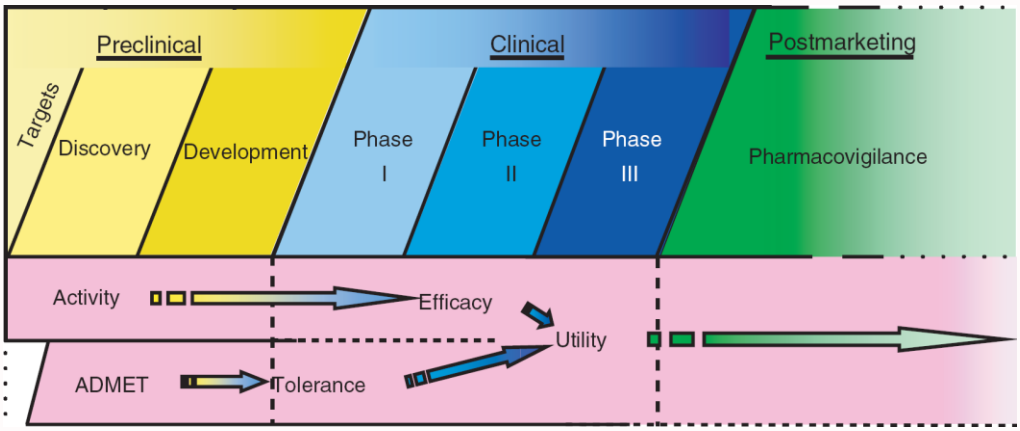


Figure 5.1 A schematic representation of modern drug research and development

In the past decade, only a few drugs out of hundreds of candidates finally reached the market due to the high failure rate at the clinical trial stage. Two main causes to these failures are the lack of efficacy and unacceptable toxicity. Before 10 years ago, about 50% of potential therapeutic compounds failed in clinical trials or were removed from the market due to unacceptable side-effects and poor ADMET properties [93]. In fact, it is now far less (about 8%) compounds that fail due to poor ADMET properties, which is because these get much more attention now [94]. Filtering and optimization of ADMET properties in the early stage of the drug discovery are intensively investigated [95]. However, the experimental evaluation of ADMET profiles is costly, and the work load can't meet the demands of drug screening and lead optimization. In conjunction with high throughput in vitro screening, computational techniques (e.g., quantitative structure-activity relationship methods) that can filter/predict ADMET profiles have become an alternative approach.

Therefore, better prediction capabilities of quantitative structure-activity relationship (QSAR)

models is of particular importance to ADMET evaluation in drug discovery. As a tool for descriptor selection using evolutionary computation approaches, ECoFFeS is able to obtain the important descriptor subsets which assist in building robust consensus (ensemble) model and facilitate the interpretability of relationship between descriptors and activities.

5.2 A Case Study: Predicting hERG Blockers

5.2.1 Background and Introduction

During cardiac depolarization and repolarization, a voltage-gated potassium channel encoded by the human ether-à-go-go related gene (hERG or Kv11.1) plays a major role in the regulation of the exchange of cardiac action potential and resting potential [96][97]. The structure of the hERG channel is a homotetramer, and each subunit contains six transmembrane helices (S1−S6). Some critical residues for the binding of hERG blockers, such as Tyr652, Phe656, and V659 located in S6, have been confirmed by a series of mutagenesis studies [98-100], but unfortunately, the whole crystal structure of the hERG channel has not been solved currently.

The hERG blockade may cause long QT syndrome (LQTS), arrhythmia, and Torsade de Pointes (TdP), which lead to palpitations, fainting, or even sudden death [101-103]. However, to date, several important drugs, such as terfenadine, astemizole, cisapride, vardenafil, and ziprasidone, have been withdrawn from the market or severely restricted in availability due to their undesirable hERG-related cardiotoxicity [100-105]. Therefore, assessment of hERG-related cardiotoxicity has become an important step in the drug design/discovery pipeline [101][106].

As the fact that hERG assays and QT animal studies are time-consuming and expensive, development of reliable and robust in silico models to predict potential hERG liability becomes quite important. In the past decade, a wide range of quantitative structure−activity relationship (QSAR) models for hERG liability have been reported using various machine learning approaches, such as k-nearest neighbor algorithm (KNN), artificial neural networks (ANN), random forest (RF), support vector machine (SVM), self-organizing mapping (SOM), recursive partitioning (RP), genetic algorithm (GA), naive Bayesian classification (NBC), etc [100][107-112]. A majority of the QSAR models are classifiers while only a few of them are quantitative regression models.

5.2.2 Materials and Pretreatment

5.2.2.1 Dataset

The total data set for model construction and validation contains 587 molecules, including 527

molecules with experimental hERG blocking bioactivities (IC50) and 60 hERG nonblockers without IC50 derived from Hou and co-workers [100]. The IC50 activities were mainly determined based on mammalian cell lines, such as HEK, CHO, and COS. IC50 data derived from a nonmammalian cell line, XO (Xenopus laevis oocytes), was included into the data set when no mammalian cell line data was available. If a molecule has multiple different experimental values, it would be compared with the entry found in the PubChem's BioAssay database and the consistent experimental value was adopted [113]. Moreover, a threshold of 40 μM was used to define hERG blockers and nonblockers, where molecules with IC50 < 40 μM were regarded as blockers and others were regarded as nonblockers [114][115].

The whole data set was divided into training set (392) and test set (195) with the ratio of 2:1. The 352 molecules with IC50 values and 40 nonblockers without IC50 in the training set were extracted from the whole data set by using the Find Diverse Molecules protocol in Discovery Studio 2.5 (DS 2.5), respectively [116], which guarantees that the selected molecules in the training set have the largest diversity evaluated by the Tanimoto coefficients based on the LCFP_8 fingerprints and log P [100][117].

5.2.2.2 Molecular descriptors

The Molecular Operating Environment software was used to calculate two-dimensional descriptors of 587 molecules, getting 192 descriptors in total [118]. For 2D descriptors, two pretreatments were performed to delete some uninformative descriptors before further feature selection: 1) delete the descriptors whose variance is 0 or approaches 0; 2) if the correlation coefficient between two descriptors is higher than 0.95, only one was reserved. Finally, 131 descriptors were used by ECoFFeS operation.

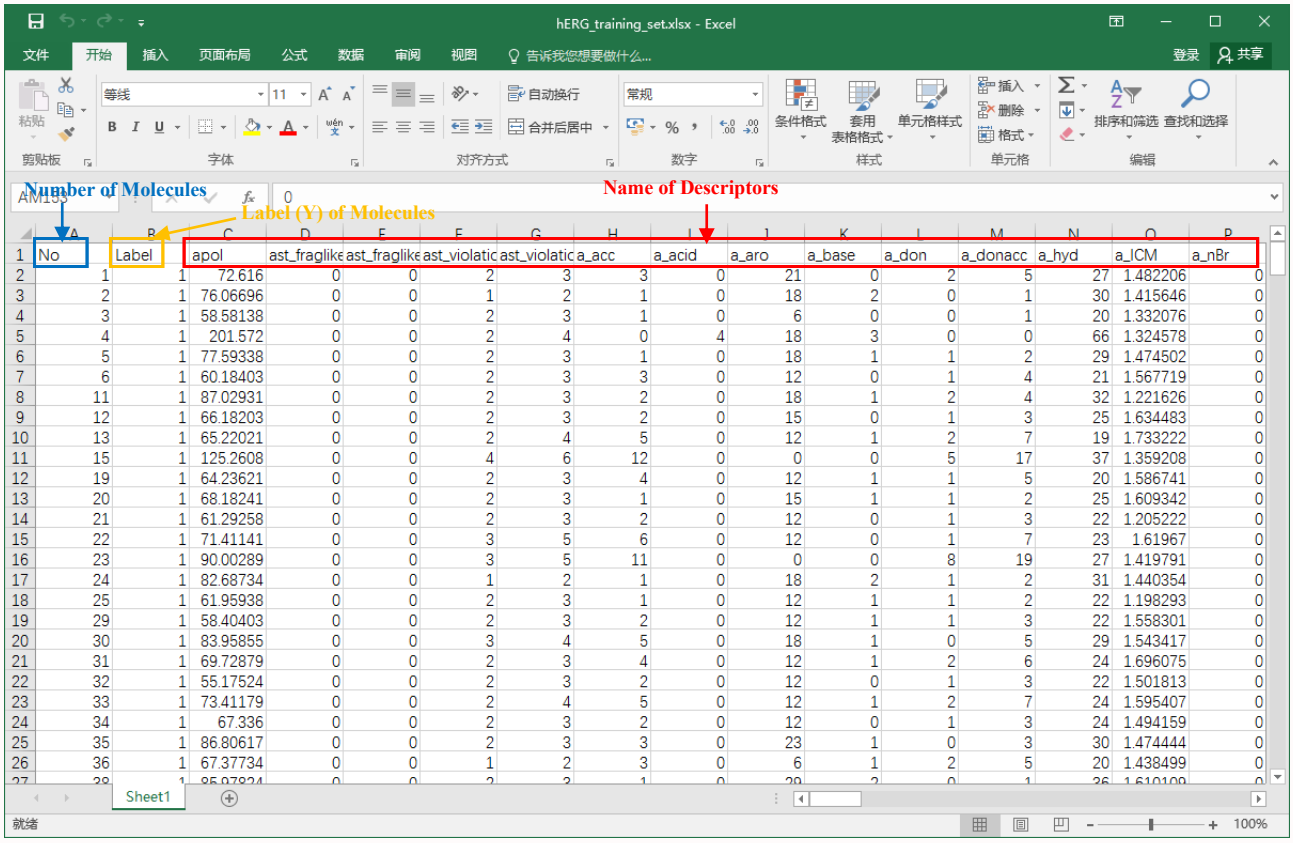
5.2.3 Methods: Classification

5.2.3.1 ECoFFeS Operation

Step 1 Preparation of Data Set

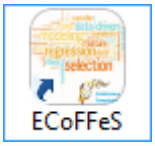
The pretreated training set: hERG_training_set.xlsx (392 molecules with 131 descriptors) https://github.com/Jiaweihuang/ECoFFeS.

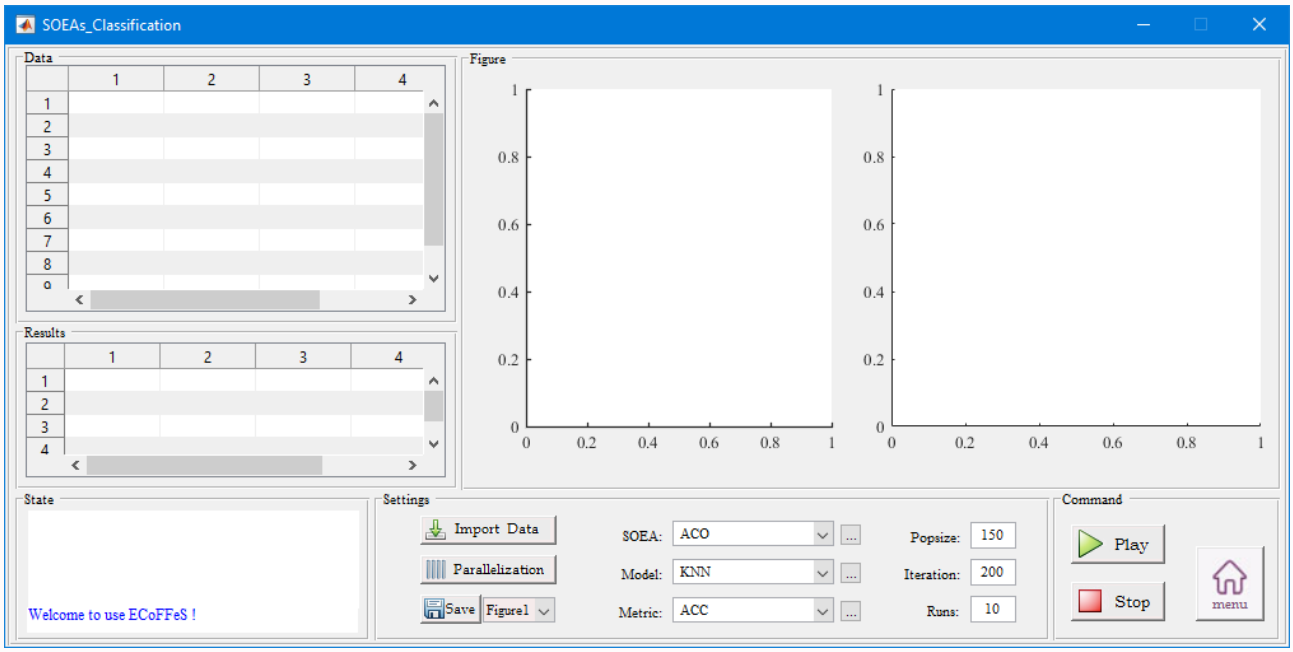
Note: The data format: the first column is Number of Molecules, the second column is Label (Y) of Molecules and the remain columns are Descriptor Values of Molecules



