ECoFFeS Manual

Release 1.0

Jiawei Huang, Yong Wang and Dongsheng Cao

February 12, 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

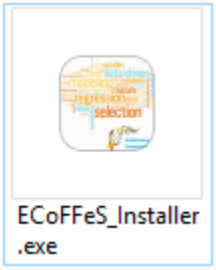
Step 1: Download

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

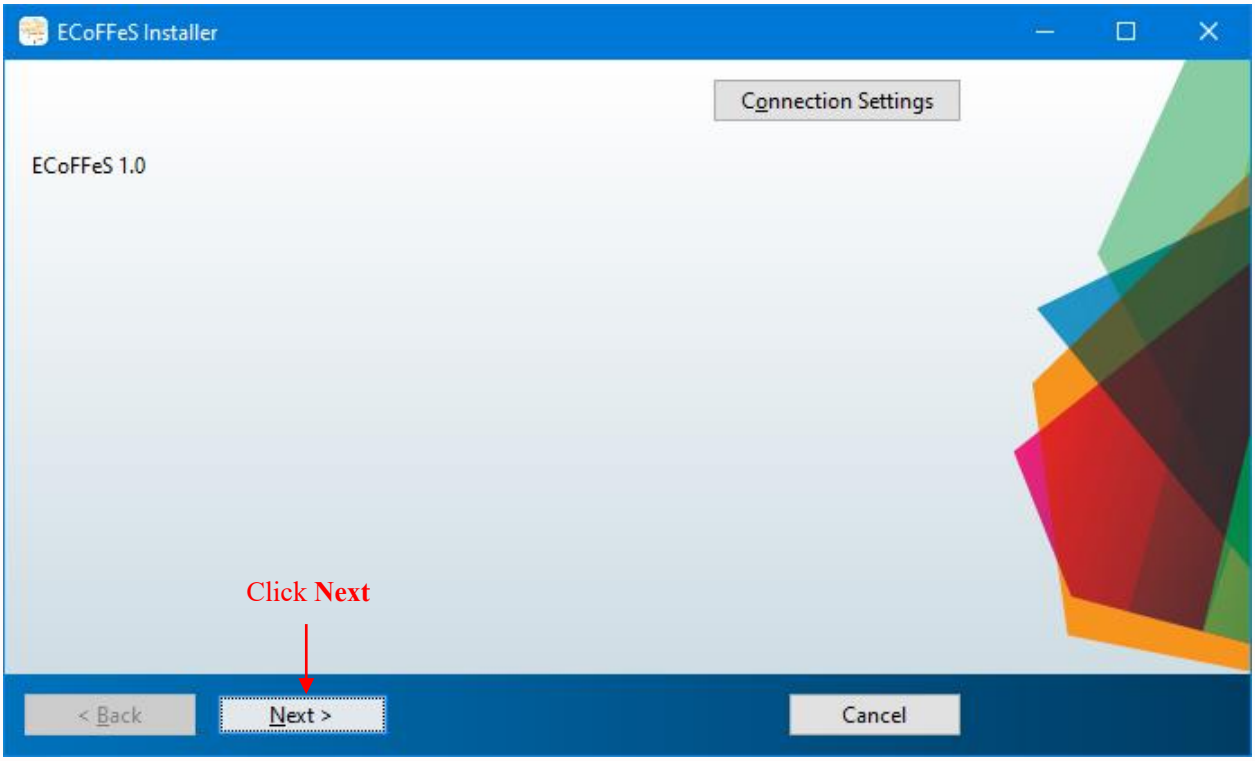
Step 2: Start the Installer

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

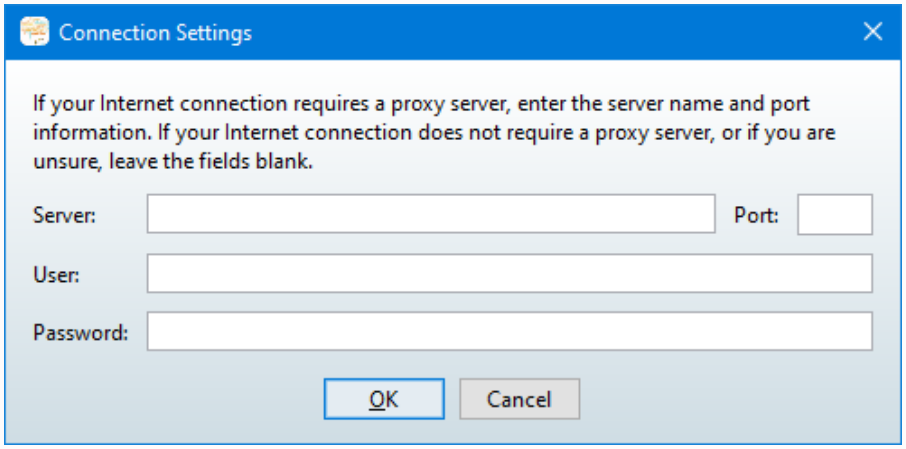


Step 3: ECoFFeS Installer

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.

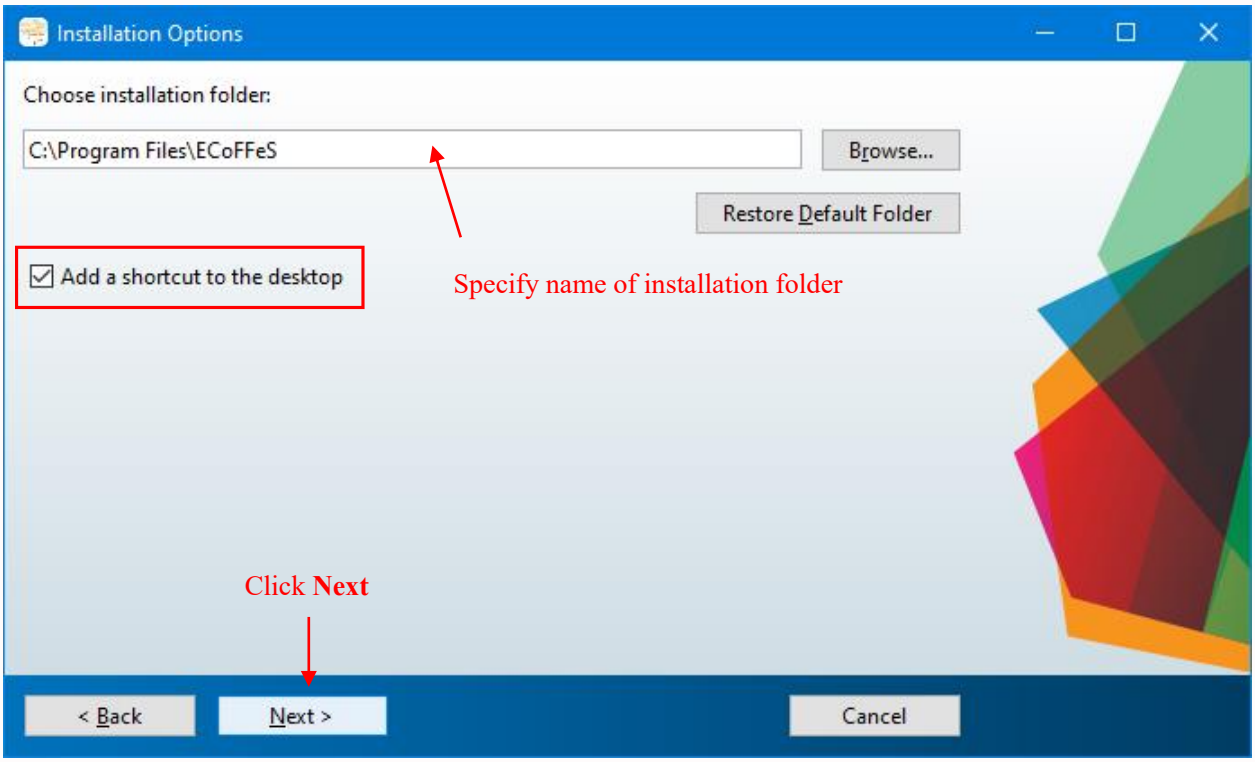


Note: On Windows systems, the installer uses your system proxy settings, by default. If your proxy server requires you to log in, the installer prompts you for your login information.

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.

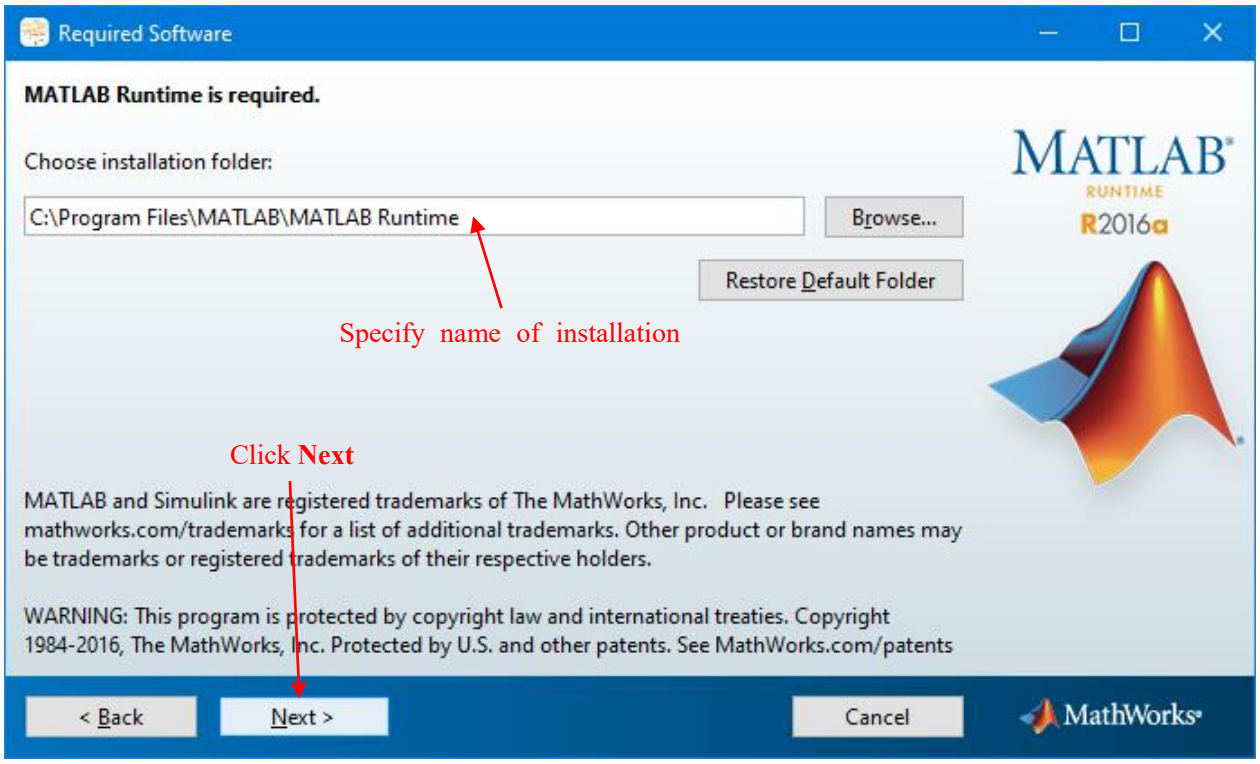
When specifying a folder name, you can use any alphanumeric character and some special characters, such as underscores. If you make a mistake while entering a folder name and want to start over, click **Restore Default Folder**. On Windows, the Installation Options dialog box lets you choose whether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



Step 5: Required Software

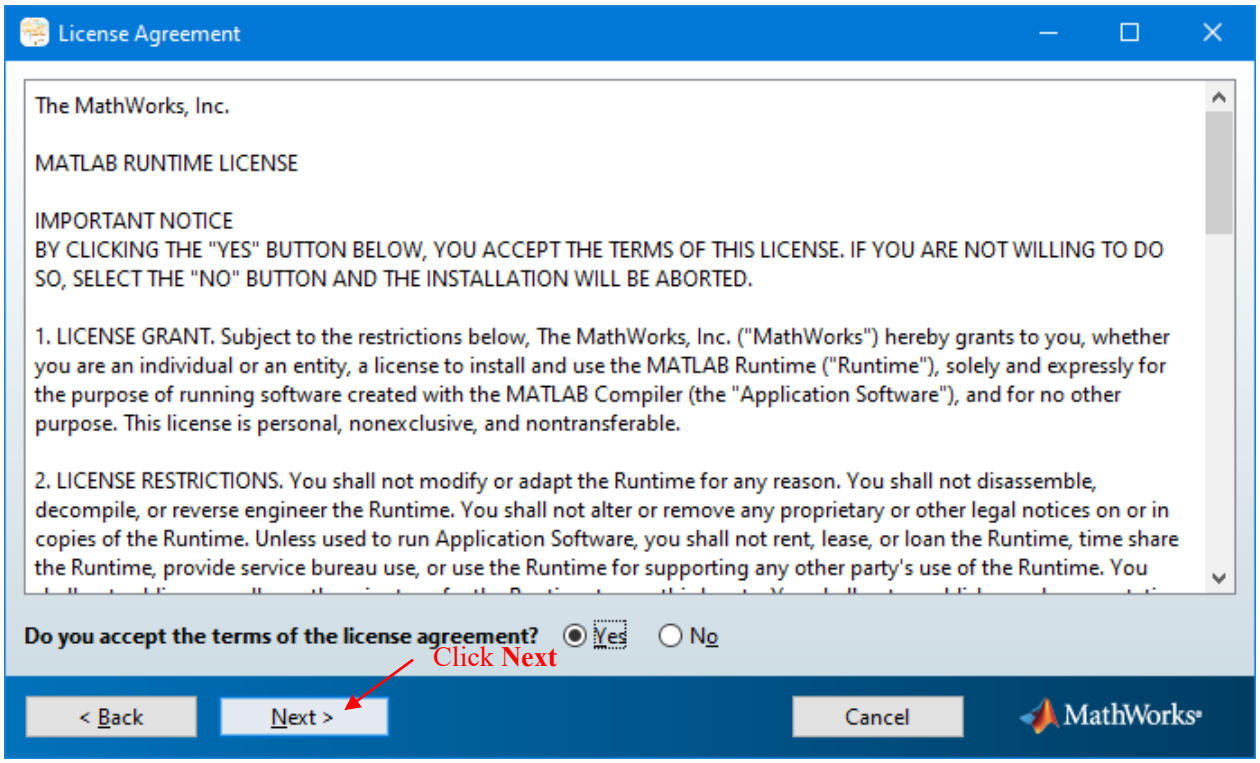
The MATLAB Runtime is a standalone set of shared libraries, MATLAB code, and other files that enable the execution of MATLAB programs on computers without an installed version of MATLAB. Specify the name of the folder where you want to install MATLAB Runtime. Accept the default installation folder, or 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.



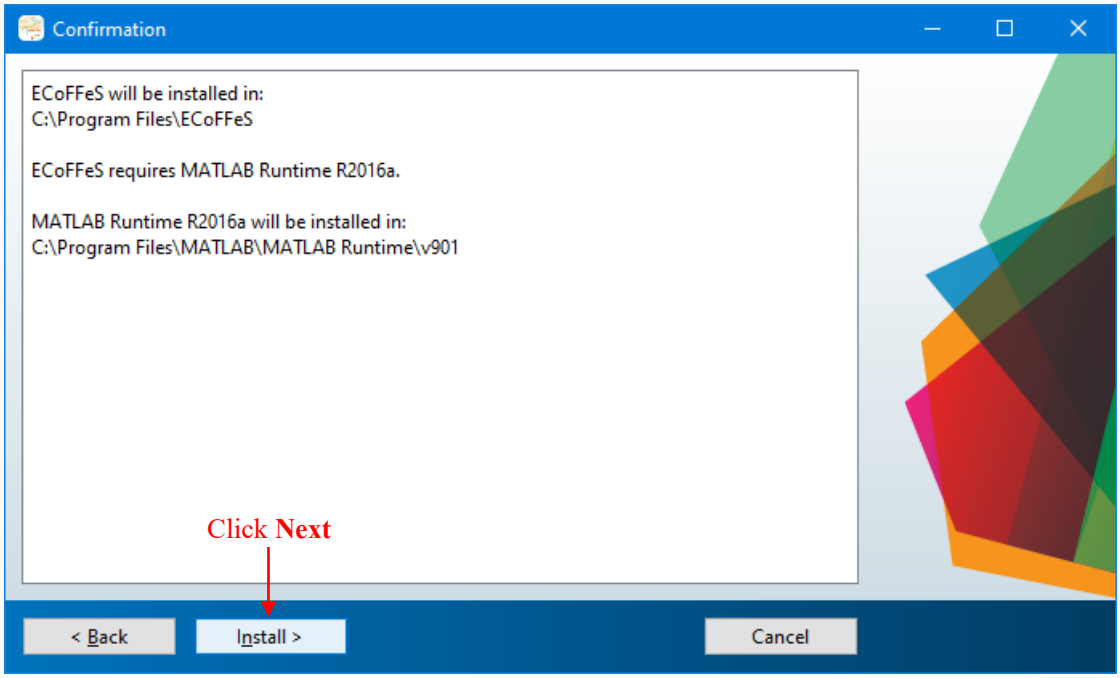
Step 6: License Agreement

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



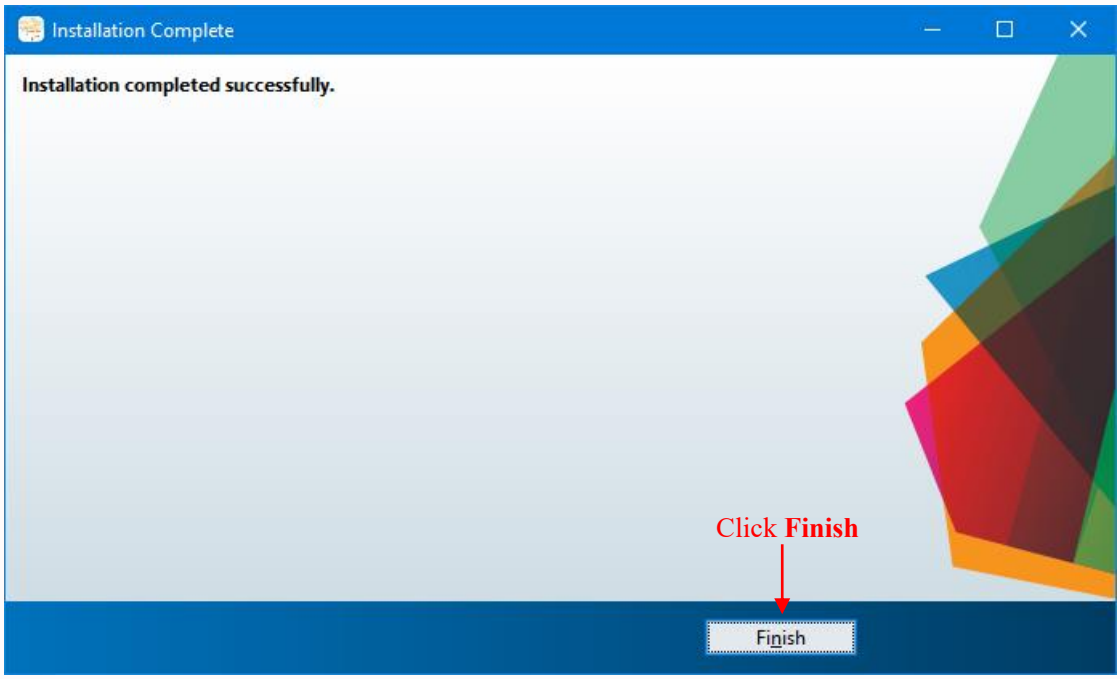
Step 7: Confirm Your Choices

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 is completed successfully, 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 [19]. 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 [19]. 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 the main interface, 'Subset Discovery' selects **'SOEAs'**, 'Subset Evaluation' selects **'Regression Model & Metric'**, click **Start**, then the secondary interface **'SOEAs_Regression'** is 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 at the end of a run.

3) **Figure**: show the charts at the end of a run.

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, and 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, and menu).

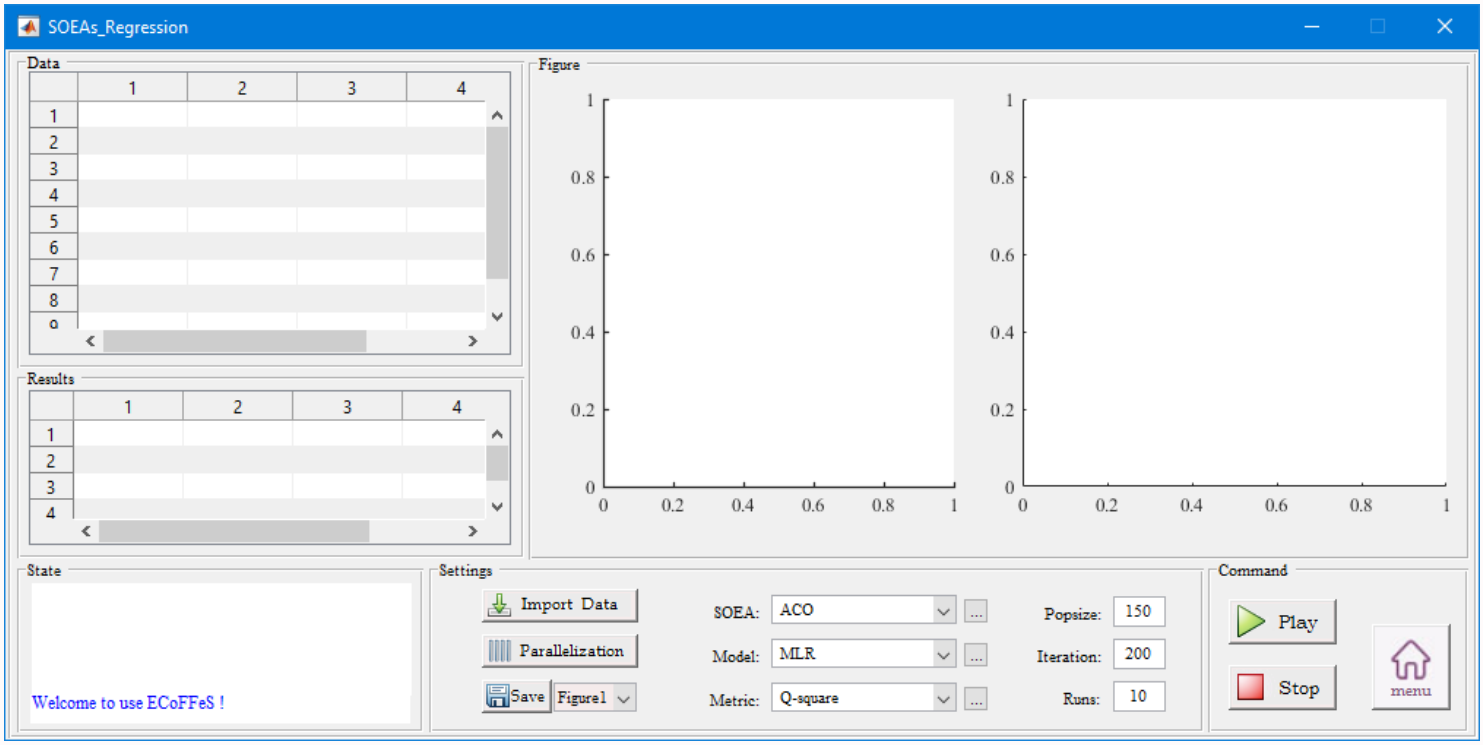


Figure 3.2 The secondary interface: 'SOEAs_Regression'

3.2.2 MOEAs_Regression

In the main interface, 'Subset Discovery' selects **'MOEAs'**, 'Subset Evaluation' selects **'Regression Model & Metric'**, click **Start**, then the secondary interface **'MOEAs_Regression'** is 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 at the end of a run.

3) **Figure**: show the charts at the end of a run.

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, and 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, and menu).

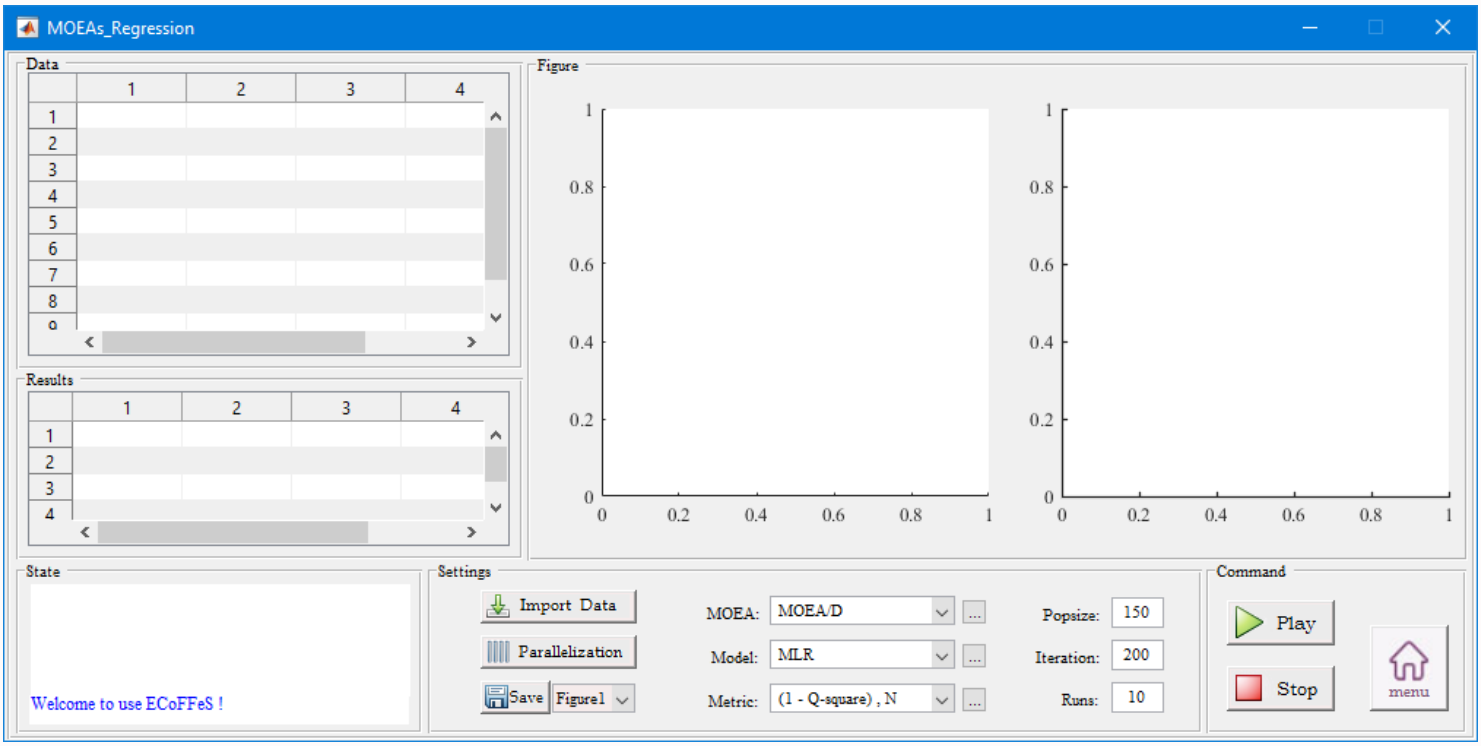


Figure 3.3 The secondary interface: MOEAs_Regression

3.2.3 SOEAs_Classification

In the main interface, 'Subset Discovery' selects **'SOEAs'**, 'Subset Evaluation' selects **'Classification Model & Metric'**, click **Start**, then the secondary interface **'SOEAs_Classification'** is 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 at the end of a run.

3) **Figure**: show the charts at the end of a run.

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, and 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, and menu).

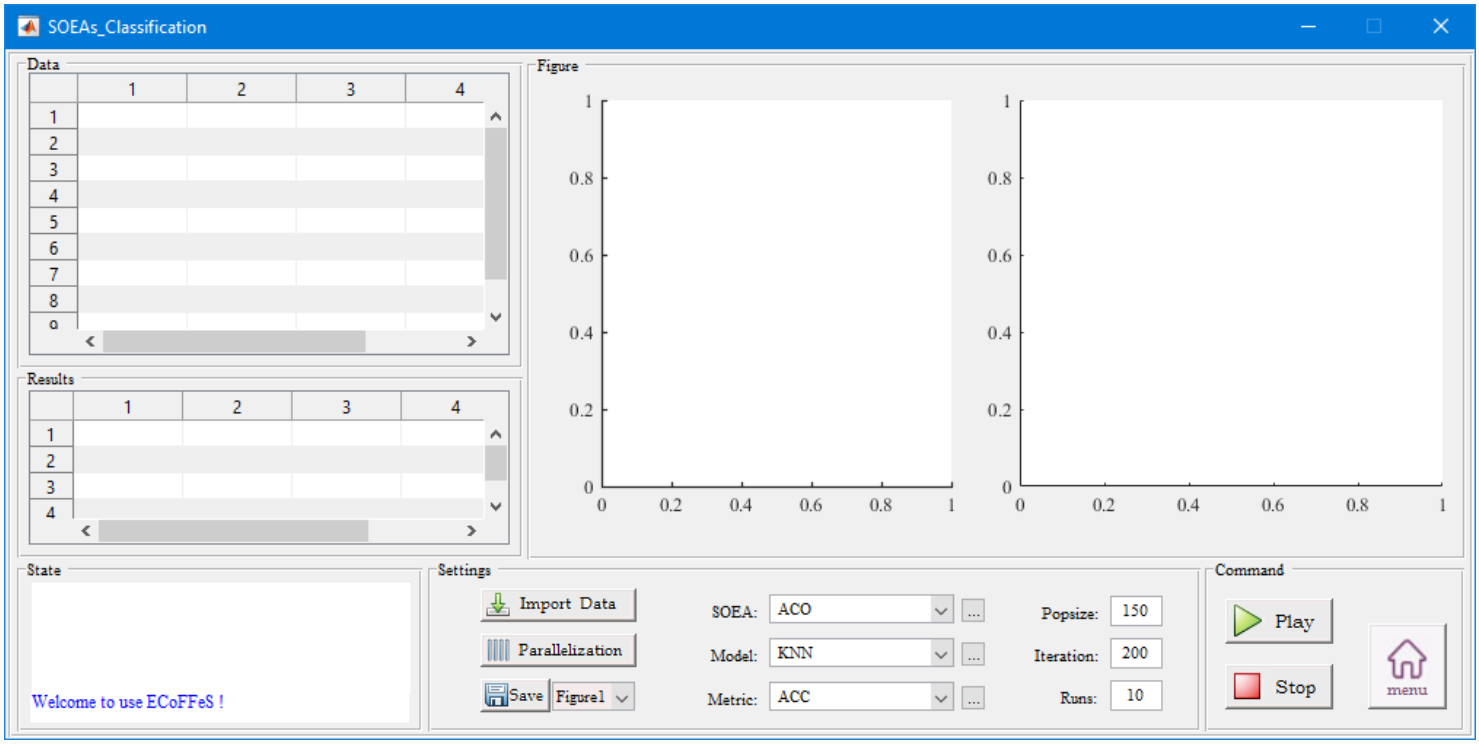


Figure 3.4 The secondary interface: SOEAs_Classification

3.2.4 MOEAs_Classification

In the main interface, 'Subset Discovery' selects **'MOEAs'**, 'Subset Evaluation' selects **'Classification Model & Metric'**, click **Start**, then the secondary interface **'MOEAs_Classification'** is 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 at the end of a run.

3) **Figure**: show the charts at the end of a run.

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, and 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, and menu).

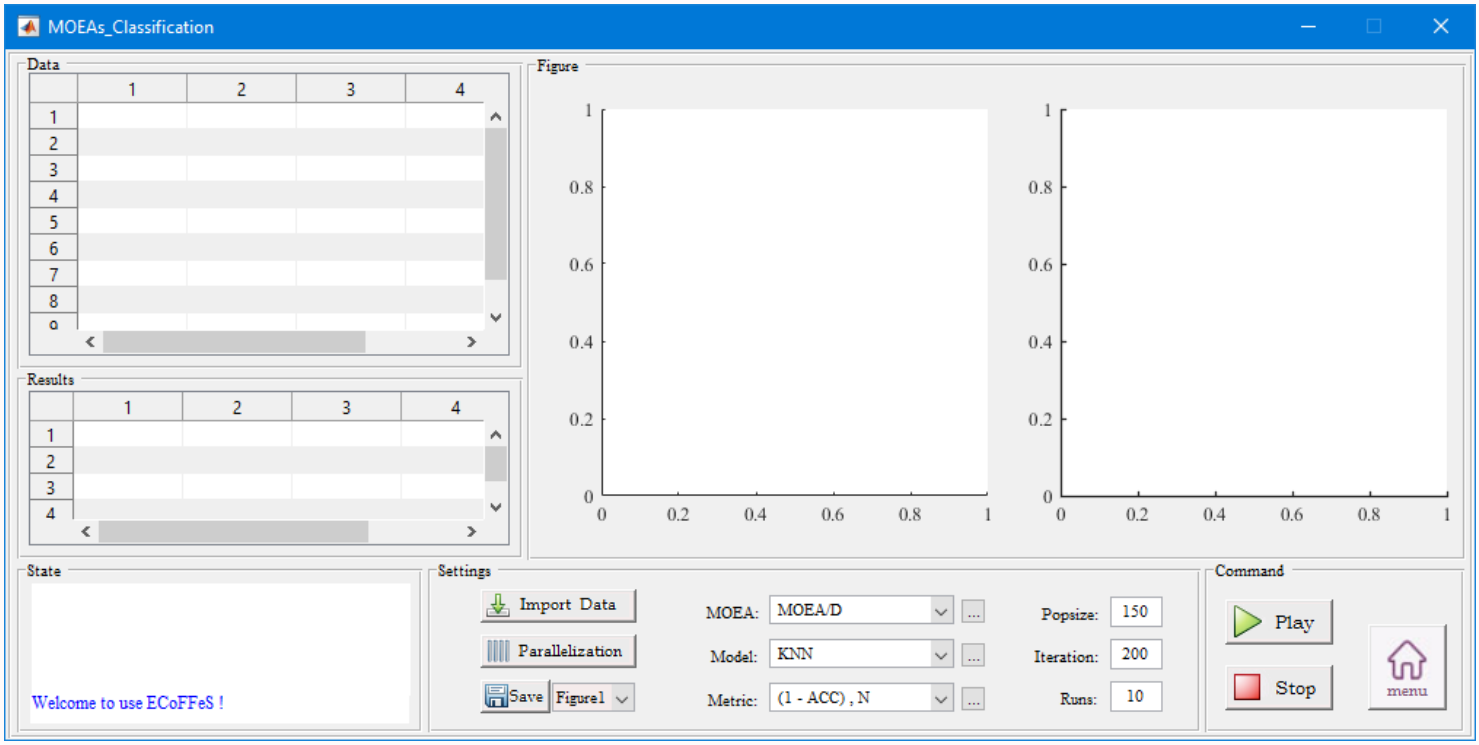


Figure 3.5 The secondary interface: MOEAs_Classification

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 its 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 found in [6-8].

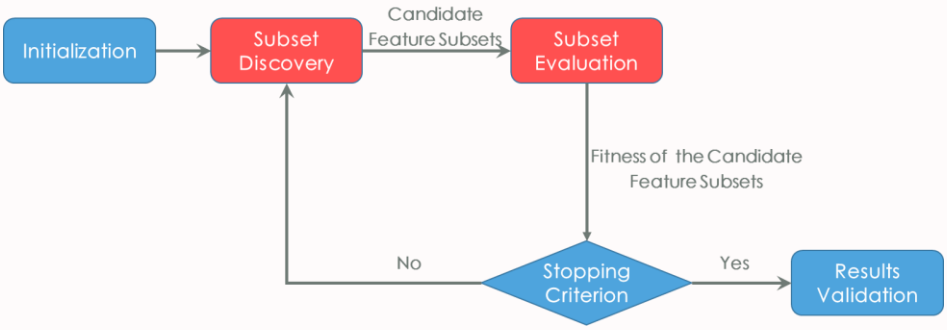


Figure 4.1 A general feature selection process

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.