Step 2 Start the SOEAs_Classification

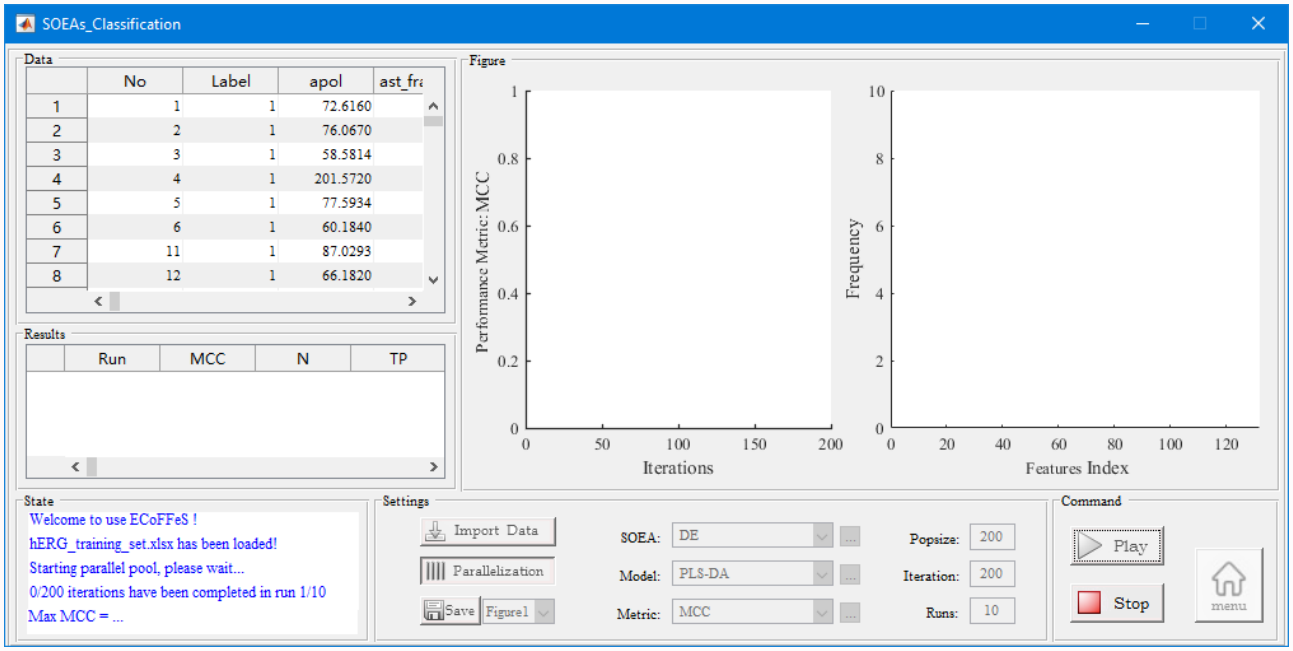
Double-click the ECoFFeS to start the main interface. In main interface, 'Subset Discovery' chooses SOEAs, 'Subset Evaluation' chooses Classification Model & Metric, click Start, then the secondary interface (SOEAs_Classification) appears.





Step 3 Set Parameters and Play

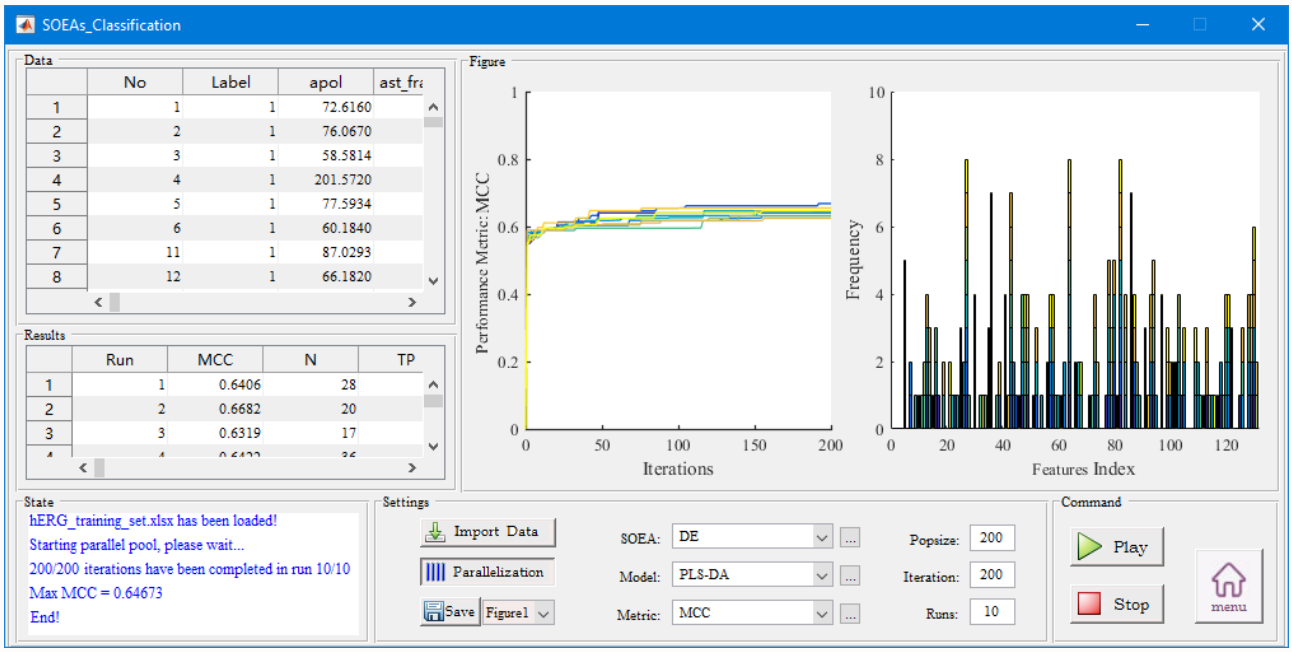
1) Import Data: hERG_training_set.xlsx;
2) Parallelization;
3) SOEA: DE; Model: PLS-DA; Metric: MCC;
4) SOEA_parameter: Default parameters;
5) Model_parameter: Default parameters;
6) Metric_parameter: Default parameters;
7) Popsize: 200; Iteration: 200; Runs: 10;
8) Click Play;



Step 4 Save Figures and Results

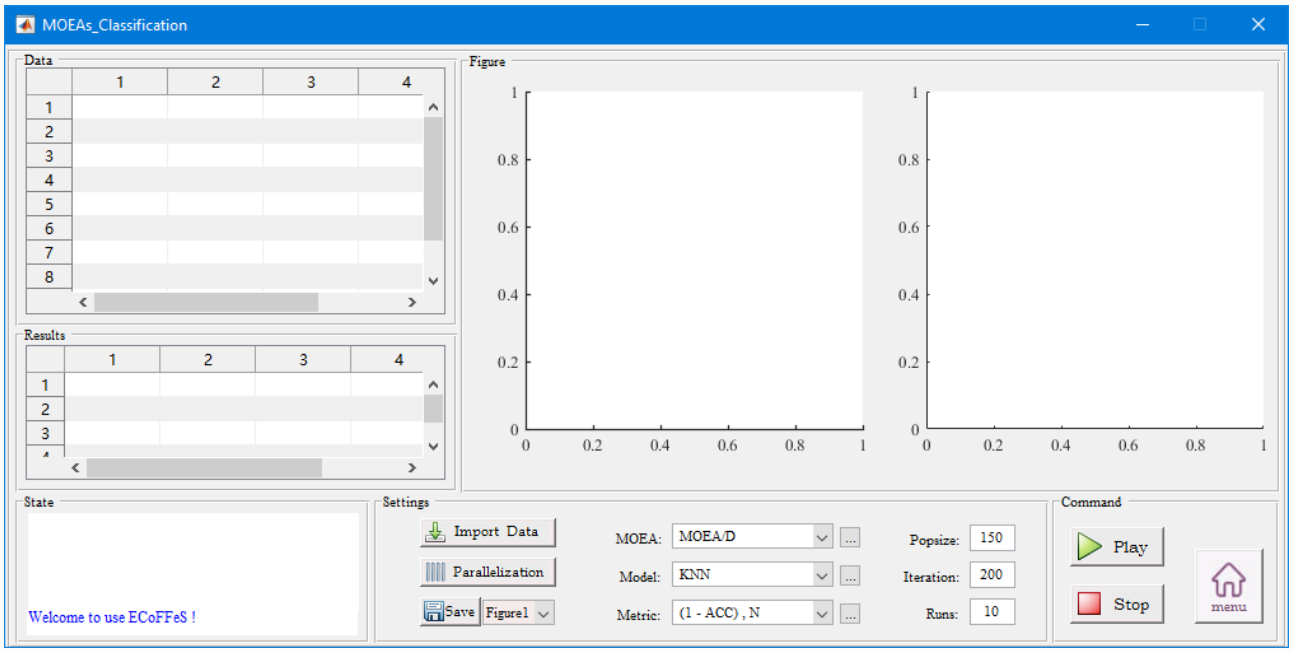
When calculation has been finished, click Save Figure1, Figure2 and Results.

1) Save Figure1: Iteration Figure.jpg
2) Save Figure2: Frequency Figure.jpg
3) Save Results: Results.xlsx



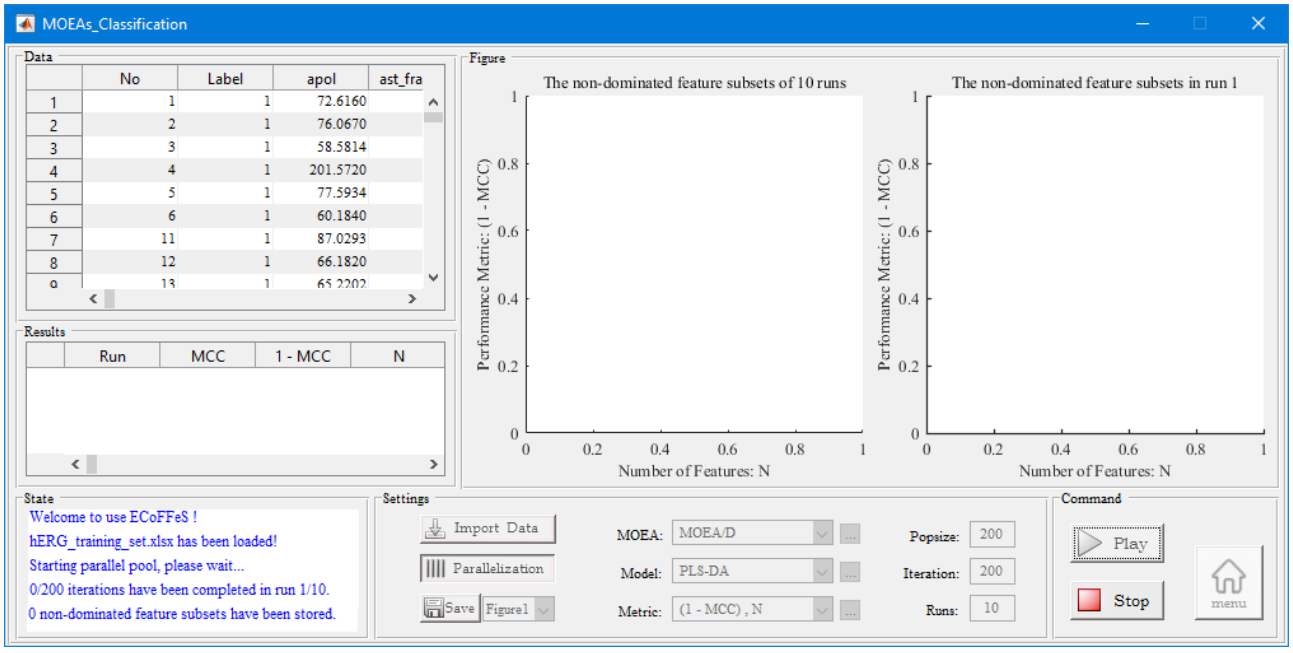
Step 5 Switch to ClassificationMOEAs

Click menu returns to the main interface. In main interface, 'Subset Discovery' chooses SOEAs, 'Subset Evaluation' chooses Classification Model & Metric, click Start, then the secondary interface (MOEAs_Classification) appears.



Step 6 Set Parameters and Play

1) Import Data: hERG_training_set.xlsx;
2) Parallelization;
3) MOEA: MOEA/D; Model: PLS-DA; Metrics: (1 - MCC), N;
4) MOEA_parameter: Default parameters;
5) Model_parameter: Default parameters;
6) Metric_parameter: Default parameters;
7) Popsize: 200; Iteration: 200; Runs: 10;
8) Click Play;



Step 7 Save Figures and Results

When calculation has been finished, click Save Figure1, Figure2 and Results.