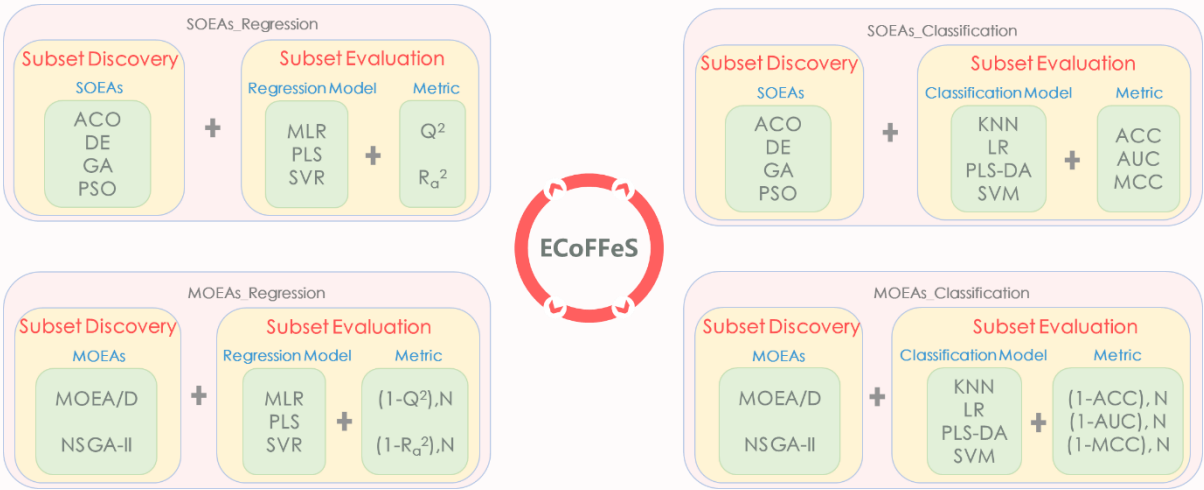


Figure 4.2 The internal structure of ECoFFeS

4.2 Subset Discovery: Evolutionary Computation

Currently, there is no feasible way to select the optimal feature subset due to the fact that feature selection 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" leads to the computational load growing exponentially as the total number of features increases [2][5]. Indeed, if we use simple enumeration, it is impossible even for the most powerful computers to search the optimal feature subset in the case of n>30. To deal with this issue, a variety of search methods has been proposed to feature selection, such as complete search, greedy search, heuristic search and random search [6][11-14]. However, most existing feature selection methods suffer from stagnation in a local optimum or high computational cost [15][16]. Therefore, an efficient global search method is needed to better solve feature selection problems. Evolutionary computation (EC) approaches have recently received much attention from the feature selection research community as they are well-known for their global search ability [19]. Compared with traditional search methods, EC approaches 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, or differentiable. Another significant advantage of EC approaches is that their population-based mechanism can produce multiple solutions in a single run. This is particularly suitable for multi-objective feature selection to find a set of non-dominated solutions with the trade-off between the number of features and the performance metric [19]. Figure 4.3 shows the categories of EC for feature subset discovery.

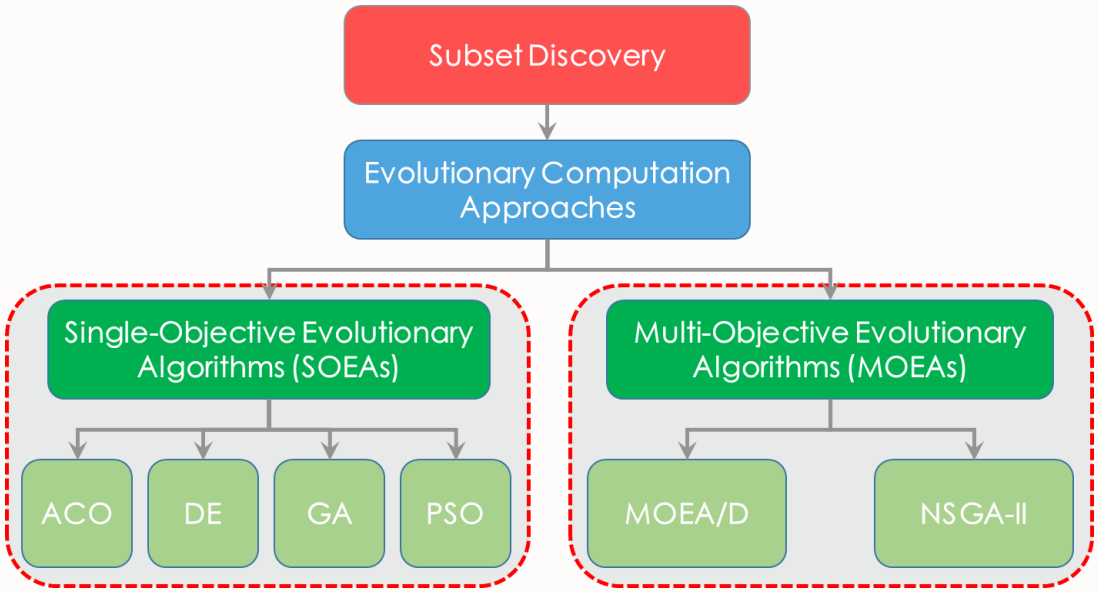


Figure 4.3 The categories of EC for feature subset discovery

4.2.1 Single-Objective Evolutionary Algorithms (SOEAs)

4.2.1.1 Ant Colony Optimization (ACO)

Ant colony optimization (ACO) 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 Dorigo in 1992 [20], 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 that an ant chooses a path increases with the increase of the number of ants that previously chose the same path [21][22].

To some extent, a feature selection problem is similar to the process that the ants find the routes between nest and food [23][24]. As depicted in Figure 4.4, the feature F_i $(i = 1,2,\dots,n)$ corresponds to two routes, i.e., the light green route r_{i1} and the blank route r_{i0}. When ant k $(k = 1,2,\dots,m)$ comes to a crossroad, it has two choices, i.e., route r_{i1} and route r_{i0}. If ant k chooses route r_{i1}, it means that the feature F_i is selected. Otherwise, ant k chooses route r_{i0}, which indicates that the feature F_i is not selected. After the tour from the nest to the food, n traversed routes are obtained and it is easy to obtain the selected feature subset FS_k $(k = 1,2,\dots,m)$. For instance, suppose $n = 3$ and the three 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 FS_k, it is easy to calculate its fitness f_k $(k = 1,2,\dots,m)$. Thus, the pheromone laid on n routes that traversed by ant k is updated by f_k. As time goes on, m ants repeatedly traverse from the nest to the food and the pheromone of each route r_{ij} gradually becomes stable. Finally, all m ants choose n routes with high pheromone and the resultant feature subset FS is acquired based on these n routes.

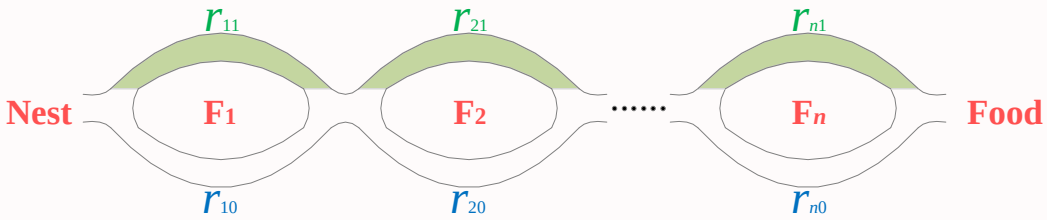


Figure 4.4 The route graph between nest and food

Based on the above background, next we will introduce a modified ACO [24]. The main characteristic of this modified ACO is that, at each iteration, the pheromone values are updated not

only by all the m ants, but also by an additional ant called best-so-far ant. This leads to the exploitation of ants around the optimal solution in subsequent iterations. The pheromone $\tau_{ij}(t)$, associated with route r_{ij}, is updated as follows:

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

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

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

where ρ is the evaporation rate, 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, $\Delta\tau_{ij}^{best}$ is additional pheromone deposited by the best-so-far ant, $f_k(t)$ is the fitness of ant k at iteration t, and $N_{ij}(t)$ is the number of ants that traverse through route r_{ij}. In Eq. 4.1, ρ is used to avoid unlimited accumulation of the pheromone trails and enables the algorithm to 'forget' bad decisions previously taken.

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

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

where N_i^k is the feasible route of ant k, $N_i^k = \{0,1\}$, and α and β are two parameters determining the relative influence of the pheromone and the heuristic information η_{ij}, respectively. The role of the parameters α and β is the following. If $\alpha = 0$, the feature F_i is less likely to be selected, which 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. In particular, $\alpha > 0$ will further leads to the rapid emergence of a stagnation situation. As a result, all the ants follow the same path and construct the same tour, which is always suboptimal. In addition, η_{ij} is defined as:

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

where pp_{ij} is the prior probability of choosing route r_{ij}. In our experiment we assume that the probability of not selecting the feature F_i is bigger than the probability of selecting. Specifically, $pp_{i0} = 0.55$ and $pp_{i1} = 0.45$ $(i = 1,2,\dots,n)$. The following is the pseudocode of ACO.

Algorithm: ACO

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 FS_k $(k = 1,2, \dots, m)$ using n routes traversed by ant k;
6: Evaluate each feature subset FS_k and use f_k as the fitness value;
7: Update the best feature subset FS_{best} and f_{best};
8: Update the pheromone $\tau_{ij}(t+1)$ according to Eq. 4.1;
9: $t = t + 1$;
10: **Stopping Criterion:** If the maximum number of fitness evaluations is reached, then stop and output the best feature subset FS_{best} and its fitness f_{best}; otherwise go to step 4.

The following is the flow chart of ACO.

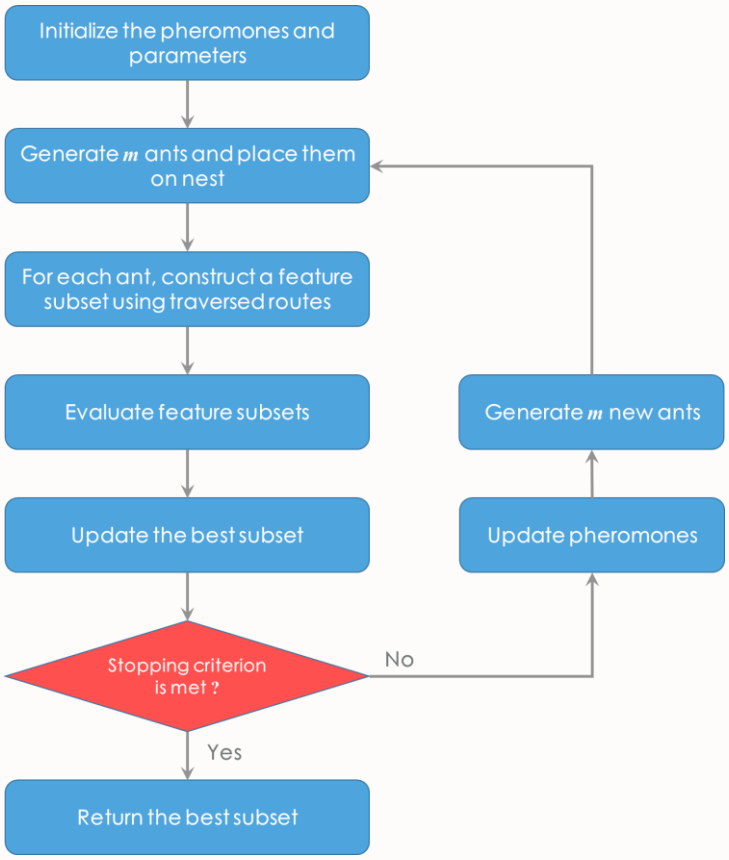


Figure 4.5 The flow chart of ACO

4.2.1.2 Differential Evolution (DE): a modified DE

Differential evolution (DE), proposed by Storn and Price in 1995 [25][26], 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 also 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 [27][28].

Due to the fact that a feature selection problem is a discrete optimization problem, it cannot be solved by DE directly. Thus, a DE variant called binary differential evolution (BDE) is proposed in [29]. This binary approach enables DE to perform optimization in binary spaces while maintaining the basic structure of DE.

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 $\vec{x}_{i,G} = (x_{i,1,G}, \dots, x_{i,D,G})$, $x_{i,j,G}$ is the jth element which is a floating-point number. Then, the corresponding binary-digits solution $\overrightarrow{xb}_{i,G} = (xb_{i,1,G}, \dots, xb_{i,D,G})$ can be obtained via the following transformation:

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

$$S(x_{i,j,G}) = {}^{1}/_{(1 + e^{-x_{i,j,G}})} \tag{4.7}$$

where $U(0,1)$ is a uniformly distributed random number in the interval (0,1) and S is the sigmoid function. Afterward, $\overrightarrow{xb}_{1,G}, \dots, \overrightarrow{xb}_{N,G}$ form population Pb_G. Finally, each individual $\overrightarrow{xb}_{i,G}$ in Pb_G is evaluated by the fitness function: $f(\overrightarrow{xb}_{i,G})$. It is evident that $f(\overrightarrow{xb}_{i,G})$ also represents the fitness of $\vec{x}_{i,G}$. A brief description of the transformation process from P_G to Pb_G is given in Figure 4.6.

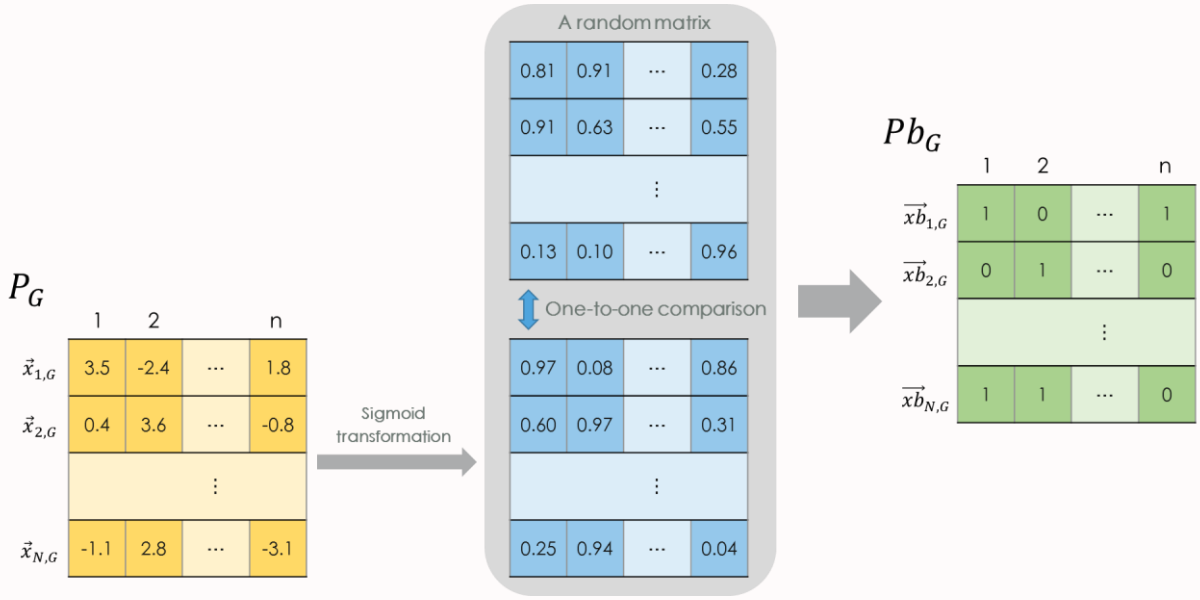


Figure 4.6 A brief description of the transformation process

Next, we will introduce how to evolve population P_G in BDE. For each individual $\vec{x}_{i,G}$ in P_G, the mutation operator named DE/current-to-best/1 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 $r1$ and $r2$ are mutually different integers chosen from $[1, N]$ and also different from i, $\vec{x}_{best,G}$ is the best individual at generation G, F is the scaling factor, and $\vec{v}_{i,G}$ is the mutant vector.

After the mutation, if the jth element $v_{i,j,G}$ of the mutant vector $\vec{v}_{i,G} = (v_{i,1,G}, \dots, v_{i,D,G})$ is out of the search region $[l_{min,j}, l_{max,j}]$, then $v_{i,j,G}$ is reset as follow:

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

where $l_{min,j}$ and $l_{max,j}$ are the lower and upper bounds of $v_{i,j,G}$, respectively.

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

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

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 $\vec{u}_{i,G}$. Figure 4.7 shows the schematic of the binomial crossover.

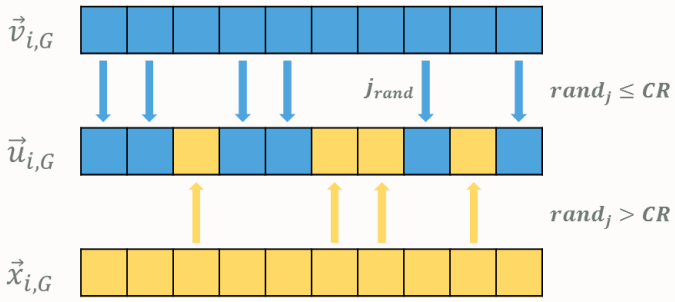


Figure 4.7 The schematic of the binomial crossover

Subsequently, the selection is performed between $\vec{x}_{i,G}$ and $\vec{u}_{i,G}$, and the better one will survive into the next population P_{G+1}

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

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

In order to further improve the performance of BDE, we propose a feedback strategy, which incorporates the information of the binary-digits population Pb_{G+1} into the original population P_{G+1} as follows:

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

where $\overrightarrow{xb}_{best,G+1}$ is the best binary-digits individual in Pb_{G+1}. Figure 4.8 shows the process of combining BDE with the proposed feedback strategy (BFDE) for feature selection.