1) Save Figure1: Iteration Figure.jpg
2) Save Figure2: Pareto Figure.jpg
3) Save Results: Results.xlsx

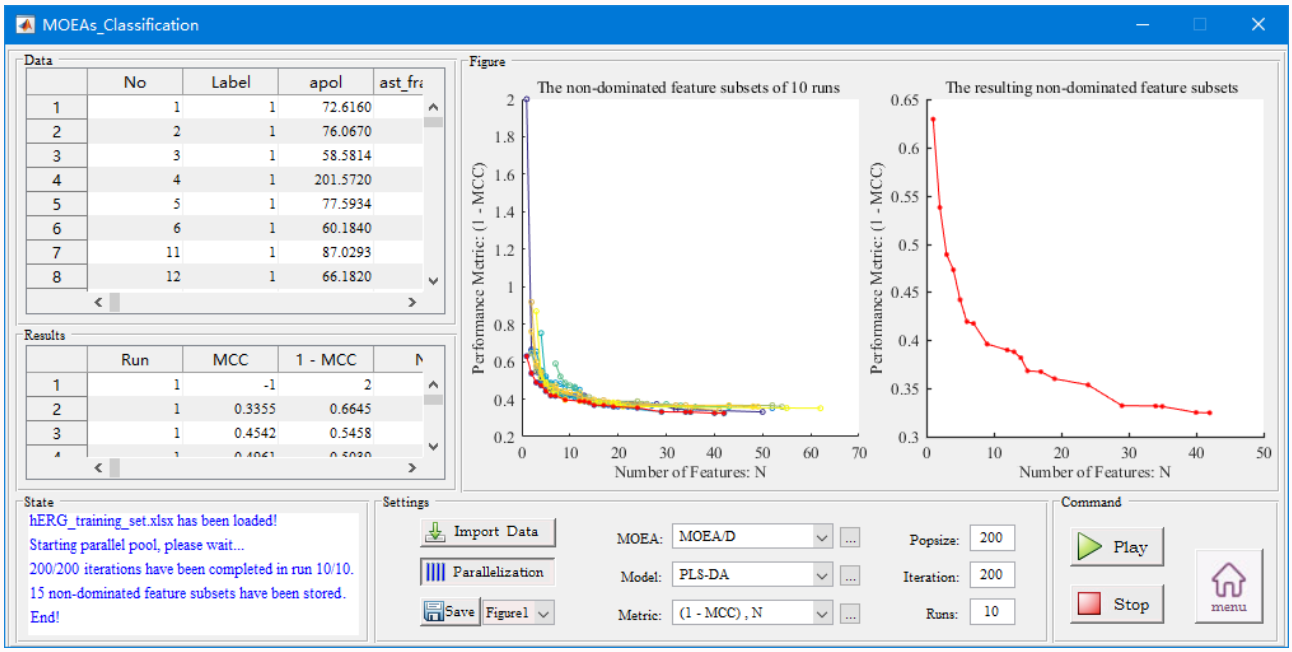


Table 5.1 The Performance of the PLS-DA Classifiers for the Training and Test Sets Based on Different Descriptor Subsets

model	subset size	training (Cross Validation)										test									
		TP	TN	FP	FN	SE	SP	PRE1	PRE2	ACC	MCC	TP	TN	FP	FN	SE	SP	PRE1	PRE2	ACC	MCC
PLS-DA-1	28	268	70	15	39	0.8730	0.8235	0.9470	0.6422	0.8622	0.6406	123	38	6	28	0.8146	0.8636	0.9535	0.5758	0.8256	0.5991
PLS-DA-2	20	270	72	13	37	0.8795	0.8471	0.9541	0.6606	0.8724	0.6682	118	38	11	28	0.8082	0.7755	0.9147	0.5758	0.8000	0.5351
PLS-DA-3	17	271	66	12	43	0.8631	0.8462	0.9576	0.6055	0.8597	0.6319	122	38	7	28	0.8133	0.8444	0.9457	0.5758	0.8205	0.5857
PLS-DA-4	36	266	72	17	37	0.8779	0.8090	0.9399	0.6606	0.8622	0.6422	123	37	6	29	0.8092	0.8605	0.9535	0.5606	0.8205	0.5868
PLS-DA-5	31	270	69	13	40	0.8710	0.8415	0.9541	0.6330	0.8648	0.6467	119	34	10	32	0.7881	0.7727	0.9225	0.5152	0.7846	0.4954
PLS-DA-6	30	270	67	13	42	0.8654	0.8375	0.9541	0.6147	0.8597	0.6323	119	36	10	30	0.7987	0.7826	0.9225	0.5455	0.7949	0.5215
PLS-DA-7	20	270	69	13	40	0.8710	0.8415	0.9541	0.6330	0.8648	0.6467	119	36	10	30	0.7987	0.7826	0.9225	0.5455	0.7949	0.5215
PLS-DA-8	28	271	65	12	44	0.8603	0.8442	0.9576	0.5963	0.8571	0.6247	122	34	7	32	0.7922	0.8293	0.9457	0.5152	0.8000	0.5352
PLS-DA-9	28	268	72	15	37	0.8787	0.8276	0.9470	0.6606	0.8673	0.6551	119	41	10	25	0.8264	0.8039	0.9225	0.6212	0.8205	0.5854
PLS-DA-10	21	270	69	13	40	0.8710	0.8415	0.9541	0.6330	0.8648	0.6467	121	41	8	25	0.8288	0.8367	0.9380	0.6212	0.8308	0.6100
Ensemble (SVM)	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	121	44	8	22	0.8462	0.8462	0.9380	0.6667	0.8462	0.6470

The quality of the PLS-DA classifiers was measured by the quantity of true positives (TP), true negatives (TN), false positives (FP), false negatives (FN), sensitivity (SE), specificity (SP), the prediction accuracy for active (or blockers) (Q+), the prediction accuracy for nonactive (or nonblockers) (Q−), the global accuracy (ACC), and the Matthews correlation coefficient (MCC):

$$\mathrm{SE} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}, \quad \mathrm{SP} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}, \quad \mathrm{PRE1} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}, \quad \mathrm{PRE2} = \frac{\mathrm{TN}}{\mathrm{TN+FN}}, \quad \mathrm{ACC} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}, \quad \mathrm{MCC} = \frac{\mathrm{TP\times TN - FP\times FN}}{\sqrt{(\mathrm{TP+FP})(\mathrm{TP+FN})(\mathrm{TN+FP})(\mathrm{TN+FN})}}$$

The values of ACC and MCC are two important indicators for the classification accuracy of models. The above quantities were calculated for both training and test sets.

5.2.3.2 Results and Discussion

1) From the saved Results.xlsx in Step 4, we organize them into Table 5.1.

2) Figure 5.1 shows the workflow of Ensemble (SVM) in Table 5.1.

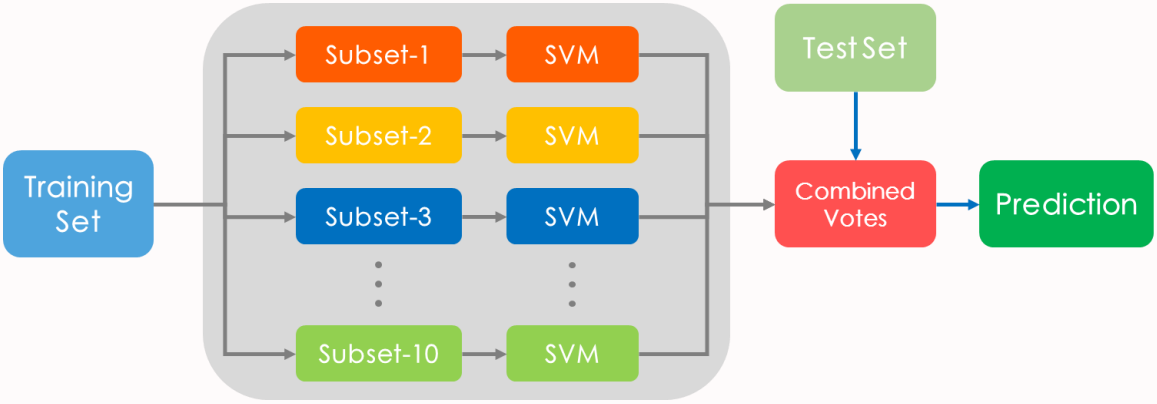


Figure 5.1 The workflow of Ensemble (SVM)

3) On the basis of the saved Iteration Figure.jpg and Frequency Figure.jpg in Step 4, the iteration figure and the frequency figure were showed in Figure 5.2.

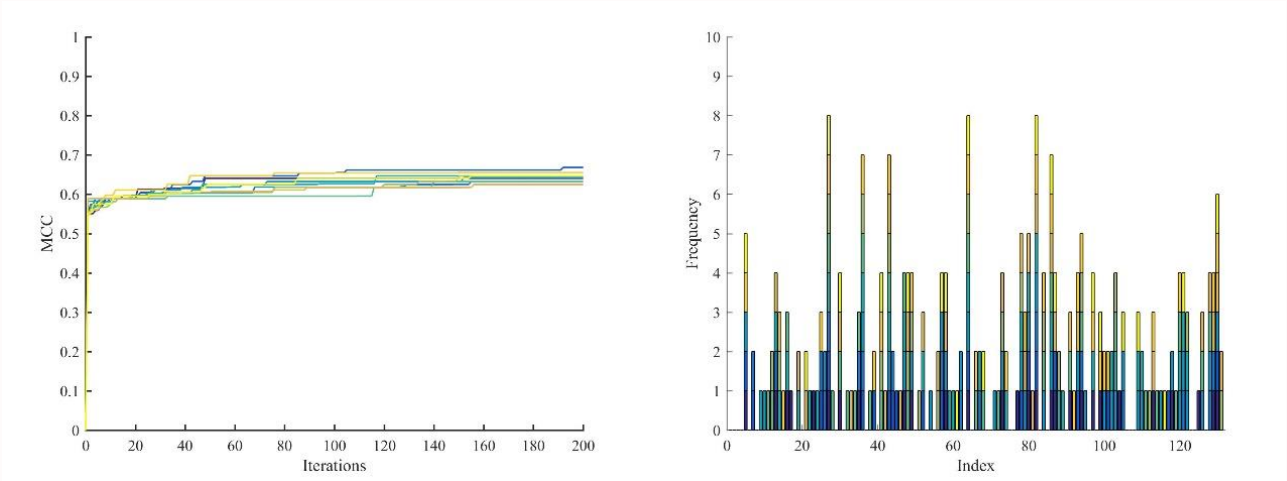


Figure 5.2 The iteration figure and the frequency figure

4) According to the frequency figure, we can acquire the importance of the molecular descriptors and the important descriptors were summarized in Table 5.2. BCUT_SLOGP_0, opr_brigid and PEOE_VSA-2 in the PLS-DA Classifiers had the highest frequency (8/10). Besides, b_max1len, FCharge, PEOE_VSA-6 and vsa_pol were also important, indicated by relatively high frequency (> 5/10).

Table 5.2 Statistical results for the important descriptors

Code	Class	Description
BCUT_SLOGP_0	2D	LogP BCUT (0/3)
opr_brigid	2D	Oprea Rigid Bond Count
PEOE_VSA-2	2D	Total negative 2 vdw surface area
b_max1len	2D	Maximum single-bond chain length
FCharge	2D	Sum of formal charges
vsa_pol	2D	VDW polar surface area (A**2)
PEOE_VSA-6	2D	Total negative 6 vdw surface area

5) From the saved Iteration Figure.jpg in Step 7, a set of non-dominated solutions is obtained, which are tradeoffs between different objectives. In Figure 5.3, the left chart presents the non-dominated descriptor subsets derived from MOEA/D in 10 independent runs, and the right chart presents the resulting non-dominated descriptor subsets from the left chart. In descriptor selection problem, the two main objectives are minimizing the number of descriptors and minimizing the performance metric. If the Pareto front was "smooth" in that adding each additional feature would reduce the performance metric by a small, but significant margin, users could weigh the tradeoff criteria to determine their preferred solutions. Here, we select 29 descriptors in order to keep a good balance between N and (1 - MCC).

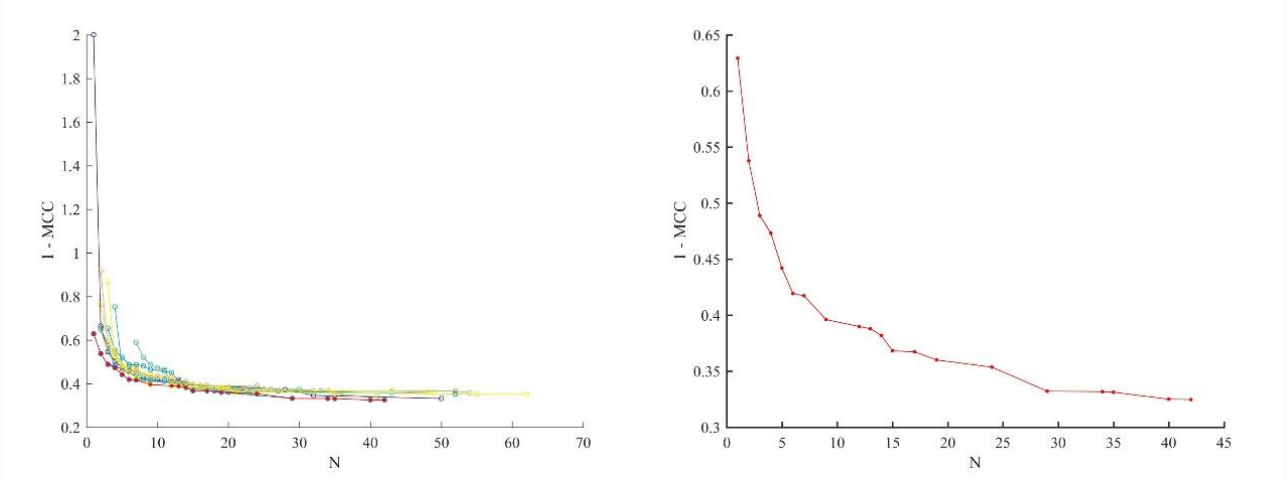


Figure 5.3 The non-dominated descriptor subsets

6) The selected 29 descriptors were presented in Table 5.3.