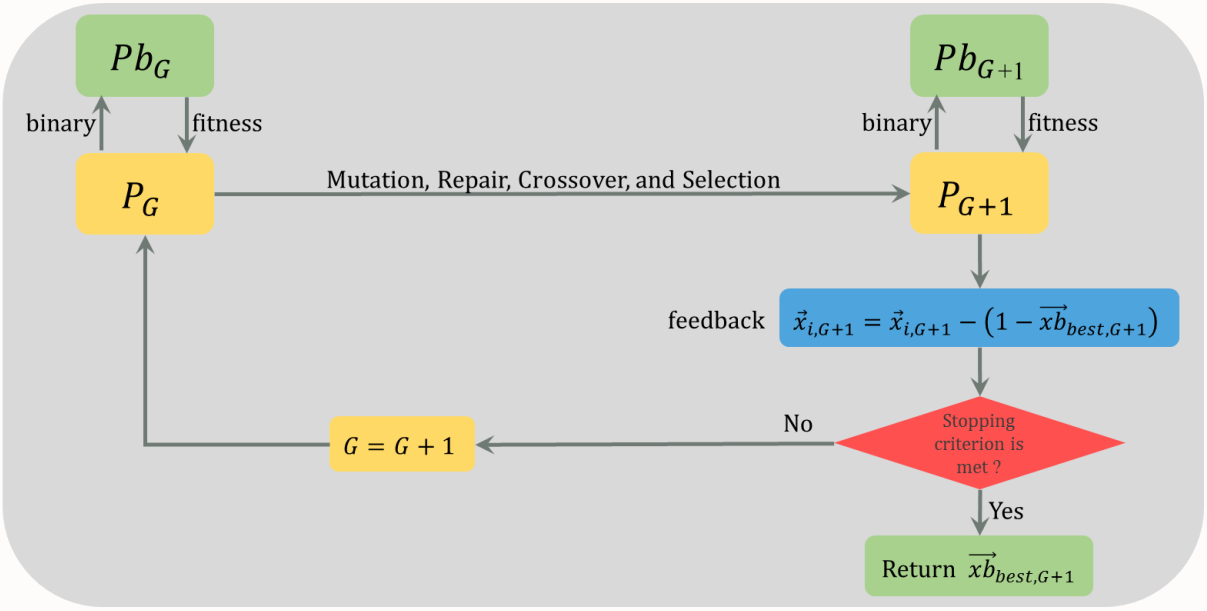


Figure 4.8 The process of BFDE

The following is the pseudocode of BFDE.

Algorithm: BFDE-FS

1: $G = 1$; /* G denotes the generation number */
2: Randomly generate an initial population P_G: $\vec{x}_{1,G}, \ldots, \vec{x}_{N,G}$;
3: Compute the corresponding binary-digits population Pb_G via Eq. 4.6 and Eq. 4.7, and evaluate the fitness of each individual in Pb_G: $f(\overrightarrow{xb}_{1,G}), \ldots, f(\overrightarrow{xb}_{N,G})$. /* $f(\overrightarrow{xb}_{i,G})$ also represents the fitness of $\vec{x}_{i,G}$ */
4: $P_{G+1} = \emptyset$ and $Pb_{G+1} = \emptyset$;
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), repair $\vec{v}_{i,G}$ via Eq. 4.9; /*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: Compute the corresponding binary-digits individual $\overrightarrow{ub}_{i,G}$ of $\vec{u}_{i,G}$ via Eq. 4.6 and Eq. 4.7, and evaluate its fitness, i.e., $f(\overrightarrow{ub}_{i,G})$;
10: If $f(\overrightarrow{ub}_{i,G}) \leq f(\overrightarrow{xb}_{i,G})$, set $\vec{x}_{i,G+1} = \vec{u}_{i,G}$ and $\overrightarrow{xb}_{i,G+1} = \overrightarrow{ub}_{i,G}$; otherwise, set $\vec{x}_{i,G+1} = \vec{x}_{i,G}$ and $\overrightarrow{xb}_{i,G+1} = \overrightarrow{xb}_{i,G}$. Afterward, store $\vec{x}_{i,G+1}$ and $\overrightarrow{xb}_{i,G+1}$ into P_{G+1} and Pb_{G+1}, respectively; /*Selection*/
11: **End For**
12: Implement the feedback strategy on P_{G+1} via Eq. 4.12;
13: $G = G + 1$;
14: **Stopping Criterion:** If the maximum number of fitness evaluations is reached, then stop and output the best individual in Pb_G; otherwise go to step 3.

4.2.1.3 Genetic Algorithm (GA)

Genetic algorithm (GA) was developed by John Holland in the 1960s and first published in 1975 [30][31]. As a population-based heuristic method, GA is inspired by the process of natural evolution. In GA, a solution is represented as a chromosome. Each chromosome in the population is associated with a fitness value. A pre-defined number of iterations of evolution follows the initial population generation. At each iteration, some pairs of chromosomes are selected by a biased random selection approach. Chromosomes with the higher fitness values are more likely to be selected from the population. A crossover operator is implemented on each pair of selected chromosomes to generate new chromosomes. Some other chromosomes are also selected from the population, followed by a mutation operator that also generates some new chromosomes. At each iteration of GA, the fitness values of all chromosomes in the population are calculated, and the best chromosome is recorded. After a large number of iterations, the best chromosome in the population will be regarded as the satisfied solution [32][33].

According to the characteristics of a feature selection problem, a string with n binary digits is used as chromosome representation. For each binary digit, value 1 means that the corresponding feature is selected and value 0 has the opposite meaning. Next, we introduce a modified GA for feature selection [34]. In the first step of the modified GA, an initial population is randomly generated: $Pb_G = \{\overrightarrow{xb}_{i,G} = (xb_{i,1,G}, \dots, xb_{i,D,G}), i = 1, \dots, N\}$, where N is the population size and $\overrightarrow{xb}_{i,G}$ is a binary string with D dimensions.

Afterward, a proportional roulette wheel selection is implemented, in which chromosomes are selected with a probability that is directly proportional to their fitness values, i.e., a chromosome's selection probability corresponds to the percentage of a portion of a roulette wheel [34]. Obviously, those with larger fitness will be selected with a higher probability. The proportional roulette wheel selection is depicted in Figure 4.9. As shown in Figure 4.9, when the wheel is spun, the wheel will finally stop and the pointer attached to it will point on one of the segments. By repeating this, the better chromosomes will be chosen more often than the poorer ones, thus fulfilling the requirements of survival of the fittest. Let $f(\overrightarrow{xb}_{1,G}), \dots, f(\overrightarrow{xb}_{N,G})$ be the fitness values of chromosomes $\overrightarrow{xb}_{1,G}, \dots, \overrightarrow{xb}_{N,G}$. 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 gives a chance to all of individuals to be selected. Therefore, diversity of the population can be preserved.

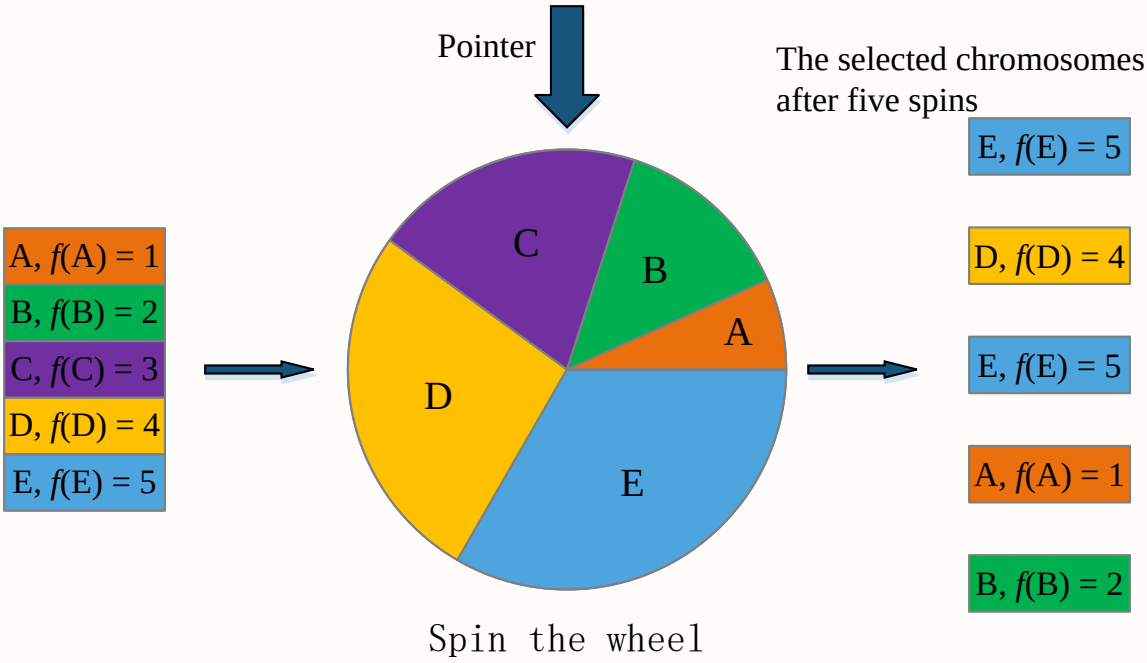


Figure 4.9 The proportional roulette wheel selection

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 merged to yield two new offspring [35]. A two-point crossover operator contains two randomly chosen crossover points $Cpoint_1$ and $Cpoint_2$, where $1 \leq Cpoint_1 < Cpoint_2 \leq n$. Then, an uniformly distributed random number $rand$ is generated in the interval $[0,1]$. If $rand < pc$, where pc is the crossover probability, then chromosomes exchange the segments located between these two points. Finally, two new offspring are created and put back into the chromosome pool. The crossover operator is explained in Figure 4.10.

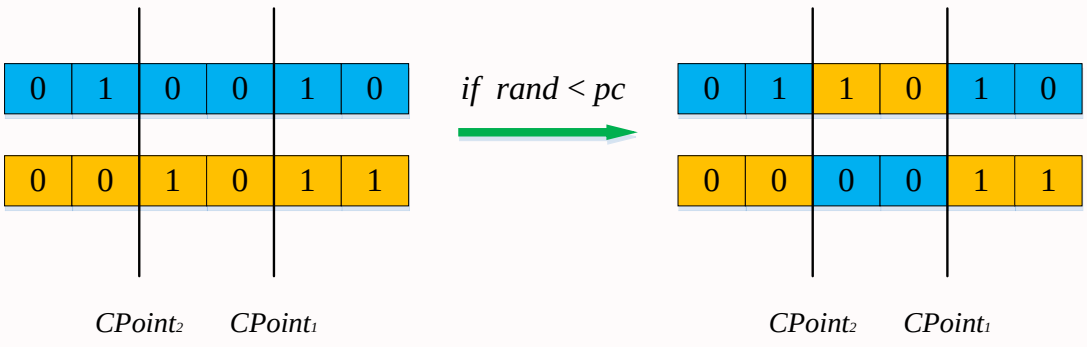


Figure 4.10 Two-point crossover operator

The mutation operator introduces new genetic structures into the population by randomly modifying some of genes, such that GA can escape from a local optimum and avoid converging too fast. In other words, the mutation operator gives GA an opportunity to search for new and more feasible chromosomes in new areas of the solution space [35]. The probability that each chromosome

performs the mutation operator is pm. For each chromosome $\overrightarrow{xb}_{i,G} = (xb_{i,1,G}, \dots, xb_{i,D,G})$, D uniformly distributed random numbers $rand_1, \dots, rand_n$ are generated in the interval [0,1]. If $rand_i < pm$, $xb_{i,j,G} = 1 - xb_{i,j,G}$. An example of the mutation operator is shown in Figure 4.11.

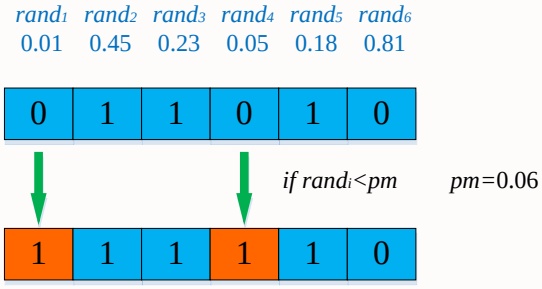


Figure 4.11 Mutation operator

The following is the pseudocode of GA.

Algorithm: GA

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

2: Randomly generate an initial population Pb_G consisting of N chromosomes $\overrightarrow{xb}_{1,G}, \overrightarrow{xb}_{2,G}, \dots, \overrightarrow{xb}_{N,G}$;

3: Evaluate each chromosome in 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 Pb_G by the proportional roulette wheel selection;

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

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

8: Combine Mb_G with $\overrightarrow{xb}_{best,G}$ 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 fitness evaluations 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) was first introduced by Kennedy and Eberhart as an optimization technique for continuous problems in 1995 [36-38]. 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 ease to implementation, PSO has been widely used in a variety of optimization problems. In 1997, Kennedy and Eberhart adapted the standard continuous PSO to binary spaces and this variant was termed as the binary PSO (BPSO) [39]. In PSO, there are two update functions: the velocity update function and the position update function [40]. Originally, the position was updated by combining its current position with velocity, but in BPSO, the position is updated based only on the current velocity. Generally, the sigmoid function is used to update the position in BPSO.

For a feature selection problem, a modified BPSO is proposed [41]. In the first step of this modified BPSO, an initial population of particles is randomly generated. Particles move in the search space to search for the optimal solution by updating the position based on the experience of its own and its neighbor particles. During movement, the current position of particle i is represented by $\overrightarrow{xb_{i,G}} = \left(xb_{i,1,G}, \dots, xb_{i,D,G}\right)$, where D is the dimension of the search space. The velocity of particle i is represented as $\vec{v}_{i,G} = \left(v_{i,1,G}, \dots, v_{i,n,G}\right)$. The best previous position of particle i is recorded as the personal best $\overrightarrow{pbestb_{i,G}}$ and the best position obtained by the swarm so far is called the global best $\overrightarrow{gbestb_G}$. The modified BPSO searches for the optimal solution by updating the position and the velocity of each particle according to the following equations:

$$v_{i,j,G+1} = \omega \cdot v_{i,j,G} + c_1 \cdot r_1\left(pbestb_{i,j,G} - xb_{i,j,G}\right) + c_2 \cdot r_2\left(gbestb_{j,G} - xb_{i,j,G}\right) \tag{4.14}$$

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

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

where G denotes the generation number in the evolutionary process, j denotes the jth dimension in the search space, ω is the inertia weight 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 $[0,1]$, $\left(xb_{i,j,G} + v_{i,j,G+1}\right)$ is limited by a predefined interval $\left[-l_{max,j}, l_{max,j}\right]$, and $S(\cdot)$ is sigmoid transformation described in Figure 4.12. $\left|xb_{i,j,G} + v_{i,j,G+1}\right| < l_{max,j}$ simply limits

the ultimate probability that bit $xb_{i,j,G+1}$ is equal to zero or one. For instance, if $l_{max,j} = 4.0$, then probabilities will be limited to $S(xb_{i,j,G} + v_{i,j,G+1}) \in [S(-4),\ S(4)]$, i.e., $S(xb_{i,j,G} + v_{i,j,G+1}) \in [0.018,\ 0.982]$ [39].

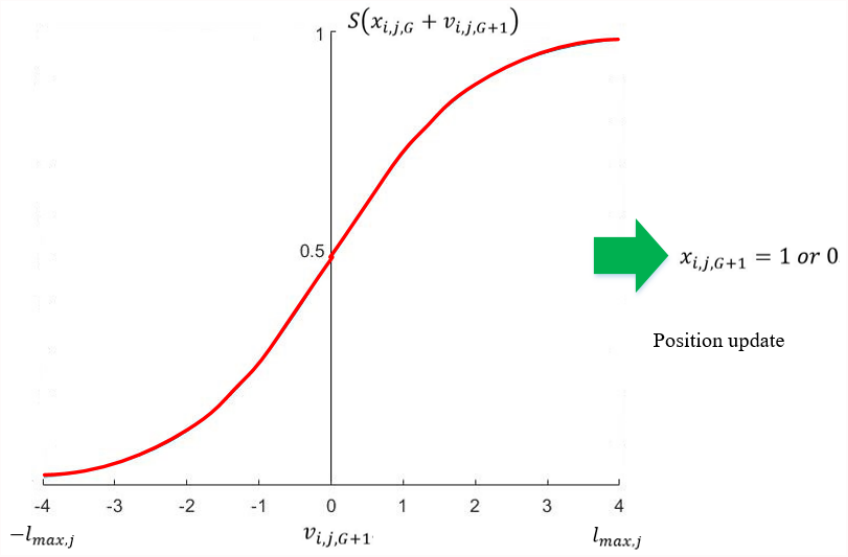


Figure 4.12 Sigmoid transformation

The following is the pseudocode of modified BPSO.

Algorithm: modified BPSO

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

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

3: Evaluate each particle in Pb_G and obtain the global best position $\overrightarrow{gbestb}_G$.

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

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

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

7: Generate the 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 $\overrightarrow{xb}_{i,G+1}$ by the fitness function: $f(\overrightarrow{xb}_{i,G+1})$;

10: Update personal best position $\overrightarrow{pbestb}_{i,G+1}$ and the global best position $\overrightarrow{gbestb}_{G+1}$ based on $f(\overrightarrow{xb}_{i,G+1})$. /*$\overrightarrow{pbestb}$ and $\overrightarrow{gbestb}$ update*/

11: **End For**

12: $G = G + 1$;

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

4.2.2 Multi-Objective Evolutionary Algorithms (MOEAs)

Feature selection involves two main objectives, i.e., minimizing the evaluation metric and minimizing the number of features. Indeed, they are always in conflict with each other. Therefore, feature selection can be treated as a multi-objective optimization problem to find a set of trade-off solutions between these two objectives. The research on this direction has gained much attention in recent years, where EC approaches contribute the most since EC approaches are a kind of population-based heuristic methods and particularly suitable for multi-objective optimization [19][42].

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

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

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

The objectives in Eq. (4.17) frequently 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. In fact, the best trade-off solutions, called the Pareto optimal solutions, are important to a decision maker. The Pareto optimality concept, which was first proposed by Edgeworth and Pareto [43], is formally defined as follows [44][48].

- ✓ Definition 1. A vector $\vec{u} = (u_1, \dots, u_m)$ is said to dominate another vector $\vec{v} = (v_1, \dots, v_m)$, denoted as $\vec{u} \prec \vec{v}$, if $\forall i \in \{1, \dots, m\}$, $u_i \leq v_i$ and $\vec{u} \neq \vec{v}$.
- ✓ Definition 2. A feasible solution $\vec{x}^* \in \Omega$ of Eq. (4.17) is called a Pareto optimal solution, if $\nexists \vec{y} \in \Omega$ such that $\vec{F}(\vec{y}) \prec \vec{F}(\vec{x}^*)$. The set of all the Pareto optimal solutions is called the Pareto set (PS), denoted as $\text{PS} = \{\vec{x} \in \Omega | \nexists \vec{y} \in \Omega, \vec{F}(\vec{y}) \prec \vec{F}(\vec{x})\}$.
- ✓ The image of the PS in the objective space is called the Pareto front (PF): $\text{PF} = \{\vec{F}(\vec{x}) | \vec{x} \in \text{PS}\}$.

Next, we will introduce two popular and representative multi-objective EAs for feature selection.

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

For feature selection, it is needed to provide the trade-offs between the two objectives (i.e., minimizing the evaluation metric and minimizing number of features). Multi-objective EAs can achieve this by providing a set of Pareto optimal solutions (i.e., a set of feature subsets), which allows a decision-maker to choose a preferred solution (i.e., a preferred feature subset) according to his/her own requirements. As a representative of the decomposition-based approaches [45], MOEA/D explicitly decomposes a MOP into a number of 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 for each subproblem. The neighborhood relationships among these subproblems are defined based on the distances between their aggregation coefficient vectors. The optimal solutions of two neighboring subproblems should be very similar. Each subproblem (i.e., a scalar aggregation function) is optimized in MOEA/D by using information from its neighboring subproblems [46][47].

MOEA/D requires a decomposition technique for converting the approximation of the PF of a MOP into a number of single-objective optimization problems. In principle, any decomposition technique can serve for this purpose. Three different decomposition techniques have been used widely. In this software, we use the Tchebycheff technique [48]. A single-objective optimization subproblem in this technique is

$$\begin{cases} \text{minimize} \quad g(\vec{x}|\vec{\lambda},\vec{z}^*) = \max\limits_{1\le i\le m}\{\lambda_i|f_i(\vec{x}) - z_i^*|\} \\ \text{s.t.} \quad \vec{x}\in\Omega \end{cases} \tag{4.18}$$

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

It is well known that, under mild conditions, for each Pareto optimal solution there exists a weight vector $\vec{\lambda}$ such that each optimal solution of Eq. (4.18) is a Pareto optimal solution of Eq. (4.17). Let $\vec{\lambda}_1, \dots, \vec{\lambda}_N$ be a set of weight vectors. Correspondingly, we have N single-objective optimization subproblems. If N is reasonably large and $\vec{\lambda}_1, \dots, \vec{\lambda}_N$ are properly selected, then the optimal solutions to these subproblems will provide a good approximation to the PF or PS of Eq. (4.17).

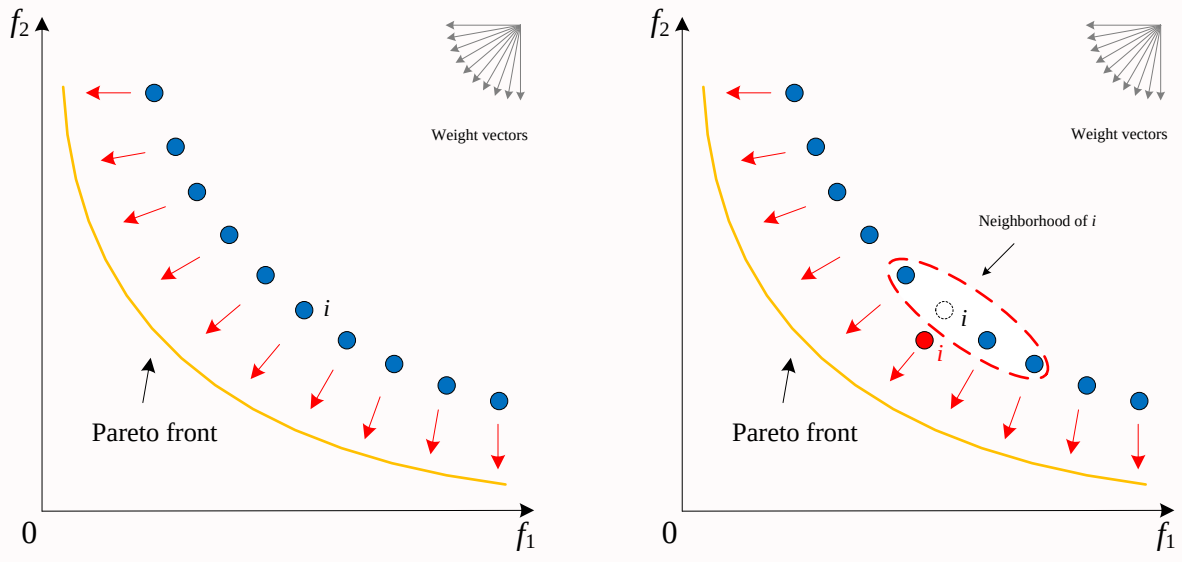


Figure 4.13 A simple graphical representation of MOEA/D

MOEA/D attempts to optimize these single-objective optimization subproblems simultaneously instead of solving a MOP directly. In MOEA/D, the T closest weight vectors in $\{\vec{\lambda}_1, \dots, \vec{\lambda}_N\}$ to a weight vector $\vec{\lambda}_i$ form the neighborhood of $\vec{\lambda}_i$. The neighborhood of the ith subproblem consists of all the subproblems with the weight vectors from the neighborhood of $\vec{\lambda}_i$. Since $g(\vec{x}|\vec{\lambda}, \vec{z}^*)$ is continuous of $\vec{\lambda}$, the optimal solutions of neighboring subproblems should be close in the decision space. MOEA/D exploits the neighborhood relationship among subproblems to make its search effectively and efficiently. A simple graphical representation of MOEA/D is depicted in Figure 4.13.

As a discrete optimization problem, a feature selection problem cannot be solved by MOEA/D directly. To address this issue, we propose a new implementation of MOEA/D for dealing with discrete MOPs. The new implementation is called MOEA/D-BFDE which uses a binary feedback differential evolution (please refer to Section 4.2.1.2) for producing new solutions. The following is the pseudocode of MOEA/D-BFDE.

Algorithm: MOEA/D-BFDE

Input: N: the population size (the number of subproblems);
δ: the probability that parents are selected from the neighborhood;
n_r: the maximal number of solutions replaced by each offspring;

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

2: Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For $i = 1, \dots, N$, set $B(i) = \{i_1, \dots, i_T\}$ where $i_1, \dots, i_T$ are the indexes of the T closest weight vectors (i.e., $\vec{\lambda}_{i_1}, \dots, \vec{\lambda}_{i_T}$) to $\vec{\lambda}_i$;

3: Randomly generate an initial population P_G: $\vec{x}_{1,G}, \dots, \vec{x}_{N,G}$;

4: Compute the corresponding binary-digits population Pb_G: $\overrightarrow{xb}_{1,G}, \dots, \overrightarrow{xb}_{N,G}$ via Eq. 4.6 and Eq. 4.7, evaluate each individual in Pb_G, and use $\vec{F}(\overrightarrow{xb}_{i,G}) = \left(f_1(\overrightarrow{xb}_{i,G}), \dots, f_m(\overrightarrow{xb}_{i,G})\right)$ as the fitness value of $\overrightarrow{xb}_{i,G}$. /* $\vec{F}(\overrightarrow{xb}_{i,G})$ also represents the fitness of $\vec{x}_{i,G}$ */

5: Initialize $\vec{z} = (z_1, \dots, z_m)$ by setting $z_j = \min_{1 \le i \le N} f_j(\overrightarrow{xb}_{i,G})$;

6: $P_{G+1} = \emptyset$ and $Pb_{G+1} = \emptyset$;

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

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

8: Set $r_1 = i$ and randomly select two indexes r_2 and r_3 from R. Generate a mutant vector $\vec{v}_{i,G}$ by using the DE/rand/1 mutation operator, i.e., $\vec{v}_{i,G} = \vec{x}_{r_1,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G})$, and then incorporate binary-digits information into $\vec{v}_{i,G}$: $\vec{v}_{i,G} = \vec{v}_{i,G} - (1 - \overrightarrow{xb}_{i,G})$; /*Reproduction*/

9: If $\vec{v}_{i,G}$ is not feasible (i.e., not in the search space), repair $\vec{v}_{i,G}$ via Eq. 4.9; /*Repair*/

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; /*Crossover*/

11: Compute the corresponding binary-digits individual $\overrightarrow{ub}_{i,G}$ of $\vec{u}_{i,G}$ via Eq. 4.6 and Eq. 4.7, and evaluate its fitness, i.e., $\vec{F}(\overrightarrow{ub}_{i,G})$

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

13: Set $c = 0$;

14: **While** $c \leq n_r$ & R is not empty

15: **If** $\mathrm{g}(\overrightarrow{ub}_{i,G} | \vec{\lambda}_j, \vec{z}) \leq \mathrm{g}(\overrightarrow{xb}_{j,G} | \vec{\lambda}_j, \vec{z})$

16: Set $\vec{x}_{j,G} = \vec{u}_{j,G}$, $\overrightarrow{xb}_{j,G} = \overrightarrow{ub}_{j,G}$, and $\vec{F}(\overrightarrow{xb}_{j,G}) = \vec{F}(\overrightarrow{ub}_{j,G})$;

17: $c = c + 1$;

18: Remove j from R;

19: **End if**

20: **End While**

21: **End For**

22: $G = G + 1$;

23: **Stopping Criterion:** If the maximum number of fitness evaluations is reached, then stop and output Pb_G and $\{\vec{F}(\overrightarrow{xb}_{1,G}), \dots, \vec{F}(\overrightarrow{xb}_{N,G})\}$; otherwise go to step 3.

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

For a feature selection problem, the representation used here is the same as described in Section 4.2.1.3, which is an n-bit string and n is the total number of features in the dataset. In the following, we will introduce the non-dominated sorting genetic algorithm-II (NSGA-II) [49], which is a representative of the Pareto domination-based approaches. NSGA-II includes the following three important components.

1) Fast-non-dominated-sort: In NSGA-II, for each solution p we calculate two entities: 1) domination count n_p, the number of solutions which dominate p, and 2) S_p, a set of solutions which are dominated by p. Firstly, all solutions, the domination count of which is equal to zero, are put into the first non-dominated front. Next, for each solution p with $n_p = 0$, we visit each member (q) in its set S_p and reduce the domination count of q by one (i.e., $n_q = n_q - 1$). Afterward, for any member q, if its domination count becomes zero, we then put it into another list r. These members belong to the second non-dominated front. Subsequently, the above process is implemented on each member in set S_r and the third non-dominated front is identified. This procedure continues until all non-dominated fronts are identified. Figure 4.14 shows the identification of different non-dominated fronts.

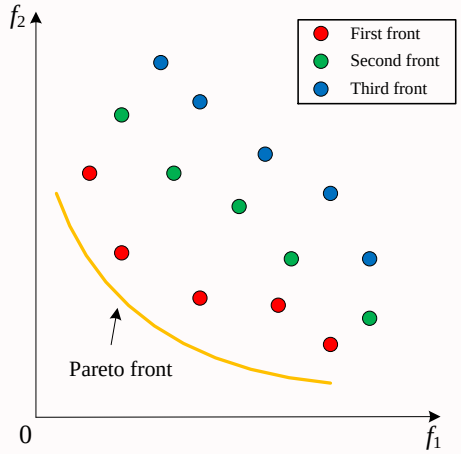


Figure 4.14 Non-dominated sorting procedure

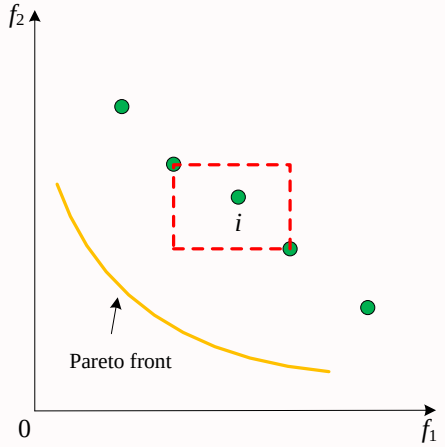


Figure 4.15 Crowding-distance calculation

2) Crowding-distance-assignment: Along with the evolution toward the PF, an EA should maintain a good spread of solutions. In order to accomplish this goal, it is necessary to estimate the density of a solution. To this end, the perimeter of the rectangle, which is formed by the two nearest solutions on either side of a specific solution along each of the objectives in its non-dominated front, serves as an estimate of the density (also called the crowding distance). In Figure 4.15, the crowding distance of the ith solution in its non-dominated front is the perimeter of the rectangle (shown with a dashed box).
3) Make-new-population: Through a binary tournament selection, crossover, and mutation, the

offspring population Q_t is created from the parent population P_t. In the binary tournament selection, two solutions are randomly chosen and compared based on the crowded-comparison operator $(\prec_n)$. Assume that each individual i in the population has two attributes: a) Non-dominated rank (i_{rank}); and b) Crowding distance $(i_{distance})$. Next, a partial order $\prec_n$ is defined as follows:

$$i \prec_n j \quad \text{if } (i_{rank} < j_{rank}) \text{ or } \left((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})\right) \tag{4.19}$$

That is, between two solutions with different non-dominated fronts, we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same non-dominated front, then we prefer the solution with a larger crowding distance.

A brief description of NSGA-II is presented in Figure 4.16. Suppose that the parent population at the tth generation is P_t and its population size is N. The offspring population created from P_t is denoted as Q_t, which also have N members. The first step is to choose the best N members from the combined parent and offspring population $R_t = P_t \cup Q_t$ (of size $2N$), with the aim of preserving elite members of the previous evolution. To achieve this, firstly the combined population R_t is split into different non-dominated fronts (F_1, F_2, and so on). Then, each non-dominated front is selected one by one to construct a new population S_t, until the size of S_t is equal to N or for the first time exceeds N. Suppose that the last accepted non-dominated front is F_l. Thus, all solutions from F_{l+1} onward are rejected from the combined population R_t. In most situations, F_l is only accepted partially. Under this condition, only several solutions with the maximum crowding distances in F_l are chosen. [50][51].

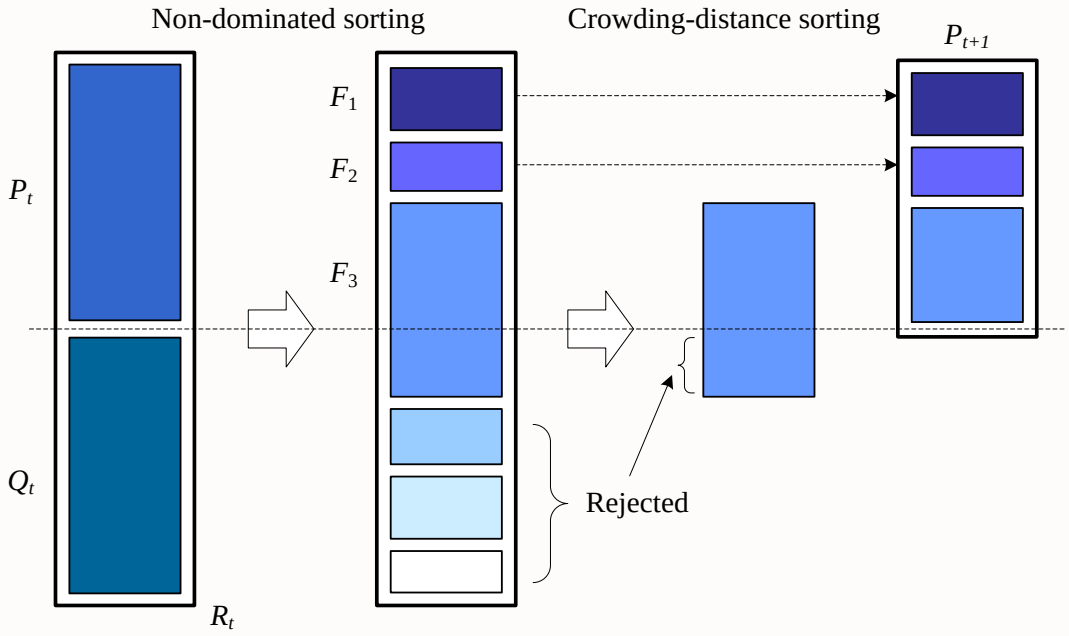


Figure 4.16 NSGA-II procedure

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) // $F = \{F_1, F_2, \ldots\}$
3: $P_{t+1} = \emptyset$ and $i = 1$
4: While $\lvert P_{t+1}\rvert + \lvert F_i\rvert \leq N$ // $\lvert P_{t+1}\rvert$ and $\lvert F_i\rvert$ denote the number of individuals in P_{t+1} and F_i, respectively.
5: Crowding-distance-assignment(F_i)
6: $P_{t+1} = P_{t+1} \cup F_i$
7: $i = i + 1$
8: End While
9: Crowding-distance-assignment(F_i)
10: Sort$(F_i, \prec_n)$
11: Put $(N - \lvert P_{t+1}\rvert)$ individuals with the maximum crowding distances of F_i into P_{t+1}
12: Q_{t+1} = Make-new-pop(P_{t+1})
13: $t = t + 1$

4.2.3 Parallel Execution

With the rapid development of information technologies and the emergence of “big data”, feature selection problems gradually become more and more data-intensive. Therefore, parallel execution is a useful strategy which makes full use of the processing power of multicore desktops, and leads to a significant reduction in the processing time. In this software, the parallel execution is applied to accelerate EAs [52][53].

4.3 Subset Evaluation: Models

Models are developed using one or more statistical modeling tools, which may be broadly categorized into regression- and classification-based models. Figure 4.17 shows the overview of the Subset Evaluation: Models.