Table 5.3 Selected descriptors

Code	Class	Description
ast_violation	2D	Astex Fragment-like Violation Count
ast_violation_ext	2D	Astex Fragment-like Violation Count (Extended)
a_donacc	2D	Number of H-bond donor + acceptor atoms
a_hyd	2D	Number of hydrophobic atoms
a_ICM	2D	Atom information content (mean)
a_nCl	2D	Number of chlorine atoms
BCUT_PEOE_0	2D	PEOE Charge BCUT (0/3)
b_max1len	2D	Maximum single-bond chain length
b_triple	2D	Number of triple bonds
FCharge	2D	Sum of formal charges
GCUT_SLOGP_0	2D	LogP GCUT (0/3)
GCUT_SLOGP_1	2D	LogP GCUT (1/3)
lip_don	2D	Lipinski Donor Count
lip_druglike	2D	Lipinski Druglike Test
logS	2D	Log Solubility in Water
opr_nrot	2D	Oprea Rotatable Bond Count
PEOE_VSA+3	2D	Total positive 3 vdw surface area
PEOE_VSA-0	2D	Total negative 0 vdw surface area
PEOE_VSA-2	2D	Total negative 2 vdw surface area
PEOE_VSA-6	2D	Total negative 6 vdw surface area
PEOE_VSA_NEG	2D	Total negative vdw surface area
PEOE_VSA_POL	2D	Total polar vdw surface area
PEOE_VSA_PPOS	2D	Total polar positive vdw surface area
rsynth	2D	Synthetic Feasibility
SlogP_VSA9	2D	Bin 9 SlogP (0.40,10]
SMR_VSA4	2D	Bin 4 SMR (0.390,0.440]
SMR_VSA7	2D	Bin 7 SMR (0.560,10]
vsa_hyd	2D	VDW hydrophobe surface area (A**2)
weinerPol	2D	Weiner polarity number

7) From the saved Results.xlsx in Step 7 and the selected 29 descriptors, we can obtain the Table 5.4. And Figure 5.4 shows the workflow of Ensemble Model in Table 5.4.

Table 5.4 The Performance of the Classifiers for the Training and Test Sets Based on Selected Descriptors

model	Training (Cross Validation)										test									
	TP	TN	FP	FN	SE	SP	PRE1	PRE2	ACC	MCC	TP	TN	FP	FN	SE	SP	PRE1	PRE2	ACC	MCC
Bagged Trees	258	63	25	46	0.8487	0.7159	0.9117	0.5780	0.8189	0.5258	117	46	12	20	0.8540	0.7931	0.9070	0.6970	0.8359	0.6252
Boosted Trees	258	61	25	48	0.8431	0.7093	0.9117	0.5596	0.8138	0.5103	117	45	12	21	0.8478	0.7895	0.9070	0.6818	0.8308	0.6126
KNN	269	47	14	62	0.8127	0.7705	0.9505	0.4312	0.8061	0.4718	121	39	8	27	0.8176	0.8298	0.9380	0.5909	0.8205	0.5851
LDA	249	61	34	48	0.8384	0.6421	0.8799	0.5596	0.7908	0.4595	118	41	11	25	0.8252	0.7885	0.9147	0.6212	0.8154	0.5735
LR	260	68	23	41	0.8638	0.7473	0.9187	0.6239	0.8367	0.5758	120	44	9	22	0.8451	0.8302	0.9302	0.6667	0.8410	0.6349
SVM	268	61	15	48	0.8481	0.8026	0.9470	0.5596	0.8393	0.5742	124	42	5	24	0.8378	0.8936	0.9612	0.6364	0.8513	0.6612
Ensemble	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	118	46	11	20	0.8551	0.8070	0.9147	0.6970	0.8410	0.6364

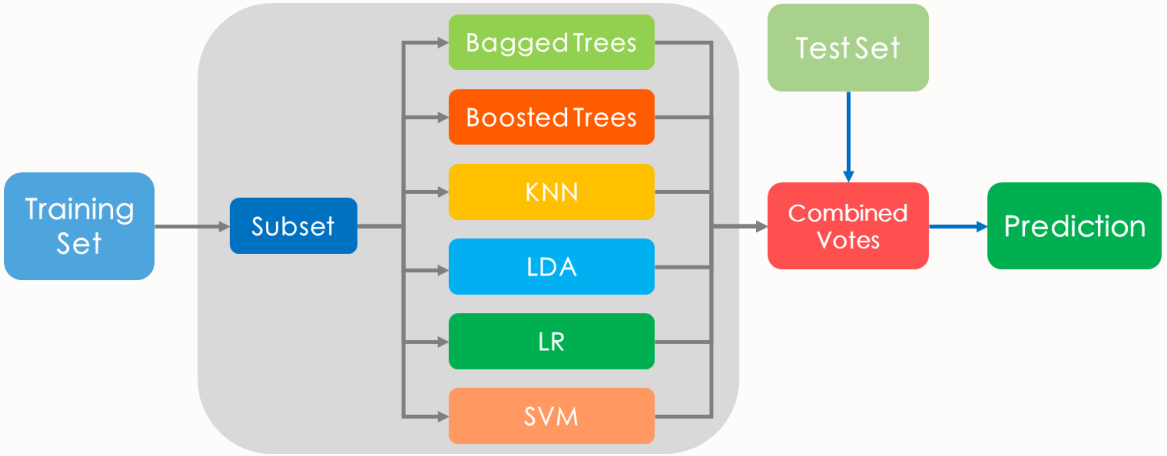


Figure 5.4 Ensemble Model

5.3 A Case Study: Predicting $logD_{7.4}$

5.3.1 Background and Introduction

To exert a therapeutic effect, one drug must enter the blood circulation and then reach the site of action. Thus, an eligible drug usually needs to keep a balance between lipophilicity and hydrophilicity to dissolve in the body fluid and penetrate the biofilm effectively. Therefore, it is very important to evaluate the lipophilicity of candidate compounds in drug research and development process[132].

The lipophilicity of a compound can be quantitatively characterized by the partition coefficient (its logarithm form is denoted as logP) or the distribution coefficient (its logarithm form is denoted as logD) if ionized molecular species are present [119][120]. Partition coefficient [121] is the equilibrium concentration ratio of the solute between two immiscible solvents (e.g., n-octanol and water). Although it is a major descriptor in many quantitative structure–activity relationship (QSAR) equations [121-124] and a crucial part of Lipinski's rule of five [125][126], logP only refers to the neutral form of the compound and is independent of the ionization under physiological conditions. However, it is estimated that 95% of all drugs are ionizable [127][128]. Thus, the distribution coefficient, which takes account of ionization, may be a more reliable measurement for the lipophilicity at physiological pH [129][130]. Distribution coefficient, also known as pH-dependent distribution coefficient, is the ratio of the sum of the equilibrium concentrations of all forms of the compound (i.e., the total sum of ionized and unionized) between two phases. There are several experimental methods to measure the logD value [131] such as the shake-flask method, the slow stirring method, the filter probe method, some chromatography methods, and pH metric techniques. However, these experimental procedures are costly and time-consuming and require substantial quantities of the compound being synthesized. Hence, it is necessary to establish a reliable prediction model to accurately determine logD values without the need for experiments, especially for new or even virtual compounds.

5.3.2 Materials and Pretreatment

5.3.2.1 Dataset

Experimental logD7.4 values were collected from two resources. One is the ChEMBL database including 1451 logD values. The other one is the online chemical database. This database includes 367 logD values. As the logD data in two databases were collected from different literature sources, we only extracted those logD values under the homogeneous experimental conditions, that is,

pH = 7.4, temperature is 25 °C, and the organic solvent is n-octanol. Only those molecules with reliable logD values were considered, and those molecules with empty or indeterminate logD values were removed. If there were two or more entries for one molecule, the arithmetic mean value of these values was adopted. The data set was filtered to remove compounds with logD values greater than 10 or less than −10 because of their potential unreliability. One record was reserved for the conformational or optical isomers, because the subsequent analysis did not consider the stereochemistry. Solvent or saline ions adhering to the molecule were removed automatically by OpenBabel. The SMILES structures of these compounds were checked one by one to ensure that they were correct. After a series of pretreatments, 1130 molecules and their logD7.4 values were finally collected [132]. Herein, all molecules were divided into two parts, namely the training set and the test set, using the Kennard–Stone method [133][134] to guarantee that the test samples could map the measured region of the input variables space completely. Thus, we obtained a training set of 904 molecules (80% of the data set) and a test set of 226 molecules (20% of the data set).

5.3.2.2 Molecular descriptors

The Molecular Operating Environment software was used to calculate two-dimensional descriptors of 1130 molecules, getting 192 descriptors in total [118]. For 2D descriptors, two pretreatments were performed to delete some uninformative descriptors before further feature selection: 1) delete the descriptors whose variance is 0 or approaches 0; 2) if the correlation coefficient between two descriptors is higher than 0.95, only one was reserved. Finally, 127 descriptors were operated by ECoFFeS operation.

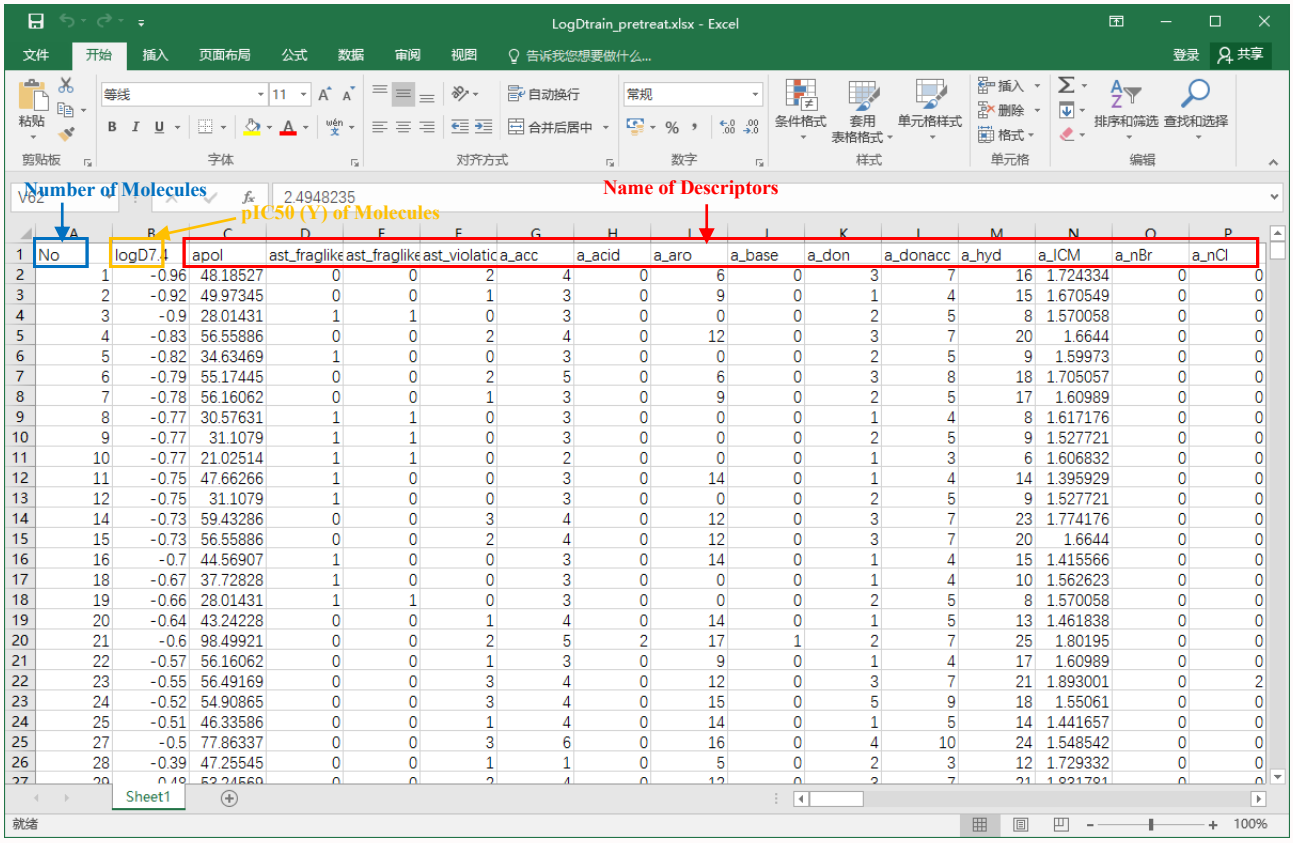
5.3.3 Methods: Regression

5.3.3.1 ECoFFeS Operation

Step 1 Preparation of Data Set

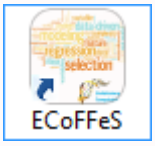
The pretreated training set: LogDtrain_pretreat.xlsx (904 molecules with 127 descriptors) https://github.com/Jiaweihuang/ECoFFeS .

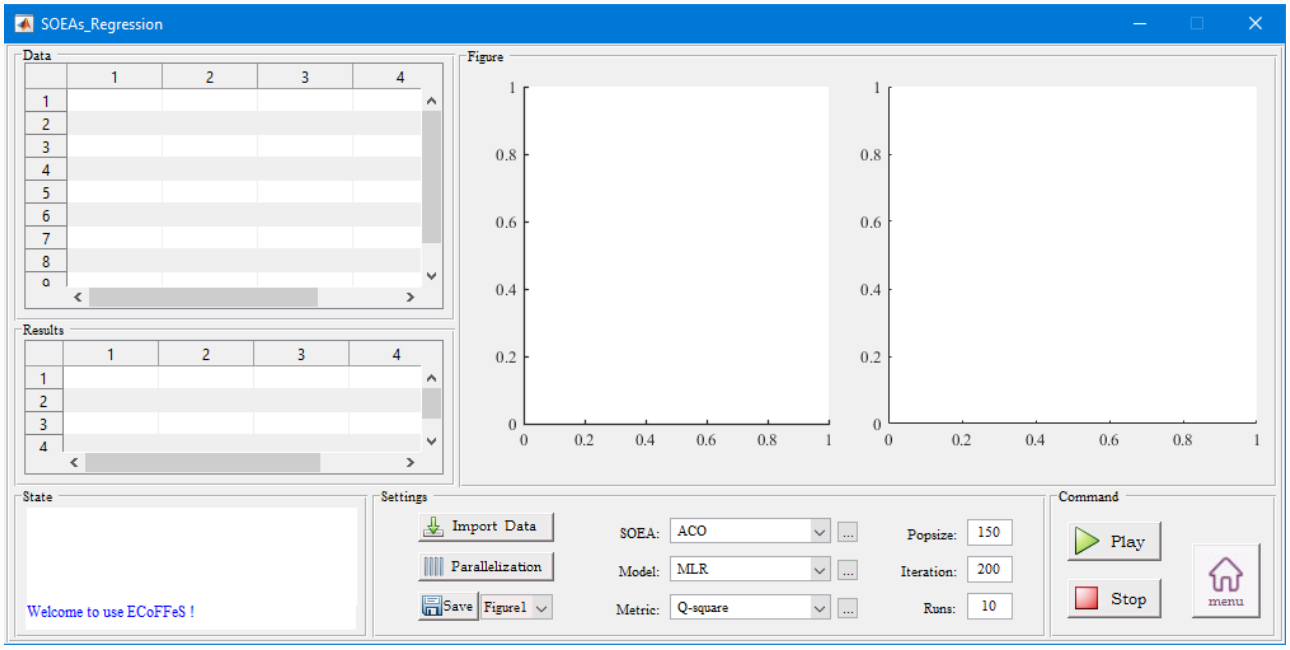
Note: The data format: the first column is Number of Molecules, the second column is $LogD_{7.4}$ (Y) of Molecules and the remain columns are Descriptor Values of Molecules



Step 2 Start the SOEAs_Regression

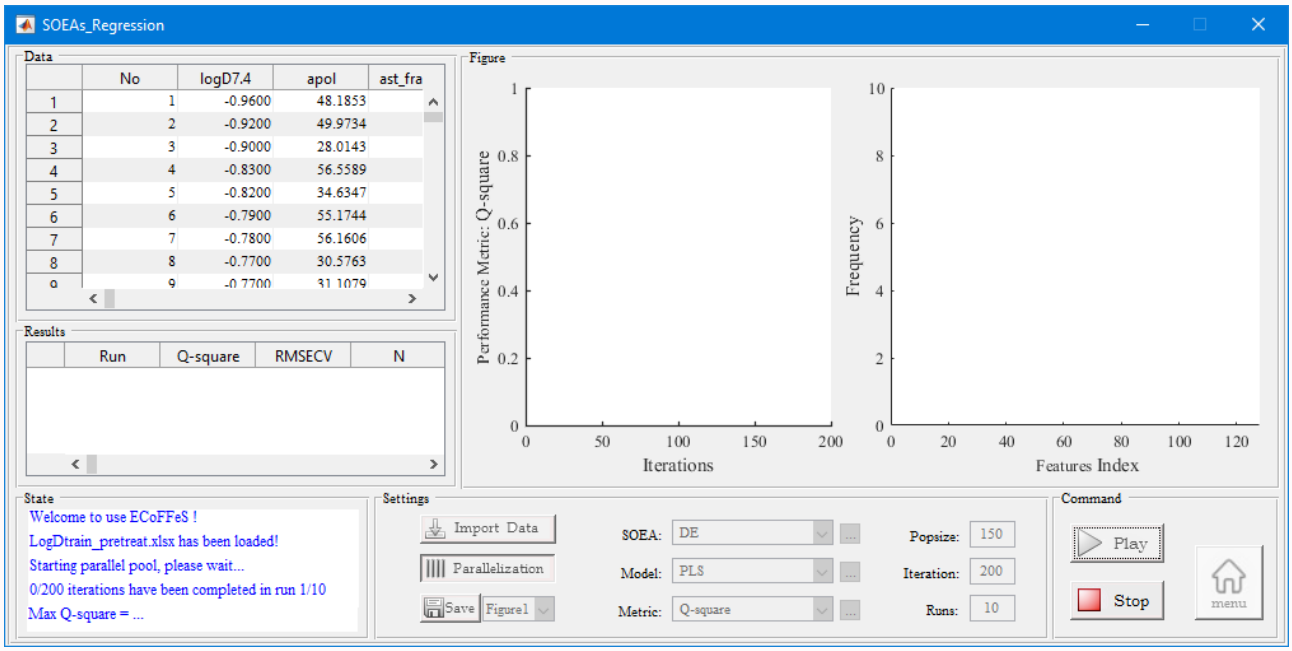
Double-click the ECoFFeS to start the main interface. In main interface, 'Subset Discovery' chooses SOEAs, 'Subset Evaluation' chooses Regression Model & Metric, click Start, then the secondary interface (SOEAs_Regression) appears.





Step 3 Set Parameters and Play

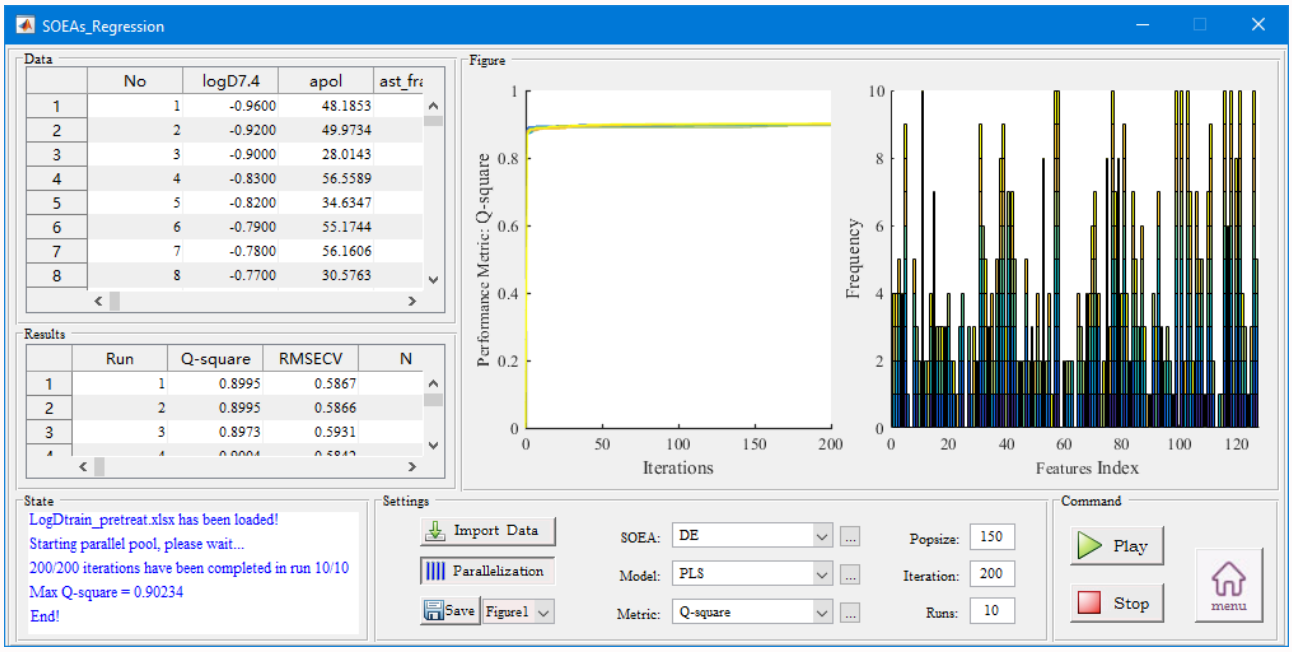
1) Import Data: LogDtrain_pretreat.xlsx;
2) Parallelization;
3) SOEA: DE; Model: PLS; Metric: Q-square;
4) SOEA_parameter: Default parameters
5) Model_parameter: 40
6) Metric_parameter: Default parameters
7) Popsize: 150; Iteration: 200; Runs: 10;
8) Click Play;



Step 4 Save Figures and Results

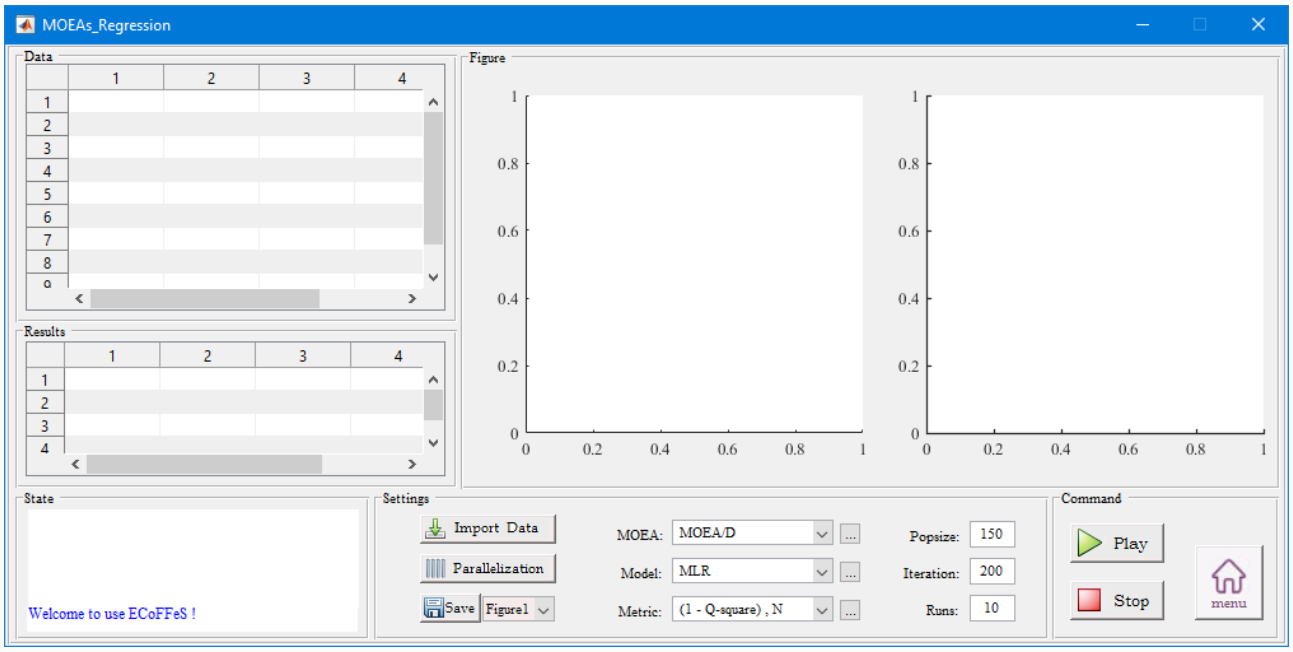
When calculation has been finished, click Save Figure1, Figure2 and Results.

1) Save Figure1: Iteration Figure.jpg
2) Save Figure2: Frequency Figure.jpg
3) Save Results: Results.xlsx



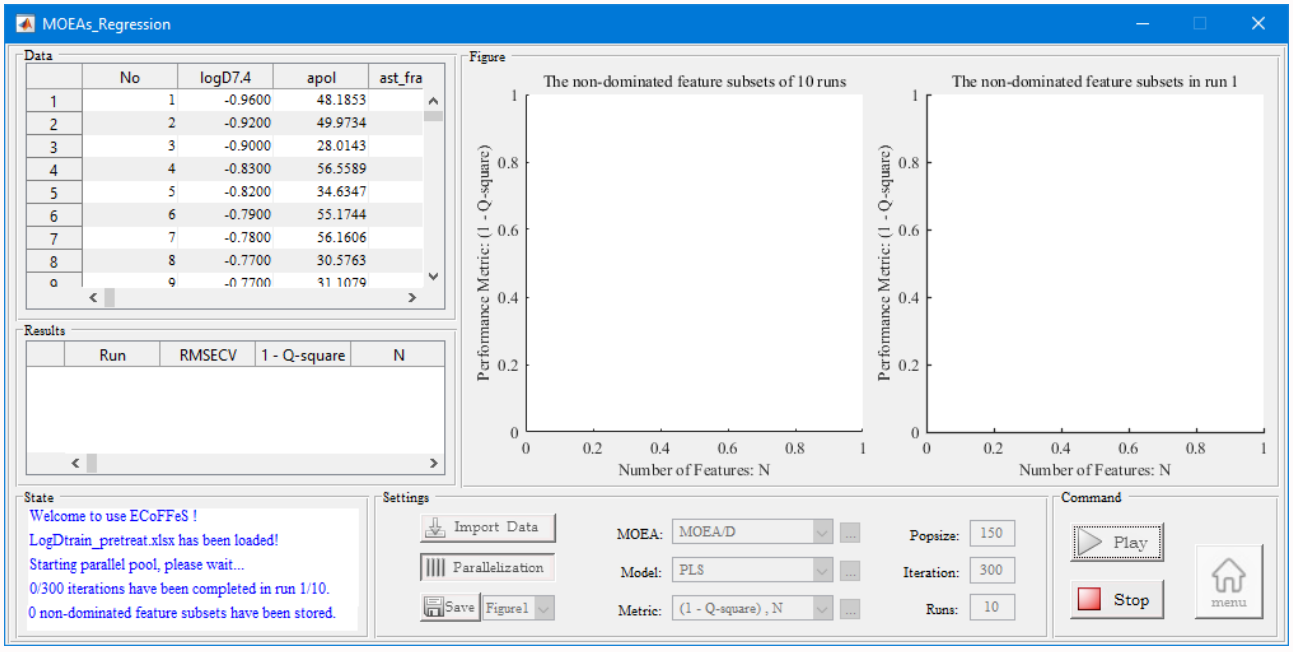
Step 5 Switch to MOEAs_Regression

Click menu returns to the main interface. In main interface, 'Subset Discovery' chooses MOEAs, 'Subset Evaluation' chooses Regression Model & Metric, click Start, then the secondary interface (MOEAs_Regression) appears.