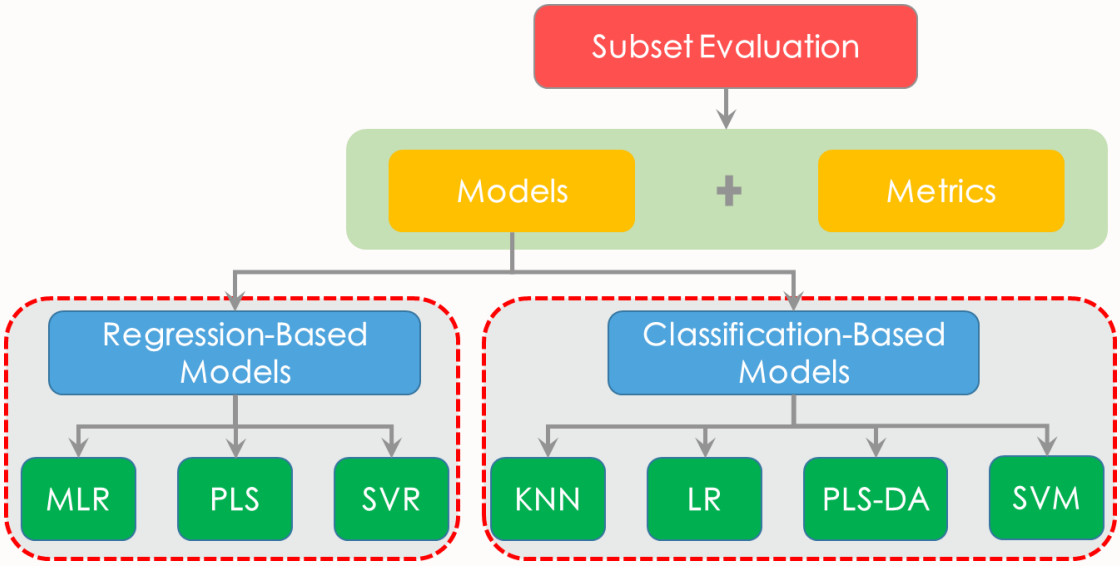


Figure 4.17 The overview of the Subset Evaluation: Models

4.3.1 Regression-Based Models

Regression-based models are used when the response variable is quantitative. The generated regression model can compute quantitative response data from the model. 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	$(Y)_m$	$(X_1)_m$	$(X_2)_m$	$(X_3)_m$	...	$(X_n)_m$

4.3.1.1 Multiple linear regression (MLR)

Multiple linear regression (MLR) [63] is one of the most popular methods due to its simplicity in

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

$$Y = a_0 + a_1 \times X_1 + a_2 \times X_2 + a_3 \times X_3 + \cdots + a_n \times X_n \quad (4.20)$$

In Eq. (4.20), 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.

4.3.1.2 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) [64] 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 the 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.

4.3.1.3 Support vector regression (SVR)

Support vector machine (SVM) is a popular machine learning tool for classification and regression, first proposed by Vladimir Vapnik and his colleagues in 1992 [65]. A detailed description of the theory of SVM can be referred to several excellent books and tutorials [66][67]. 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. In support vector regression (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 function, polynomial function, 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, 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.

4.3.2 Classification-Based Models

Classification-based models are used when the response variable is Label (like positive-negative). 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	$(Y)_m$	$(X_1)_m$	$(X_2)_m$	$(X_3)_m$	…	$(X_n)_m$

4.3.2.1 K-nearest Neighbor (KNN)

The K-nearest Neighbor (KNN) [68] 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 [69]. 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 [69]. 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 [70].

4.3.2.2 Logistic Regression (LR)

Logistic Regression (LR) [71] 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

$$\text{logit}[p(x)] = log\left[\frac{p(x)}{1-p(x)}\right] = a + b_1x_1 + b_2x_2 + b_3x_3 + \cdots \quad (4.21)$$

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. (4.21), 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.

4.3.2.3 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 [72]. 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 matrix Y in the usual way [73]. 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 value 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 [74][75]. 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.

4.3.2.4 Support Vector Machine (SVM)

Support vector machines (SVMs) are a kind of 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 [76]. 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 [76].

SVMs were primarily designed for binary classification. Different methods have been developed to use SVMs for multiple (C) class classification [77]. A common way is to build C "one-versus-rest" classifiers (commonly referred to as "one-versus-all" classification) [78], 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 [79], 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 [80]. 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 [81].

4.4 Subset Evaluation: Metrics

Feature subsets produced by EAs (SOEAs or MOEAs) 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 4.18 shows the overview of the Subset Evaluation: Metrics.

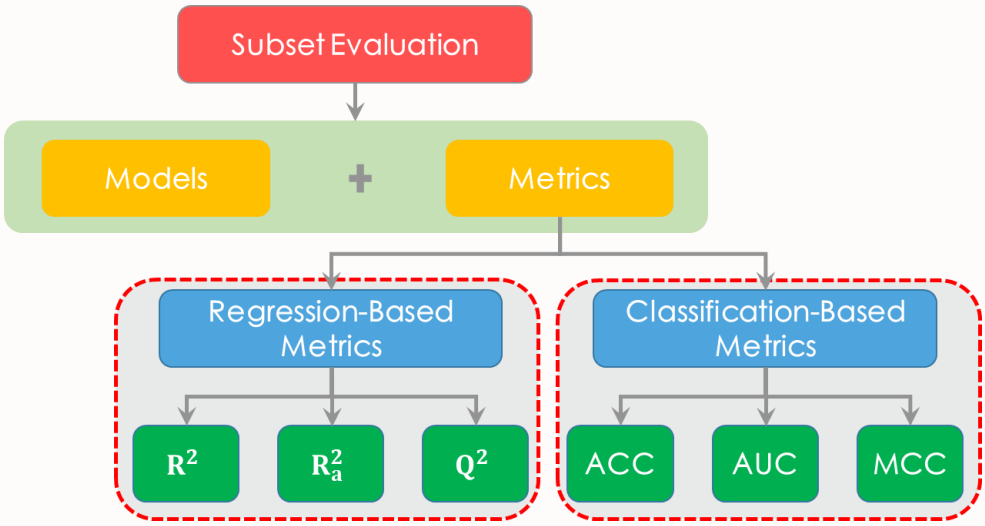


Figure 4.18 The overview of the Subset Evaluation: Metrics

4.4.1 Metrics for Regression-Based Models

4.4.1.1 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. For a good model, the calculated Y value (Y_{calc}) for a particular observation from the model should be exactly the same as the corresponding observed Y value (Y_{obs}), 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, $\frac{\sum(Y_{obs}-Y_{calc})^2}{\sum(Y_{obs}-\bar{Y}_{obs})^2}$ should have a low value for a good model. We can define the determination coefficient (R^2) in the following manner:

$$R^2 = 1 - \frac{\sum(Y_{obs}-Y_{calc})^2}{\sum(Y_{obs}-\bar{Y}_{obs})^2} \tag{4.22}$$

For the ideal model, the sum of squared residuals is 0 and 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).

4.4.1.2 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 R_a^2, which is defined in the following manner:

$$R_a^2 = \frac{(m-1)\times R^2 - n}{m-1-n} \tag{4.23}$$

In Eq. (4.23), 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 (i.e., R_a^2) value is penalized due to an increase in the number of predictor variables.

4.4.1.3 Cross-validated Q^2

The cross-validation technique mainly involves internal validation [55][56], 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. 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 4.19.

To achieve 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 [57] for the model. PRESS is the

sum of squared differences between experimental and LOO predicted data. Eq. (4.24) and Eq. (4.25) give the expressions for PRESS and Q^2, respectively:

$$\text{PRESS} = \sum\left(Y_{obs} - Y_{pred}\right)^2 \tag{4.24}$$

$$Q^2 = 1 - \frac{\sum\left(Y_{obs(train)} - Y_{pred(train)}\right)^2}{\sum\left(Y_{obs(train)} - \bar{Y}_{train}\right)^2} = 1 - \frac{\text{PRESS}}{\sum\left(Y_{obs(train)} - \bar{Y}_{train}\right)^2} \tag{4.25}$$

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

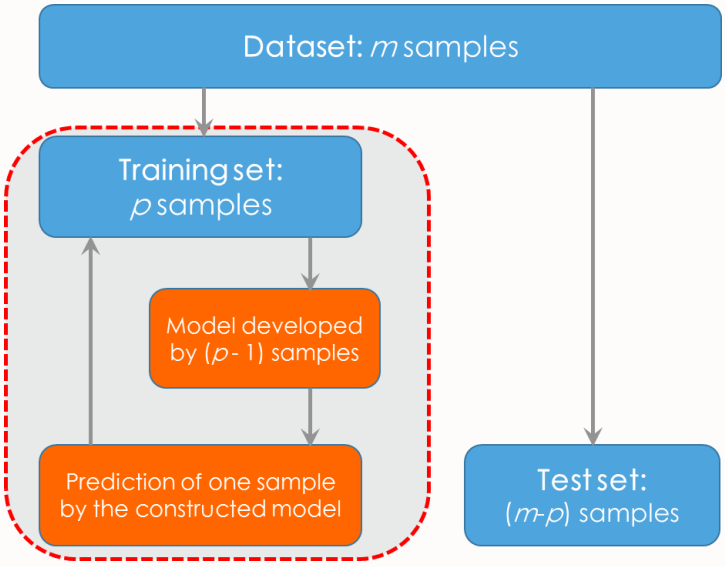


Figure 4.19 Key steps for the calculation of LOO-CV

Note: In *n*-fold cross-validation, a sample of *n* observations is randomly partitioned into *n*-folds (partitions) and the folds are with 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) [54]. 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.

4.4.2 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 [58]. The validation for classification model is usually executed for two-class problems, where the samples are categorized either as positive or as negative.

4.4.2.1 Sensitivity, specificity and accuracy

The samples classified by 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 [59]. Based on this classification and the number of test-set samples, a two-by-two confusion matrix [60], also referred to as the contingency table (confusion matrix), may be constructed that corresponds to the dispositions of the samples under consideration. The confusion matrix 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:

$$\text{Sensitivity} = \text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{4.26}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} \tag{4.27}$$

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \tag{4.28}$$

where 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

4.4.2.2 Matthews correlation coefficient

The Matthews correlation coefficient (MCC) [61] is utilized as a measure of the quality of binary classifications. It is generally regarded as a balanced measure that can be used even if the classes have different sizes. 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. MCC can be calculated directly from the confusion matrix using the following formula:

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}} \tag{4.29}$$

If any of the four sums in the denominator is zero, the denominator can be arbitrarily set to 1. This metric in particular addresses the issue of improper explanation of a confusion matrix and the cases where the data set sizes are higher.

4.4.2.3 AUC-ROC

Area under curve-receiver operating characteristics (AUC-ROC) is equivalent to a simple average of the ranks of the positive samples [62]. Let m be the number of positive samples and M be the total number of samples; 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. (4.30), is linearly related to the rank sum of positive samples, which is also called the Mann-Whitney U test.

$$\text{AUC-ROC} = 1 - \frac{\sum_{i=1}^{m} r_i}{m\times(M-m)} + \frac{m+1}{2\times(M-m)} \tag{4.30}$$

where r_i is the rank of the ith positive sample.

4.5 Performance Analysis and Comparison

In ECoFFeS, we present two novel algorithms (modified DE and modified MOEA/D, called BFDE and MOEA/D-BFDE, respectively) to choose the optimal feature subset. In order to verify the effectiveness of them, some experiments were conducted in this section.

4.5.1 Benchmark Datasets

In this study, three quantitative structure-activity/property relationship datasets are used. The first is Artemisinin dataset, which consists of 211 artemisinin analogues [82]. 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 has 178 compounds. As pointed out in [83], 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 [84], 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 [85]. The BZR dataset used in our study is presented in [86]. 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 [87], which has become a benchmark to evaluate the performance of different methods and has been well-studied in QSAR/QSPR [88]. 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 above three datasets, together with calculated molecular descriptors [89], can 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 has a main characteristic, namely a feedback strategy (please refer to Section 4.2.1.2). In order to assess the effectiveness of BFDE, four different methods are employed for comparison on the three datasets. These methods include: (a) PLS [64]; (b) WS-PSO-PLS [89]; (c) BDE-PLS [29], which is an algorithm without a feedback strategy; and (d) BFDE-PLS. Among them, the PLS regression is applied to calculate Q^2.

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
BDE-PLS	150	200	30000	30
BFDE-PLS	150	200	30000	30

In order to compare the performance of the involved methods, three performance metrics have been chosen: Q^2, the root mean square error from five-fold cross-validation (denoted as *RMSECV*) and the number of the selected descriptors (denoted as *N*). Here, the reason why we choose five-fold cross validation is the following. 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. Note that all the involved methods have been implemented 30 times to obtain the statistical results. All the data were firstly auto-scaled to have zero mean and unit variance before modeling. The parameter settings for all the involved methods are listed in Table 4.3 and the experimental results are presented below.

(1) The necessity of descriptor selection

In Table 4.4, 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 BFDE-PLS decreases drastically, the mean Q^2 is 0.7594, 0.5863, and 0.9206 and the mean *RMSECV* is 0.7690, 0.7063, and

0.2087 for the Artemisinin, BZR, and Selwood datasets, respectively. The aforementioned results suggest that BFDE-PLS with less number of descriptors is significantly better than PLS, which verifies the necessity to perform descriptor selection before the QSAR model development.

Table 4.4 Experimental results of PLS and BDE-PLS 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
	BFDE-PLS	**0.7594 ± 0.0072**	**0.7690 ± 0.0115**	**23.6000 ± 2.9196**
BZR	PLS	0.4007	0.8501	75
	BFDE-PLS	**0.5863 ± 0.0087**	**0.7063 ± 0.0074**	**21.5000 ± 2.3599**
Selwood	PLS	0.2407	0.6461	53
	BFDE-PLS	**0.9206 ± 0.0067**	**0.2087 ± 0.0087**	**12.0000 ± 1.4384**

(2) The effectiveness of the feedback strategy

In Table 4.5, 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 is formed by combining BDE-PLS with the proposed strategy (a feedback strategy). Therefore, the feedback strategy can be adopted to enhance the performance of BDE-PLS based on the experimental results.

Table 4.5 Experimental results of BDE-PLS and BFDE-PLS on the three datasets

Datasets	Methods	Mean Q^2 ± Standard deviation	Mean *RMSECV* ± Standard deviation	Mean *N* ± Standard deviation
Artemisinin	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	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	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**

In addition, Figure 4.20 provides the boxplots of Q^2 for BDE-PLS and BFDE-PLS on the Artemisinin, BZR and Selwood datasets, respectively.

Note: 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 the red "+" represents outliers.

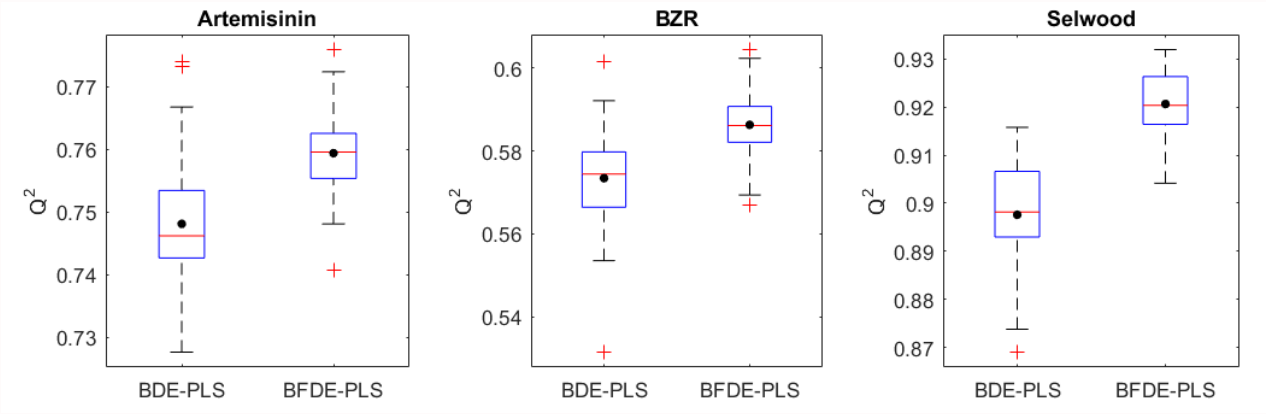


Figure 4.20 Boxplots of Q^2 for BDE-PLS and BFDE-PLS on the three datasets

(3) The superiority performance of BFDE-PLS

As shown in Table 4.6, BFDE-PLS is better than WS-PSO-PLS in term of the mean Q^2, the mean *RMSECV* and the mean *N* on all datasets, except that the mean Q^2 and the mean *RMSECV* of BFDE-PLS are slightly worse than that of WS-PSO-PLS on the Selwood dataset. As a result, we can conclude that, overall, BFDE-PLS performs better.

Table 4.6 Experimental results of WS-PSO-PLS and BFDE-PLS on the three datasets

Datasets	Methods	Mean Q^2 ± Standard deviation	Mean *RMSECV* ± Standard deviation	Mean *N* ± Standard deviation
Artemisinin	WS-PSO-PLS	0.7588 ± 0.0090	0.7699 ± 0.0144	30.8333 ± 3.6774
	BFDE-PLS	**0.7594 ± 0.0072**	**0.7690 ± 0.0115**	**23.6000 ± 2.9196**
BZR	WS-PSO-PLS	0.5652 ± 0.0058	0.7241 ± 0.0048	28.3667 ± 2.4138
	BFDE-PLS	**0.5863 ± 0.0087**	**0.7063 ± 0.0074**	**21.5000 ± 2.3599**
Selwood	WS-PSO-PLS	**0.9242 ± 0.0152**	**0.2031 ± 0.0212**	16.2333 ± 3.5202
	BFDE-PLS	0.9206 ± 0.0067	0.2087 ± 0.0087	**12.0000 ± 1.4384**

In addition, Figure 4.21 provides the boxplots of Q^2 for WS-PSO-PLS and BFDE-PLS on the Artemisinin, BZR and Selwood datasets, respectively.