Step 6 Set Parameters and Play

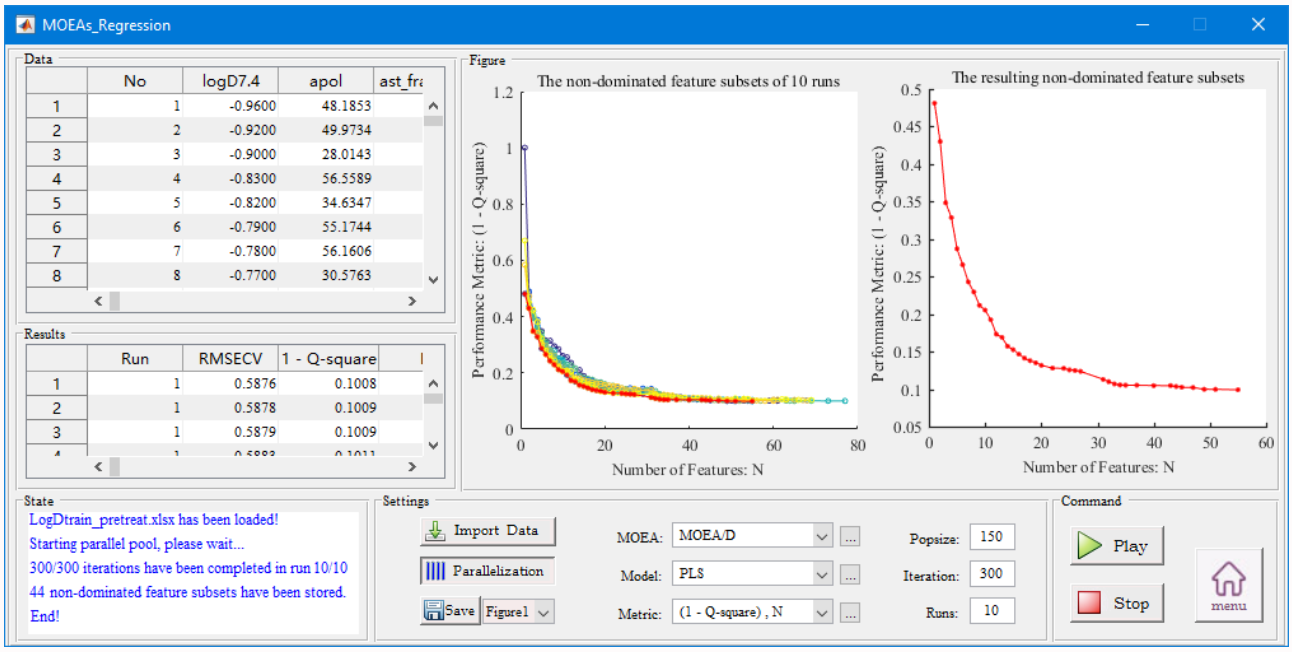
1) Import Data: LogDtrain_pretreat.xlsx;
2) Parallelization;
3) MOEA: MOEA/D; Model: PLS; Metric: (1 – Q-square), N;
4) MOEAs_parameter: Default parameters;
5) Models_parameter: 40;
6) Metrics_parameter: Default parameters;
7) Popsize: 150; Iteration: 300; Runs: 10;
8) Click Play



Step 7 Save Figures and Results

When calculation has been finished, click Save Figure1, Figure2 and Results.

1) Save Figure1: Iteration Figure.jpg
2) Save Figure2: Pareto Figure.jpg
3) Save Results: Results.xlsx



5.2.4.2 Results and Discussion

1) From the saved Results.xlsx in Step 4, we organize them into Table 5.5. Moreover, other two parameters, mean absolute error (MAE) and root mean square error (RMSE), were used to evaluate the quality of each model.

2) Figure 5.5 shows the workflow of Consensus Modelling (PLS) in Table 5.5.

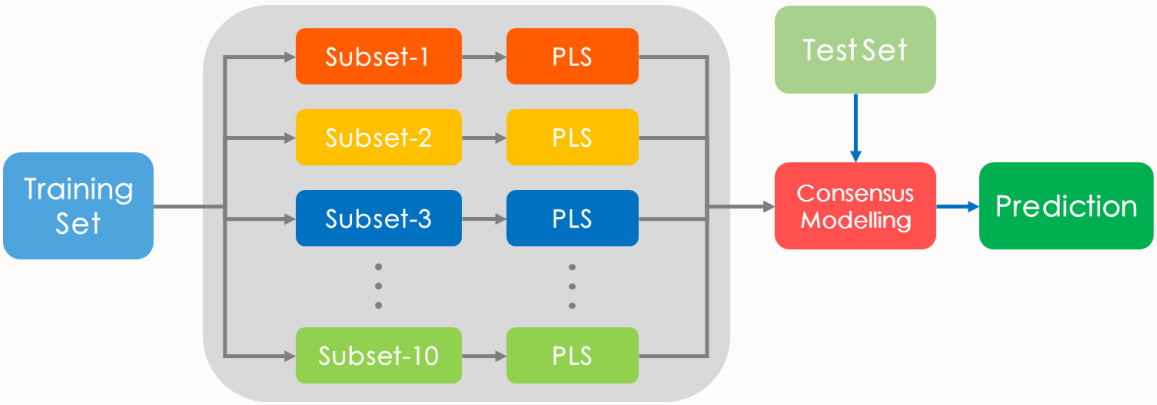


Figure 5.5 the workflow of Consensus Modelling (PLS)

Table 5.5 Statistical Results for the QSAR Models Based on Different Descriptor Subsets

model	subset size	training				test		
		R^2_{adj}	Q^2	$\mathrm{MAE_{train}}$	$\mathrm{RMSE_{train}}$	Q^2_{ext}	$\mathrm{MAE_{test}}$	$\mathrm{RMSE_{test}}$
PLS-1	47	0.9042	0.8986	0.4441	0.5578	0.8517	0.4999	0.6491
PLS-2	58	0.9030	0.8969	0.4440	0.5575	0.8182	0.5287	0.7186
PLS-3	65	0.9020	0.8944	0.4476	0.5581	0.8556	0.5023	0.6405
PLS-4	41	0.9043	0.8980	0.4473	0.5593	0.8756	0.4726	0.5944
PLS-5	47	0.9035	0.8973	0.4499	0.5597	0.8762	0.4827	0.5929
PLS-6	69	0.9034	0.8974	0.4429	0.5530	0.8613	0.4851	0.6276
PLS-7	72	0.9006	0.8929	0.4526	0.5598	0.8736	0.4799	0.5991
PLS-8	42	0.9054	0.8989	0.4443	0.5557	0.8736	0.4718	0.5992
PLS-9	45	0.9048	0.8993	0.4447	0.5566	0.8554	0.4940	0.6409
PLS-10	47	0.9059	0.9007	0.4433	0.5529	0.8779	0.4684	0.5891
Consensus Modelling (PLS)	NA	NA	NA	0.4348	0.5428	0.8731	0.4762	0.6005

3) On the basis of the saved Iteration Figure.jpg and Frequency Figure.jpg in Step 4, the iteration figure and the frequency figure were showed in Figure 5.6.

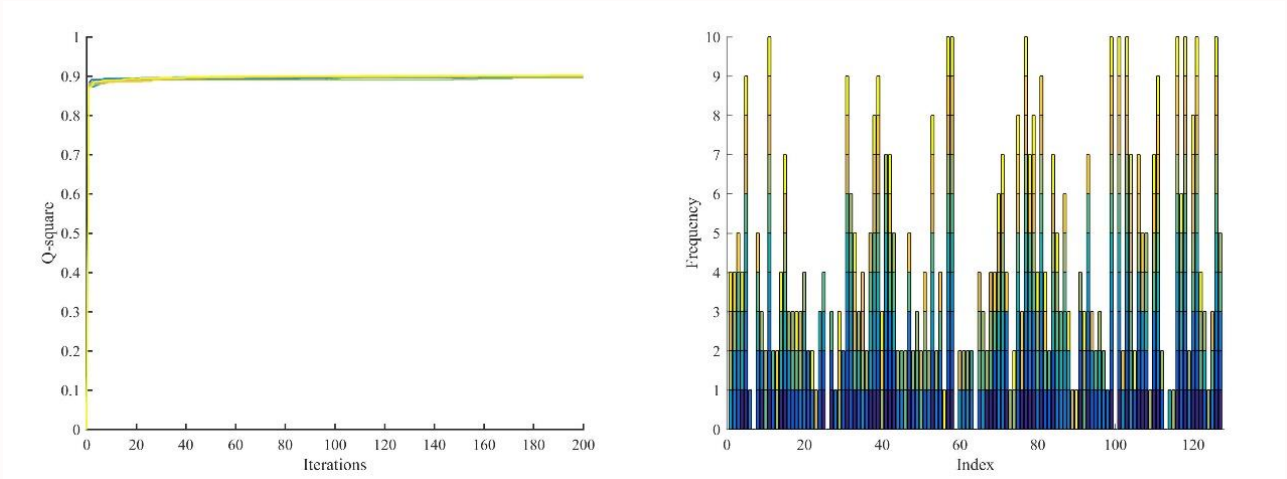


Figure 5.6 The iteration figure and the frequency figure

4) According to the frequency figure, we can acquire the importance of the molecular descriptors and the important descriptors were summarized in Table 5.6. a_hyd, logP(o/w), logS, PEOE_VSA-1, SlogP, SlogP_VSA1, SlogP_VSA3, SMR_VSA6, TPSA, vsa_acc and vsa_pol in the PLS models had the highest frequency (10/10).

Table 5.6 Statistical results for the important descriptors

Code	Class	Description
a_hyd	2D	Number of hydrophobic atoms
logP(o/w)	2D	Log octanol/water partition coefficient
logS	2D	Log Solubility in Water
PEOE_VSA-1	2D	Total negative 1 vdw surface area
SlogP	2D	Log Octanol/Water Partition Coefficient
SlogP_VSA1	2D	Bin 1 SlogP (-0.40,-0.20]
SlogP_VSA3	2D	Bin 3 SlogP (0.00,0.10]
SMR_VSA6	2D	Bin 6 SMR (0.485,0.560]
TPSA	2D	Topological Polar Surface Area (A**2)
vsa_acc	2D	VDW acceptor surface area (A**2)
vsa_pol	2D	VDW polar surface area (A**2)

5) From the saved Pareto Figure.jpg in Step 7, a set of Pareto front (nondominated) solutions is obtained, which are tradeoffs between different objectives. In Figure 5.7, the left chart presents the non-dominated descriptor subsets derived from MOEA/D in 10 independent runs, and the right chart presents the resulting non-dominated descriptor subsets from the left chart. In descriptor selection problem, the two main objectives are minimizing the number of descriptors and minimizing the performance metric. If the Pareto front was “smooth” in that adding each additional feature would reduce the performance metric by a small, but significant margin, users could weigh the tradeoff criteria to determine their preferred solutions. Here, we select 20 descriptors in order to keep a good balance between N and (1 – Q-square).

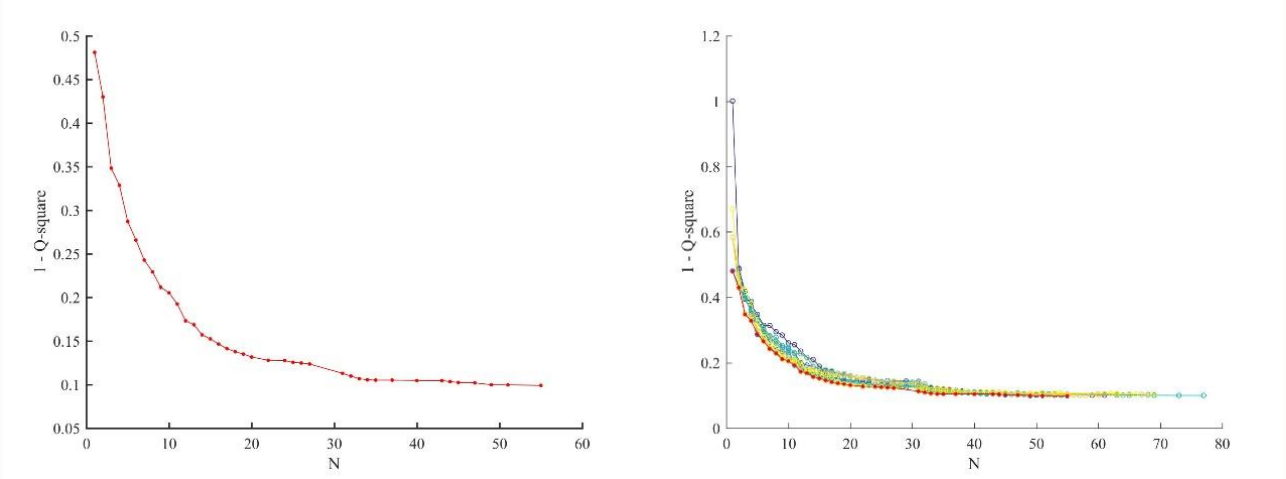


Figure 5.7 The non-dominated descriptor subsets

6) The selected 20 descriptors were presented in Table 5.7.