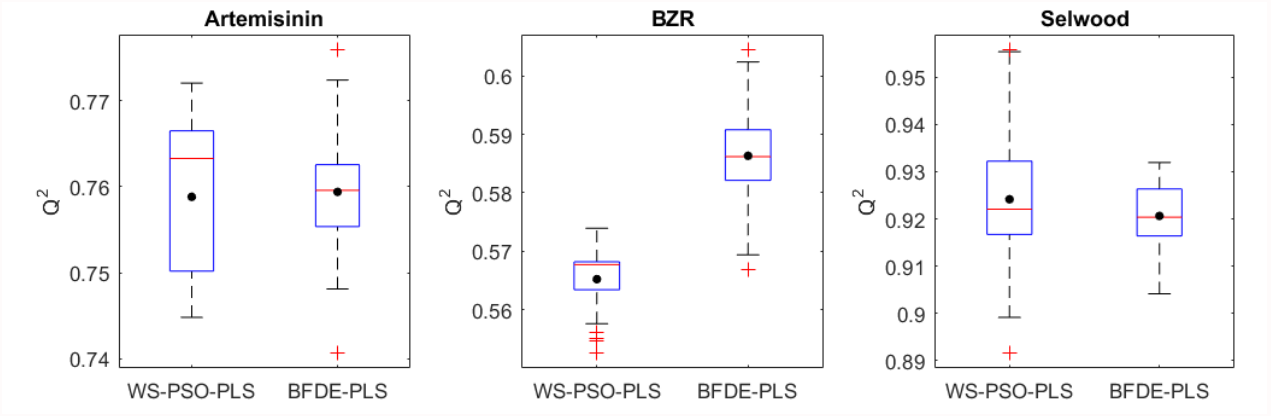


Figure 4.21 Boxplots of Q^2 for WS-PSO-PLS and BFDE-PLS on the three datasets

4.5.2.2 MOEA/D-BFDE

In order to analyze the performance of the proposed MOEA/D-BFDE, three performance metrics have been chosen: Max Q^2, Min *RMSECV* (the minimum root mean square error from five-fold cross-validation) and Max *N* (the maximum number of the selected descriptors). The parameter settings of MOEA/D-BFDE-MLR and NSGA-II-MLR are listed in Table 4.7. Among them, the MLR regression is applied to calculate Q^2.

Table 4.7 The parameter values used in MOEA/D-BFDE-MLR and NSGA-II-MLR

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

According to the experimental results presented in Table 4.8, Table 4.9 and Table 4.10, for the BZR and Selwood datasets, MOEA/D-BFDE-MLR performs better than NSGA-II-MLR, but the performance is slightly worse than NSGA-II-MLR on the Artemisinin dataset.

Table 4.8 Experimental results on Artemisinin dataset

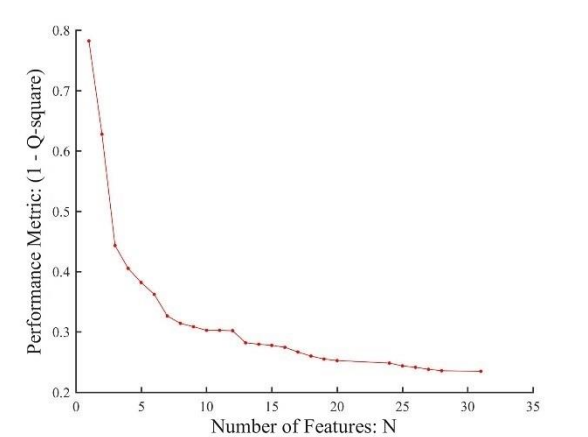
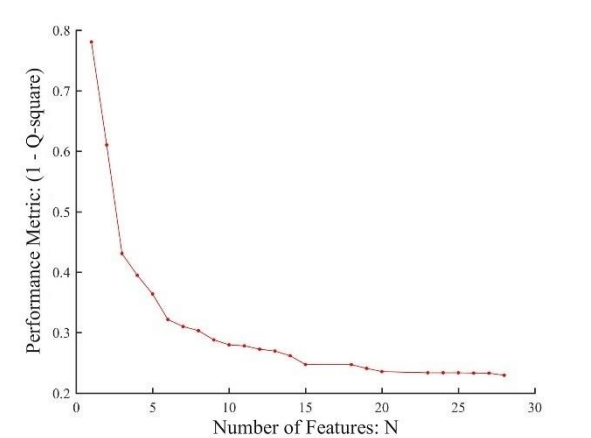
Artemisinin Dataset					
MOEA/D-BFDE-MLR			NSGA-II-MLR		
Max Q^2	Min *RMSECV*	Max *N*	Max Q^2	Min *RMSECV*	Max *N*
0.7652	0.7597	31	**0.7703**	**0.7514**	**28**
					

Table 4.9 Experimental results on BZR dataset

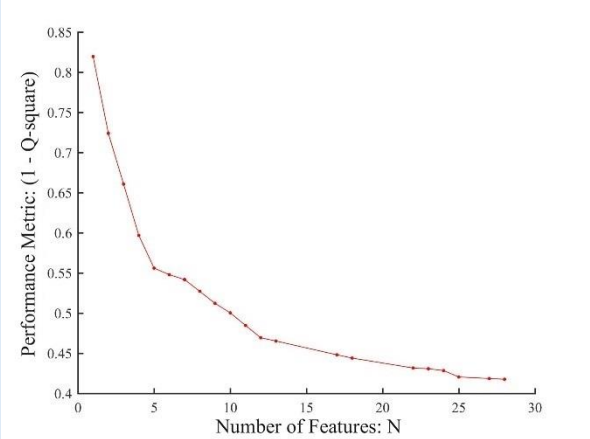
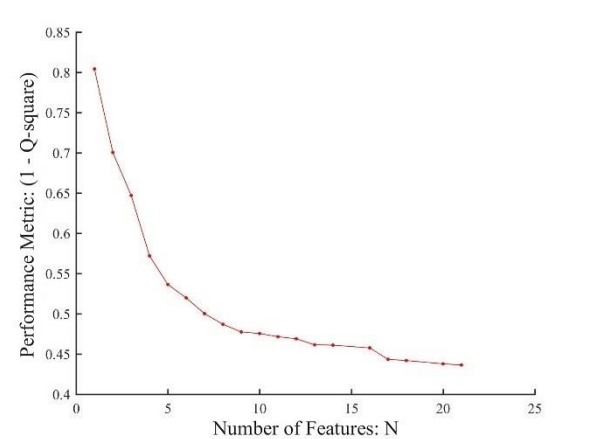
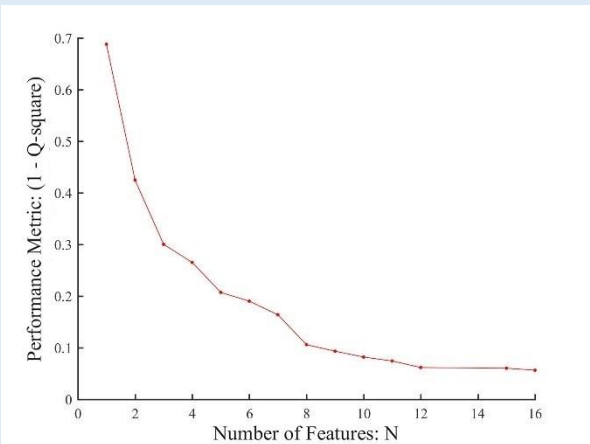
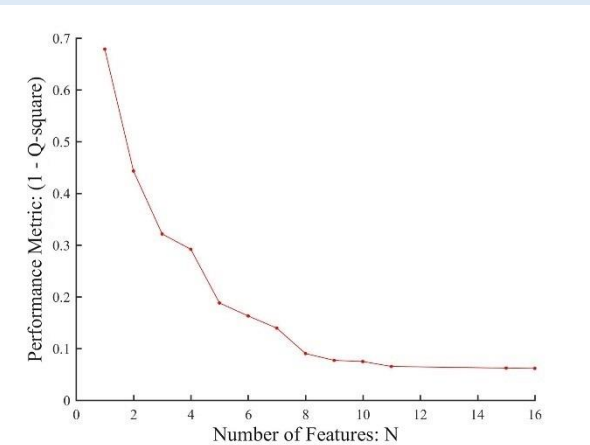
BZR Dataset					
MOEA/D-BFDE-MLR			NSGA-II-MLR		
Max Q^2	Min *RMSECV*	Max *N*	Max Q^2	Min *RMSECV*	Max *N*
0.5824	**0.7097**	28	0.5636	0.7255	21
					

Table 4.10 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 [90][91]. A schematic diagram of drug discovery, development, and clinical assessment is shown in Figure 5.1 [90]. 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.

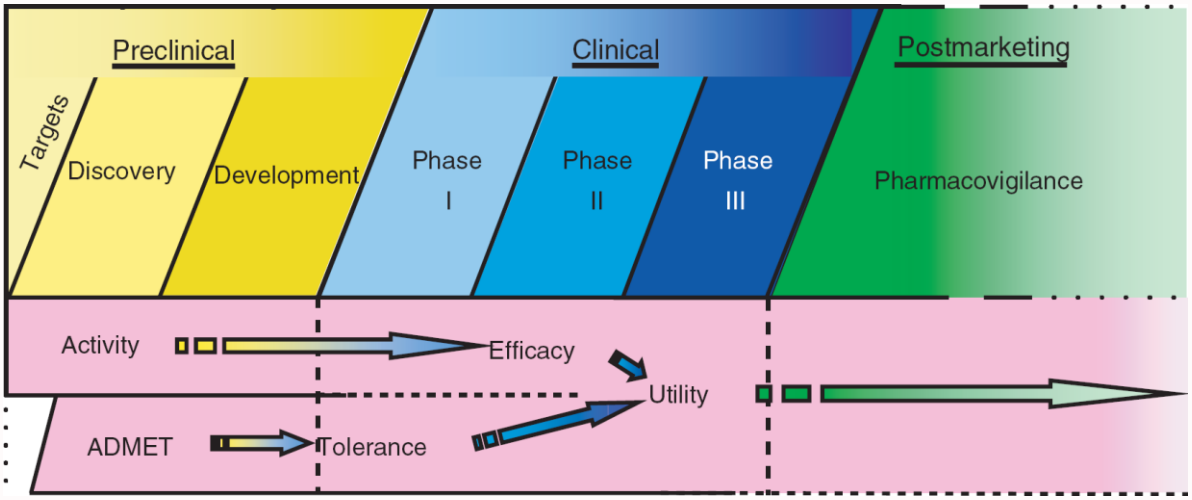


Figure 5.1 A schematic representation of modern drug research and development

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 [92]. 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 [93]. Filtering and optimization of ADMET properties in the early stage of the drug discovery are intensively investigated [94]. However, the experimental evaluation of ADMET profiles is costly, and the work load cannot 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 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 biological 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 [95][96]. 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 [97-99], 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 [100-102]. 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 [99-104]. Therefore, assessment of hERG-related cardiotoxicity has become an important step in the drug design/discovery pipeline [100][105].

Due to 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 QSAR models for hERG liability has 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), and naive Bayesian classification (NBC) [99][106-111]. 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 dataset 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 [99]. 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 [112]. 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 [113][114].

The whole dataset 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 dataset by using the Find Diverse Molecules protocol in Discovery Studio 2.5 (DS 2.5), respectively [115]. It 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 [99][116].

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 [117]. 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

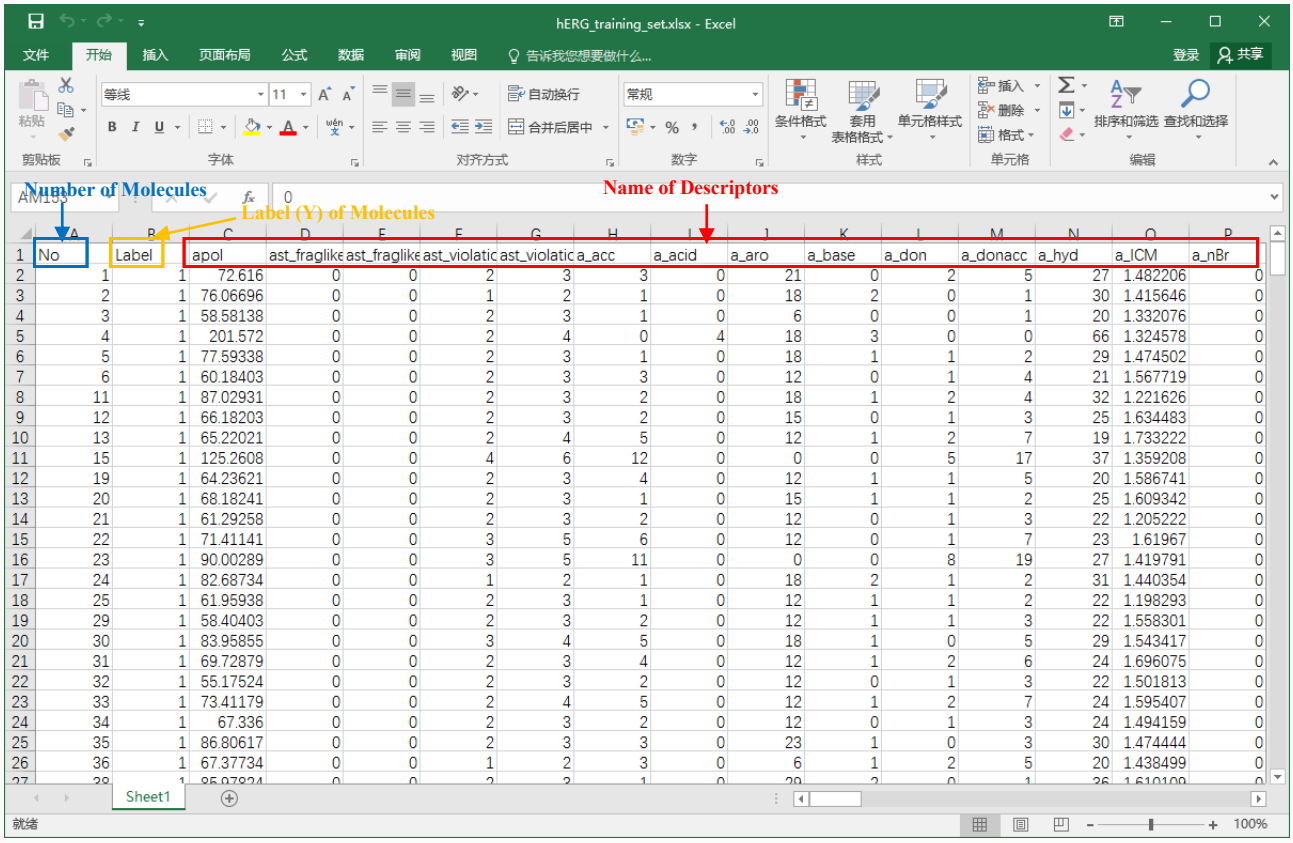
5.2.3.1 ECoFFeS Operation

Step 1 Preparation of Dataset

The pretreated training set: hERG_training_set.xlsx (392 molecules with 131 descriptors)