Table 5.7 The selected descriptors

Code	Class	Description
apol	2D	Sum of atomic polarizabilities
a_donacc	2D	Number of H-bond donor + acceptor atoms
a_hyd	2D	Number of hydrophobic atoms
balabanJ	2D	Balaban averaged distance sum connectivity
logP(o/w)	2D	Log octanol/water partition coefficient
logS	2D	Log Solubility in Water
nmol	2D	Number of molecules
PEOE_VSA+0	2D	Total positive 0 vdw surface area
PEOE_VSA+3	2D	Total positive 3 vdw surface area
PEOE_VSA-2	2D	Total negative 2 vdw surface area
PEOE_VSA-5	2D	Total negative 5 vdw surface area
PEOE_VSA_FPOS	2D	Fractional positive vdw surface area
PEOE_VSA_POS	2D	Total positive vdw surface area
SlogP_VSA1	2D	Bin 1 SlogP (-0.40,-0.20]
SlogP_VSA2	2D	Bin 2 SlogP (-0.20,0.00]
SlogP_VSA3	2D	Bin 3 SlogP (0.00,0.10]
SMR_VSA1	2D	Bin 1 SMR (0.110,0.260]
SMR_VSA6	2D	Bin 6 SMR (0.485,0.560]
TPSA	2D	Topological Polar Surface Area (A**2)
VDistEq	2D	Vertex distance equality index

7) From the saved Results.xlsx in Step 7 and the selected 20 descriptors, we can obtain the Table 5.8.

Table 5.8 The Performance of the Models for the Training and Test Sets Based on Selected Descriptors

model	training				test		
	R^2_{adj}	Q^2	MAE_{train}	$RMSE_{train}$	Q^2_{ext}	MAE_{test}	$RMSE_{test}$
PLS	0.8719	0.8681	0.5243	0.6550	0.8177	0.5849	0.7196
GP	0.9934	0.9075	0.1118	0.1486	0.8985	0.4014	0.5371
SVR	0.8706	0.8646	0.5207	0.6584	0.8131	0.5891	0.7287
Boosted Trees	0.8919	0.8367	0.4660	0.6017	0.8154	0.5662	0.7241
Bagged Trees	0.9374	0.8266	0.3304	0.4581	0.8423	0.5179	0.6693
Consensus Modelling	NA	NA	0.3568	0.4483	0.8705	0.4851	0.6064

8) Figure 5.8 shows the workflow of consensus modeling in Table 5.8.

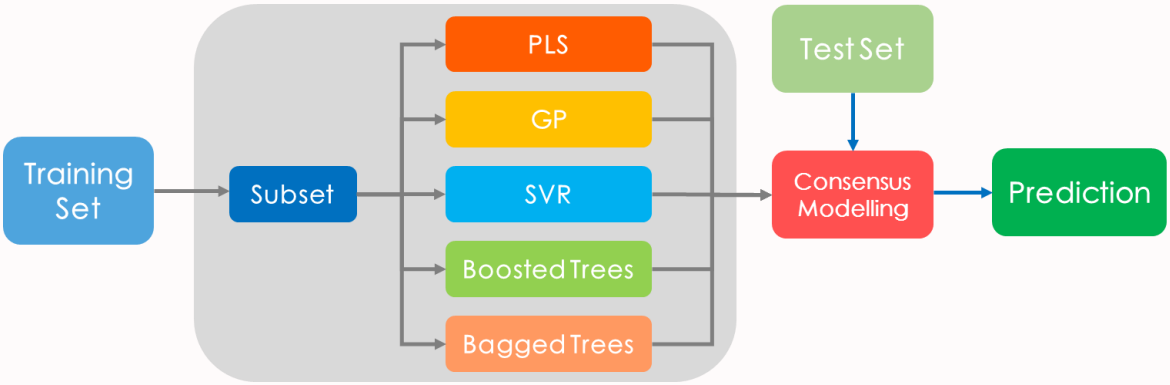


Figure 5.8 Consensus Modelling

6. About ECoFFeS

ECoFFeS is a free and open-source software for feature selection using evolutionary computation. It is being developed in CITAA and CBDD groups at Central South University.

6.1 Citation

If ECoFFeS is utilized in scientific work that results in a publication, it is expected that the following reference is cited:

Jiawei Huang, Yong Wang and Dongsheng Cao
ECoFFeS: a software for feature selection using evolutionary computation
In submission

Jiawei Huang
Contact: jiaweihuangchina@gmail.com

Yong Wang
Contact: ywang@csu.edu.cn

Dongsheng Cao
Contact: oriental-cds@163.com

6.2 License

Bibliography

[1] Saeys Y, Inza I, Larrañaga P. A review of feature selection techniques in bioinformatics[J]. bioinformatics, 2007, 23(19): 2507-2517.

[2] Wang L, Wang Y, Chang Q. Feature selection methods for big data bioinformatics: A survey from the search perspective[J]. Methods, 2016, 111: 21-31.

[3] Khan A U. Descriptors and their selection methods in QSAR analysis: paradigm for drug design[J]. Drug Discovery Today, 2016, 21(8): 1291-1302.

[4] Eiben A E, Smith J. From evolutionary computation to the evolution of things[J]. Nature, 2015, 521(7553): 476-482.

[5] Cao D, Liang Y, Xu Q, et al. Toward better QSAR/QSPR modeling: simultaneous outlier detection and variable selection using distribution of model features[J]. Journal of computer-aided molecular design, 2011, 25(1): 67-80.

[6] Dash M, Liu H. Feature selection for classification[J]. Intelligent data analysis, 1997, 1(3): 131-156.

[7] Guyon I, Elisseeff A. An introduction to variable and feature selection[J]. Journal of machine learning research, 2003, 3(Mar): 1157-1182.

[8] John G H, Kohavi R, Pfleger K. Irrelevant features and the subset selection problem[C]//Machine learning: proceedings of the eleventh international conference. 1994: 121-129.

[9] Roy K, Kar S, Das R N. Understanding the basics of QSAR for applications in pharmaceutical sciences and risk assessment[M]. Academic press, 2015.

[10] Roy K, Kar S, Das R N. A Primer on QSAR/QSPR Modeling: Fundamental Concepts[M]. Springer, 2015.

[11] Liu Y, Tang F, Zeng Z. Feature selection based on dependency margin[J]. IEEE transactions on cybernetics, 2015, 45(6): 1209-1221.

[12] Liu H, Zhao Z. Manipulating data and dimension reduction methods: Feature selection[M]//Encyclopedia of Complexity and Systems Science. Springer New York, 2009: 5348-5359.

[13] Liu H, Motoda H, Setiono R, et al. Feature Selection: An Ever Evolving Frontier in Data Mining[J]. FSDM, 2010, 10: 4-13.

[14] Liu H, Yu L. Toward integrating feature selection algorithms for classification and clustering[J]. IEEE Transactions on knowledge and data engineering, 2005, 17(4): 491-502.

[15] Unler A, Murat A. A discrete particle swarm optimization method for feature selection in binary classification problems[J]. European Journal of Operational Research, 2010, 206(3): 528-539.

[16] Liu Y, Wang G, Chen H, et al. An improved particle swarm optimization for feature selection[J]. Journal of Bionic Engineering, 2011, 8(2): 191-200.

[17] Bäck T, Fogel D B, Michalewicz Z. Handbook of evolutionary computation[J]. Release, 1997, 97(1): B1.

[18] Fogel D B. Evolutionary computation: toward a new philosophy of machine intelligence[M]. John Wiley & Sons, 2006.

[19] Eiben A E, Smith J. From evolutionary computation to the evolution of things[J]. Nature, 2015, 521(7553): 476-482.

[20] Xue B, Zhang M, Browne W, et al. A survey on evolutionary computation approaches to feature selection[J]. 2015.
[21] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies[C]//Proceedings of the first European conference on artificial life. 1991, 142: 134-142.
[22] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1996, 26(1): 29-41.
[23] Dorigo M, Birattari M, Stutzle T. Ant colony optimization[J]. IEEE computational intelligence magazine, 2006, 1(4): 28-39.
[24] Chen B, Chen L, Chen Y. Efficient ant colony optimization for image feature selection[J]. Signal processing, 2013, 93(6): 1566-1576.
[25] Yu H, Gu G, Liu H, et al. A modified ant colony optimization algorithm for tumor marker gene selection[J]. Genomics, proteomics & bioinformatics, 2009, 7(4): 200-208.
[26] Storn R, Price K. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces[M]. Berkeley: ICSI, 1995.
[27] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of global optimization, 1997, 11(4): 341-359.
[28] WANG Y, LIU Z Z, LI J, et al. On the selection of solutions for mutation in differential evolution[J]. 2016.
[29] Wang Y, Cai Z, Zhang Q. Enhancing the search ability of differential evolution through orthogonal crossover[J]. Information Sciences, 2012, 185(1): 153-177.
[30] Pampara G, Engelbrecht A P, Franken N. Binary differential evolution[C]//2006 IEEE International Conference on Evolutionary Computation. IEEE, 2006: 1873-1879.
[31] Holland J H. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. U Michigan Press, 1975.
[32] Holland J H. Genetic algorithms[J]. Scientific american, 1992, 267(1): 66-72.
[33] Roberge V, Tarbouchi M, Labonté G. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning[J]. IEEE Transactions on Industrial Informatics, 2013, 9(1): 132-141.
[34] Qiu M, Ming Z, Li J, et al. Phase-change memory optimization for green cloud with genetic algorithm[J]. IEEE Transactions on Computers, 2015, 64(12): 3528-3540.
[35] Noraini M R, Geraghty J. Genetic algorithm performance with different selection strategies in solving TSP[J]. 2011.
[36] Yen Y S, Chao H C, Chang R S, et al. Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs[J]. Mathematical and Computer Modelling, 2011, 53(11): 2238-2250.
[37] Eberhart R C, Kennedy J. A new optimizer using particle swarm theory[C]//Proceedings of the sixth international symposium on micro machine and human science. 1995, 1: 39-43.
[38] Shi Y, Eberhart R. A modified particle swarm optimizer[C]//Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, 1998: 69-73.
[39] Kennedy J. Particle swarm optimization[M]//Encyclopedia of machine learning. Springer US, 2011: 760-766.
[40] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm[C]//Systems,

Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. IEEE, 1997, 5: 4104-4108.

[41] Liu H, Cai Z, Wang Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization[J]. Applied Soft Computing, 2010, 10(2): 629-640.

[42] Lee S, Soak S, Oh S, et al. Modified binary particle swarm optimization[J]. Progress in Natural Science, 2008, 18(9): 1161-1166.

[43] Xue B, Zhang M, Browne W N. Particle swarm optimization for feature selection in classification: a multi-objective approach[J]. IEEE transactions on cybernetics, 2013, 43(6): 1656-1671.

[44] Stadler W. A survey of multicriteria optimization or the vector maximum problem, part I: 1776–1960[J]. Journal of Optimization Theory and Applications, 1979, 29(1): 1-52.

[45] Deb K. Multi-objective optimization using evolutionary algorithms[M]. John Wiley & Sons, 2001.

[46] Zhou A, Qu B Y, Li H, et al. Multiobjective evolutionary algorithms: A survey of the state of the art[J]. Swarm and Evolutionary Computation, 2011, 1(1): 32-49.

[47] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition[J]. IEEE Transactions on evolutionary computation, 2007, 11(6): 712-731.

[48] Li H, Zhang Q. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II[J]. IEEE Transactions on Evolutionary Computation, 2009, 13(2): 284-302.

[49] Miettinen K. Nonlinear multiobjective optimization[M]. Springer Science & Business Media, 2012.

[50] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE transactions on evolutionary computation, 2002, 6(2): 182-197.

[51] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 577-601.

[52] Jain H, Deb K. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: handling constraints and extending to an adaptive approach[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(4): 602-622.

[53] Durillo J J, Zhang Q, Nebro A J, et al. Distribution of computational effort in parallel MOEA/D[C]//International Conference on Learning and Intelligent Optimization. Springer Berlin Heidelberg, 2011: 488-502.

[54] Gong Y J, Chen W N, Zhan Z H, et al. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art[J]. Applied Soft Computing, 2015, 34: 286-300.

[55] Molinaro A M, Simon R, Pfeiffer R M. Prediction error estimation: a comparison of resampling methods[J]. Bioinformatics, 2005, 21(15): 3301-3307.

[56] Wold S. Cross-validatory estimation of the number of components in factor and principal components models[J]. Technometrics, 1978, 20(4): 397-405.

[57] Wold S. Validation of QSAR's[J]. Quantitative Structure-Activity Relationships, 1991, 10(3): 191-193.

[58] Debnath A K, Ghose A K, Viswanadhan V N. Combinatorial library design and evaluation[J]. Principles, Software, Tools and Application in Drug Discovery, 2001: 73-129.

[59] Walker J D, Carlsen L, Jaworska J. Improving opportunities for regulatory acceptance of QSARs:

the importance of model domain, uncertainty, validity and predictability[J]. Qsar & Combinatorial Science, 2003, 22(3): 346-350.
[60] Afantitis A, Melagraki G, Sarimveis H, et al. A combined LS-SVM & MLR QSAR workflow for predicting the inhibition of CXCR3 receptor by quinazolinone analogs[J]. Molecular diversity, 2010, 14(2): 225-235.
[61] Fawcett T. An introduction to ROC analysis[J]. Pattern recognition letters, 2006, 27(8): 861-874.
[62] Matthews B W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme[J]. Biochimica et Biophysica Acta (BBA)-Protein Structure, 1975, 405(2): 442-451.
[63] Truchon J F, Bayly C I. Evaluating virtual screening methods: good and bad metrics for the "early recognition" problem[J]. Journal of chemical information and modeling, 2007, 47(2): 488-508.
[64] Snedecor G W. Cochran. WG, Statistical Methods[J]. 1967.
[65] Wold S, Sjöström M, Eriksson L. PLS-regression: a basic tool of chemometrics[J]. Chemometrics and intelligent laboratory systems, 2001, 58(2): 109-130.
[66] Vapnik V N, Vapnik V. Statistical learning theory[M]. New York: Wiley, 1998.
[67] Cristianini N, Shawe-Taylor J. An introduction to support vector machines and other kernel-based learning methods[M]. Cambridge university press, 2000.
[68] Schölkopf B, Smola A J. Learning with kernels: support vector machines, regularization, optimization, and beyond[M]. MIT press, 2002.
[69] Cover T, Hart P. Nearest neighbor pattern classification[J]. IEEE transactions on information theory, 1967, 13(1): 21-27.
[70] Alpaydin E. Introduction to machine learning[M]. MIT press, 2014.
[71] Finley A O, McRoberts R E. Efficient k-nearest neighbor searches for multi-source forest attribute mapping[J]. Remote Sensing of Environment, 2008, 112(5): 2203-2211.
[72] Harrell F E. Regression Modeling Strategies Springer[J]. New York, 2001.
[73] Höskuldsson A. PLS regression methods[J]. Journal of chemometrics, 1988, 2(3): 211-228.
[74] Barker M, Rayens W. Partial least squares for discrimination[J]. Journal of chemometrics, 2003, 17(3): 166-173.
[75] Pérez N F, Ferré J, Boqué R. Calculation of the reliability of classification in discriminant partial least-squares binary classification[J]. Chemometrics and Intelligent Laboratory Systems, 2009, 95(2): 122-128.
[76] Ballabio D, Consonni V. Classification tools in chemistry. Part 1: linear models. PLS-DA[J]. Analytical Methods, 2013, 5(16): 3790-3798.
[77] Hsu C W, Chang C C, Lin C J. A practical guide to support vector classification[J]. 2003.
[78] Hsu C W, Lin C J. A comparison of methods for multiclass support vector machines[J]. IEEE transactions on Neural Networks, 2002, 13(2): 415-425.
[79] Weston J, Watkins C. Support vector machines for multi-class pattern recognition[C]//ESANN. 1999, 99: 219-224.
[80] Kreßel U H G. Pairwise classification and support vector machines[C]//Advances in kernel methods. MIT Press, 1999: 255-268.
[81] Hearst M A, Dumais S T, Osman E, et al. Support vector machines[J]. IEEE Intelligent Systems and their Applications, 1998, 13(4): 18-28.
[82] Valyon J, Horváth G. A Weighted Generalized LS-SVM[J]. Periodica Polytechnica, Electrical Engineering, 2003, 47(3): 229-251.

[83] Avery M A, Alvim-Gaston M, Rodrigues C R, et al. Structure-activity relationships of the antimalarial agent artemisinin. 6. The development of predictive in vitro potency models using CoMFA and HQSAR methodologies[J]. Journal of medicinal chemistry, 2002, 45(2): 292-303.
[84] Guha R, Jurs P C. Development of QSAR models to predict and interpret the biological activity of artemisinin analogues[J]. Journal of chemical information and computer sciences, 2004, 44(4): 1440-1449.
[85] Cao D S, Xu Q S, Hu Q N, et al. ChemoPy: freely available python package for computational biology and chemoinformatics[J]. Bioinformatics, 2013: btt105.
[86] Neaz M M, Muddassar M, Pasha F A, et al. 2D-QSAR of non-benzodiazepines to benzodiazepines receptor (BZR)[J]. Medicinal chemistry research, 2009, 18(2): 98-111.
[87] Haefely W, Kyburz E, Gerecke M, et al. Recent advances in the molecular pharmacology of benzodiazepine receptors and in the structure-activity relationships of their agonists and antagonists[J]. Advances in drug research, 1985, 14: 165-322.
[88] Selwood D L, Livingstone D J, Comley J C W, et al. Structure-activity relationships of antifilarial antimycin analogs: a multivariate pattern recognition study[J]. Journal of medicinal chemistry, 1990, 33(1): 136-142.
[89] Nicolotti O, Gillet V J, Fleming P J, et al. Multiobjective optimization in quantitative structure-activity relationships: Deriving accurate and interpretable QSARs[J]. Journal of Medicinal Chemistry, 2002, 45(23): 5069-5080.
[90] Wang Y, Huang J J, Zhou N, et al. Incorporating PLS model information into particle swarm optimization for descriptor selection in QSAR/QSPR[J]. Journal of Chemometrics, 2015, 29(12): 627-636.
[91] Comprehensive medicinal chemistry II[M]. Elsevier Science Ltd, 2007.
[92] Cheng F, Li W, Zhou Y, et al. admetSAR: a comprehensive source and free tool for assessment of chemical ADMET properties[J]. Journal of chemical information and modeling, 2012, 52(11): 3099-3105.
[93] Kola I, Landis J. Can the pharmaceutical industry reduce attrition rates?[J]. Nature reviews Drug discovery, 2004, 3(8): 711-716.
[94] Merlot C. Computational toxicology—a tool for early safety evaluation[J]. Drug discovery today, 2010, 15(1): 16-22.
[95] Hou T, Wang J. Structure–ADME relationship: still a long way to go?[J]. Expert opinion on drug metabolism & toxicology, 2008, 4(6): 759-770.
[96] Wang S, Li Y, Wang J, et al. ADMET evaluation in drug discovery. 12. Development of binary classification models for prediction of hERG potassium channel blockage[J]. Molecular pharmaceutics, 2012, 9(4): 996-1010.
[97] Wang S, Sun H, Liu H, et al. ADMET evaluation in drug discovery. 16. Predicting hERG blockers by combining multiple pharmacophores and machine learning approaches[J]. Molecular Pharmaceutics, 2016, 13(8): 2855-2866.
[98] Vandenberg J I, Perry M D, Perrin M J, et al. hERG K+ channels: structure, function, and clinical significance[J]. Physiological reviews, 2012, 92(3): 1393-1478.
[99] Recanatini M, Poluzzi E, Masetti M, et al. QT prolongation through hERG K+ channel blockade: current knowledge and strategies for the early prediction during drug development[J]. Medicinal research reviews, 2005, 25(2): 133-166.
[100] Villoutreix B O, Taboureau O. Computational investigations of hERG channel blockers: New

insights and current predictive models[J]. Advanced drug delivery reviews, 2015, 86: 72-82.
[101] Witchel H J. The hERG potassium channel as a therapeutic target[J]. Expert opinion on therapeutic targets, 2007, 11(3): 321-336.
[102] Brugada R, Hong K, Dumaine R, et al. Sudden death associated with short-QT syndrome linked to mutations in HERG[J]. Circulation, 2004, 109(1): 30-35.
[103] Curran M E, Splawski I, Timothy K W, et al. A molecular basis for cardiac arrhythmia: HERG mutations cause long QT syndrome[J]. Cell, 1995, 80(5): 795-803.
[104] Brown A M. Drugs, hERG and sudden death[J]. Cell calcium, 2004, 35(6): 543-547.
[105] Aronov A M. Predictive in silico modeling for hERG channel blockers[J]. Drug Discovery Today, 2005, 10(2): 149-155.
[106] Raschi E, Vasina V, Poluzzi E, et al. The hERG K+ channel: target and antitarget strategies in drug development[J]. Pharmacological research, 2008, 57(3): 181-195.
[107] Bains W, Basman A, White C. HERG binding specificity and binding site structure: evidence from a fragment-based evolutionary computing SAR study[J]. Progress in biophysics and molecular biology, 2004, 86(2): 205-233.
[108] Roche O, Trube G, Zuegge J, et al. A virtual screening method for prediction of the HERG potassium channel liability of compound libraries[J]. ChemBioChem, 2002, 3(5): 455-459.
[109] Broccatelli F, Mannhold R, Moriconi A, et al. QSAR modeling and data mining link torsades de pointes risk to the interplay of extent of metabolism, active transport, and hERG liability[J]. Molecular pharmaceutics, 2012, 9(8): 2290-2301.
[110] Sinha N, Sen S. Predicting hERG activities of compounds from their 3D structures: Development and evaluation of a global descriptors based QSAR model[J]. European journal of medicinal chemistry, 2011, 46(2): 618-630.
[111] Wang S, Li Y, Xu L, et al. Recent developments in computational prediction of HERG blockage[J]. Current topics in medicinal chemistry, 2013, 13(11): 1317-1326.
[112] Jing Y, Easter A, Peters D, et al. In silico prediction of hERG inhibition[J]. Future medicinal chemistry, 2015, 7(5): 571-586.
[113] Wang Y, Xiao J, Suzek T O, et al. PubChem's BioAssay database[J]. Nucleic acids research, 2012, 40(D1): D400-D412.
[114] Aronov A M, Goldman B B. A model for identifying HERG K+ channel blockers[J]. Bioorganic & medicinal chemistry, 2004, 12(9): 2307-2315.
[115] Li Q, Jørgensen F S, Oprea T, et al. hERG classification model based on a combination of support vector machine method and GRIND descriptors[J]. Molecular pharmaceutics, 2008, 5(1): 117-127.
[116] Studio D. version 2.5[J]. Accelrys Inc.: San Diego, CA, USA, 2009.
[117] Waring M J, Johnstone C. A quantitative assessment of hERG liability as a function of lipophilicity[J]. Bioorganic & medicinal chemistry letters, 2007, 17(6): 1759-1764.
[118] ChemicalComputingGroup M O E. Molecular Operating Environment[J]. 2008.
[119] Waring M J. Lipophilicity in drug discovery[J]. Expert opinion on drug discovery, 2010, 5(3): 235-248.
[120] Remko M, Boháč A, Kováčiková L. Molecular structure, pKa, lipophilicity, solubility, absorption, polar surface area, and blood brain barrier penetration of some antiangiogenic agents[J]. Structural Chemistry, 2011, 22(3): 635-648.
[121] Leo A, Hansch C, Elkins D. Partition coefficients and their uses[J]. Chemical reviews, 1971,

71(6): 525-616.
[122] NU S R R. Evaluation of the use of partition coefficients and molecular surface properties as predictors of drug absorption: a provisional biopharmaceutical classification of the list of national essential medicines of Pakistan[J]. Daru, 2011, 19(2).
[123] Souza E S, Zaramello L, Kuhnen C A, et al. Estimating the octanol/water partition coefficient for aliphatic organic compounds using semi-empirical electrotopological index[J]. International journal of molecular sciences, 2011, 12(10): 7250-7264.
[124] Cao D S, Xu Q S, Liang Y Z, et al. Prediction of aqueous solubility of druglike organic compounds using partial least squares, back-propagation network and support vector machine[J]. Journal of Chemometrics, 2010, 24(9): 584-595.
[125] Pollastri M P. Overview on the rule of five[J]. Current Protocols in Pharmacology, 2010: 9.12. 1-9.12. 8.
[126] Lipinski C A, Lombardo F, Dominy B W, et al. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings[J]. Advanced drug delivery reviews, 2012, 64: 4-17.
[127] Bhal S K, Kassam K, Peirson I G, et al. The rule of five revisited: applying log D in place of log P in drug-likeness filters[J]. Molecular pharmaceutics, 2007, 4(4): 556-560.
[128] Comer J, Tam K. Lipophilicity profiles: theory and measurement[J]. Testa, B.; van de Waterbeemd, H.; Folkers, G, 2001: 275-304.
[129] Ermondi G, Lorenti M, Caron G. Contribution of ionization and lipophilicity to drug binding to albumin: a preliminary step toward biodistribution prediction[J]. Journal of medicinal chemistry, 2004, 47(16): 3949-3961.
[130] Zhivkova Z, Doytchinova I. Quantitative structure–clearance relationships of acidic drugs[J]. Molecular pharmaceutics, 2013, 10(10): 3758-3768.
[131] Kah M, Brown C D. LogD: Lipophilicity for ionisable compounds[J]. Chemosphere, 2008, 72(10): 1401-1408.
[132] Wang J B, Cao D S, Zhu M F, et al. In silico evaluation of logD7. 4 and comparison with other prediction methods[J]. Journal of Chemometrics, 2015, 29(7): 389-398.
[133] Kennard R W, Stone L A. Computer aided design of experiments[J]. Technometrics, 1969, 11(1): 137-148.
[134] Kocjančič R, Zupan J. Modelling of the river flowrate: the influence of the training set selection[J]. Chemometrics and Intelligent Laboratory Systems, 2000, 54(1): 21-34.