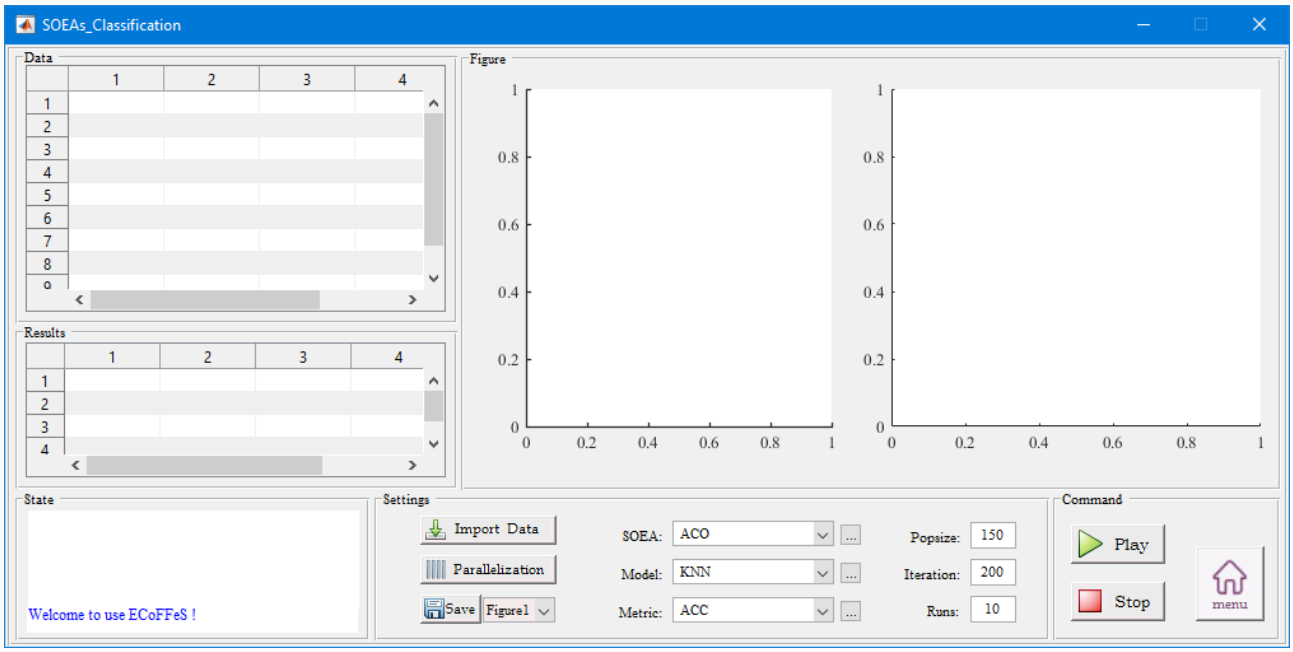
https://github.com/Jiaweihuang/ECoFFeS

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



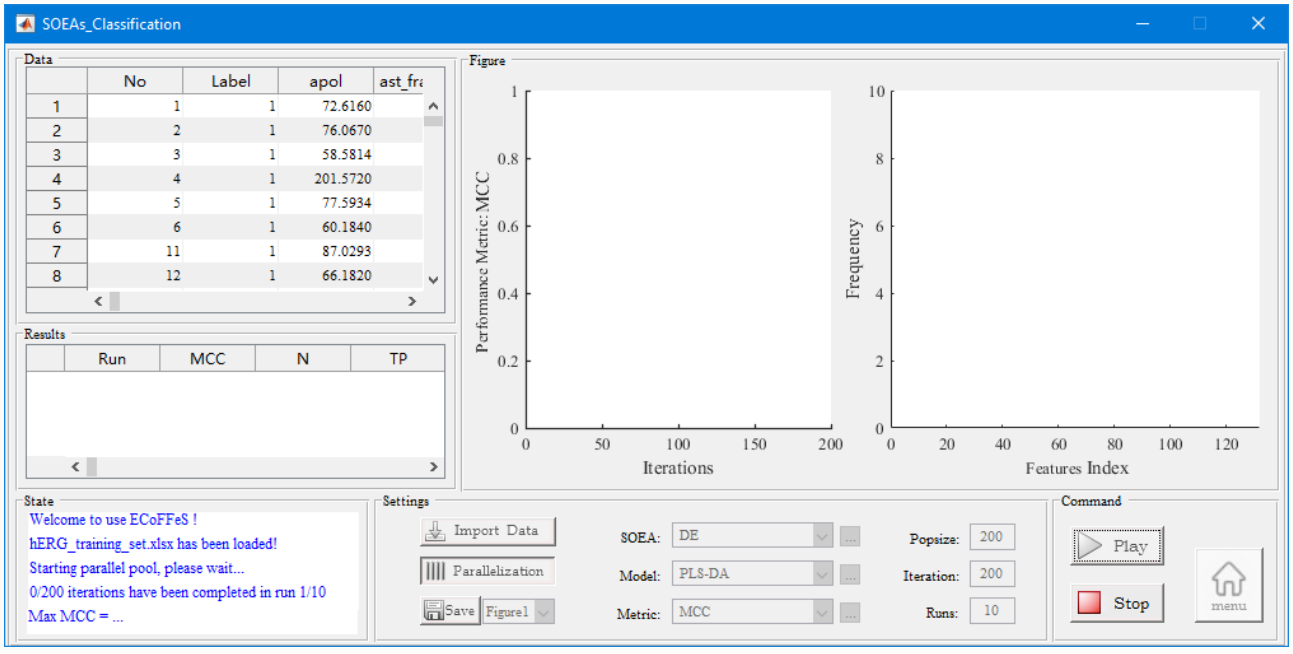
Step 2 Start the SOEAs_Classification

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



Step 3 Set Parameters and Play

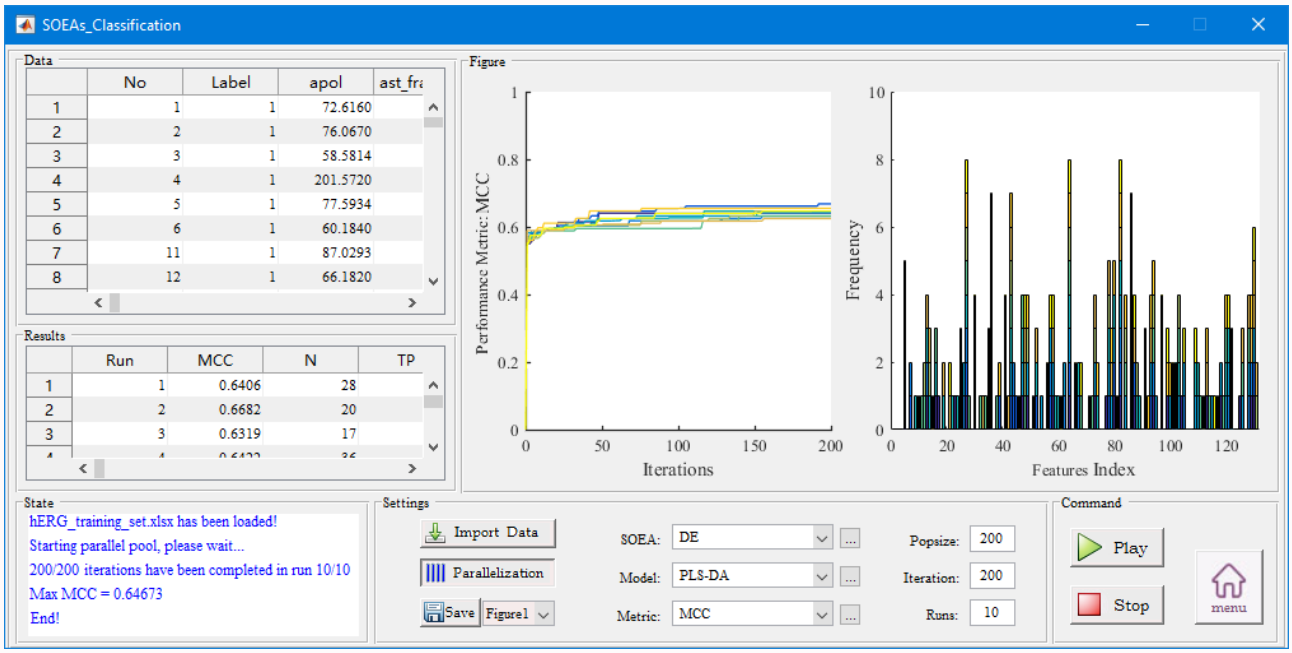
1) Import Data: hERG_training_set.xlsx;
2) Parallelization;
3) SOEA: DE; Model: PLS-DA; Metric: *MCC*;
4) SOEA_parameter: Default parameters;
5) Model_parameter: Default parameters;
6) Metric_parameter: Default parameters;
7) Popsize: 200; Iteration: 200; Runs: 10;
8) Press Play;



Step 4 Save Figures and Results

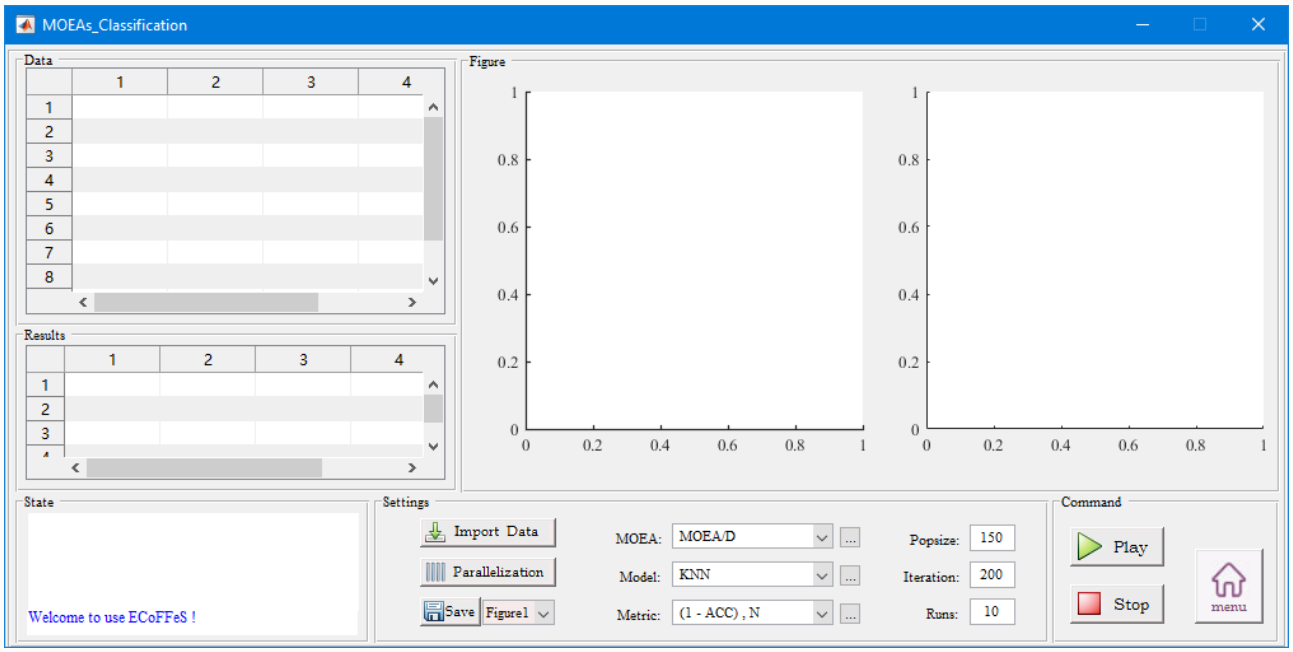
When calculation has been completed, press Save Figure1, Figure2, or Results in the drop-down menu.

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



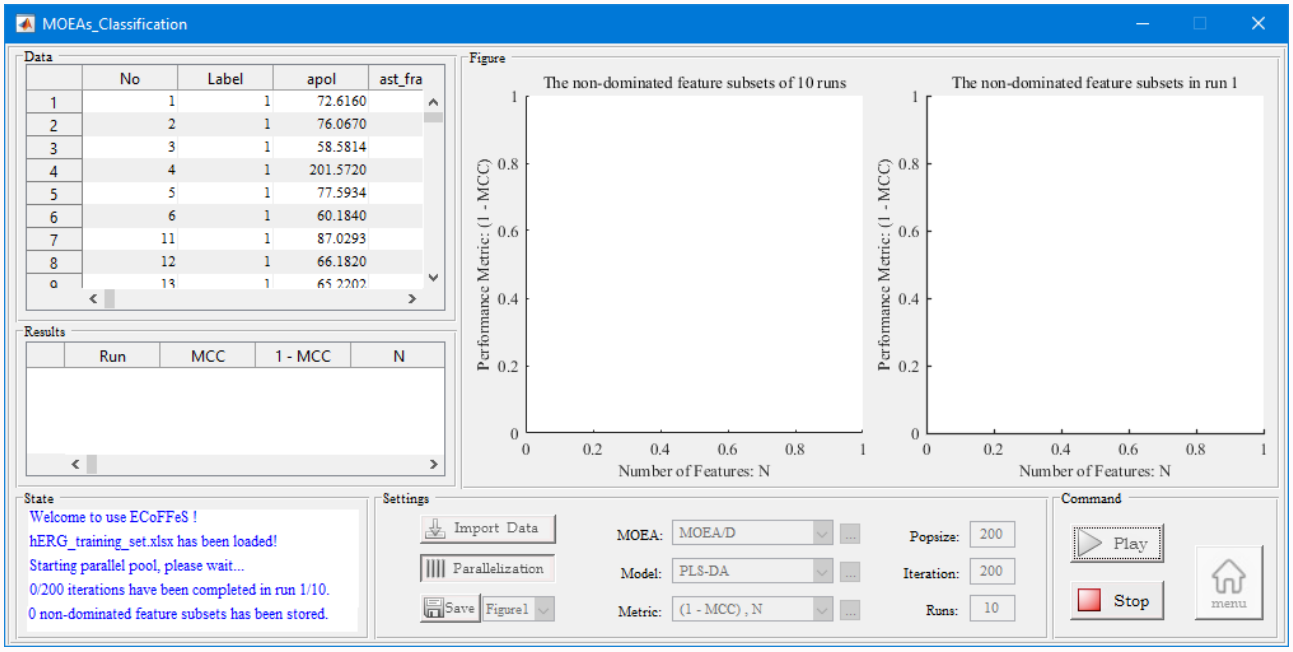
Step 5 Switch to MOEAs_Classification

Press menu to return to the main interface. In main interface, 'Subset Discovery' chooses MOEAs, 'Subset Evaluation' chooses Classification Model & Metric, press 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) Press Play;



Step 7 Save Figures and Results

When calculation has been completed, press Save Figure1, Figure2, or Results in the drop-down menu.

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

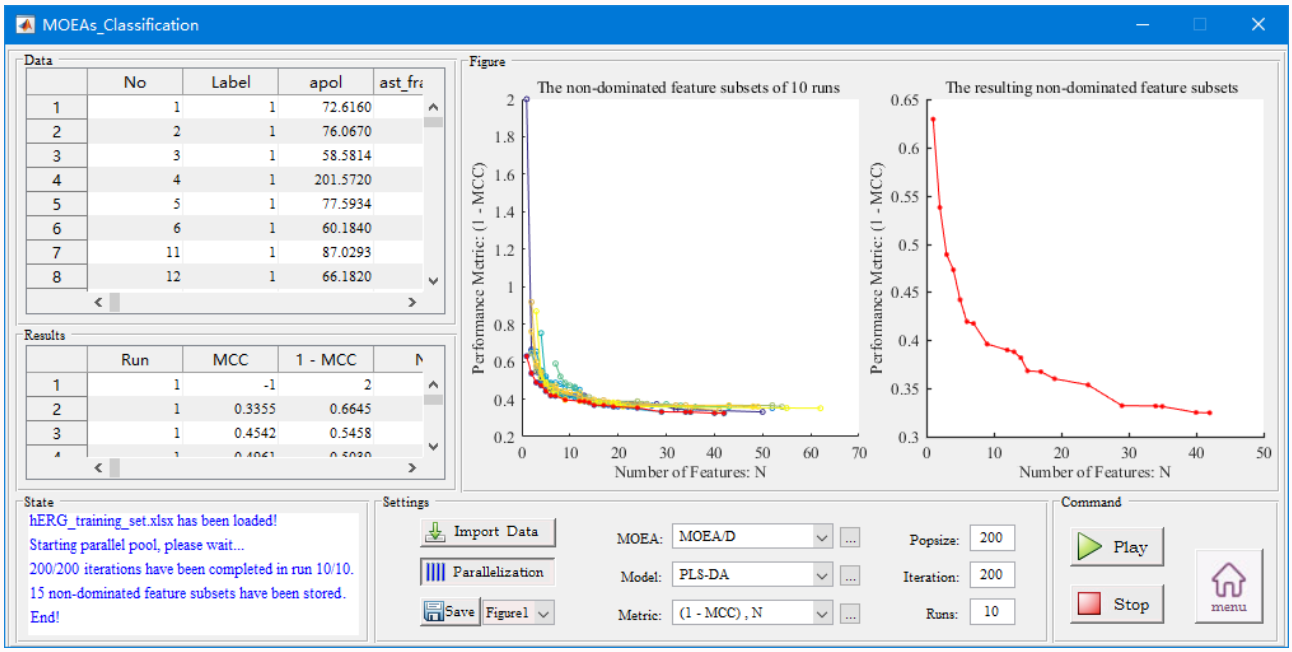


Table 5.1 The Performance of the PLS-DA Classifiers for the Training and Test Sets Based on Different Descriptor Subsets

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.

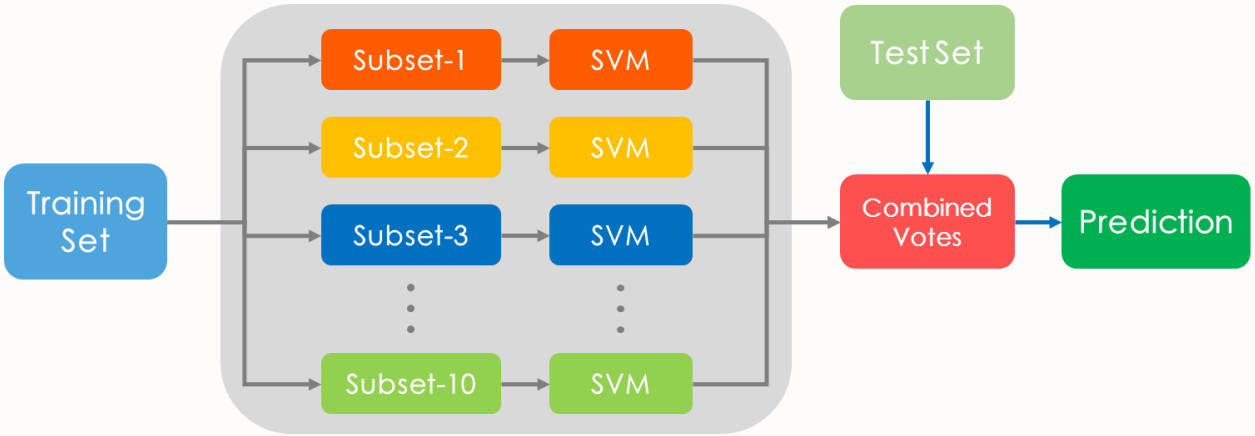


Figure 5.1 The workflow of Ensemble (SVM)

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

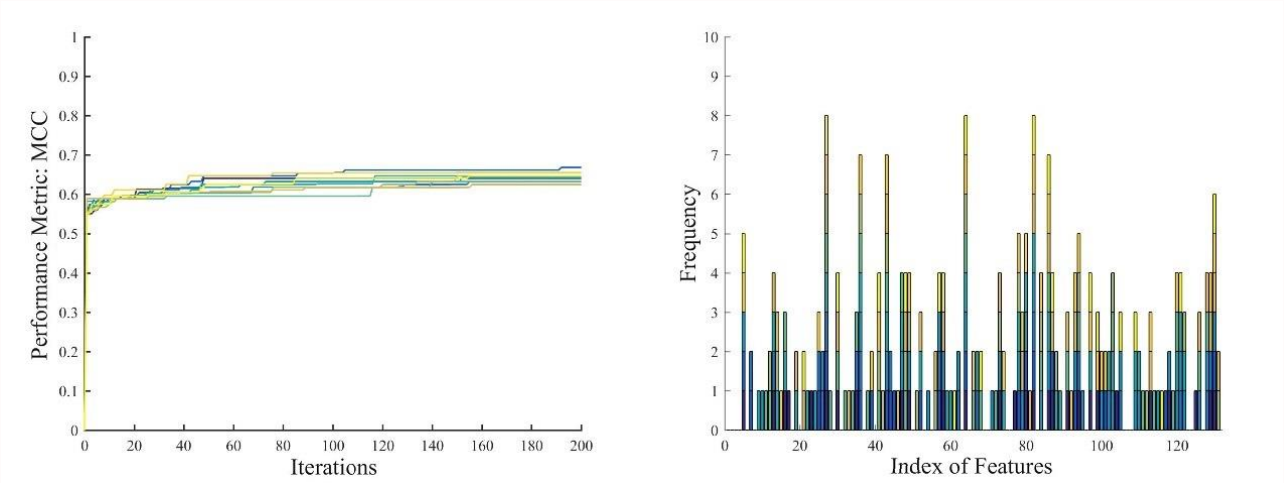


Figure 5.2 The iteration figure and the frequency figure

4) According to the frequency figure, we can acquire the importance of the molecular descriptors and the important descriptors are summarized in Table 5.2. BCUT_SLOGP_0, opr_brigid and PEOE_VSA-2 in the PLS-DA Classifiers have the highest frequency (8/10). Besides, b_max1len, FCharge, PEOE_VSA-6 and vsa_pol are 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 Pareto fronts (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. According to the obtained Pareto optimal solutions (i.e., a set of descriptor subsets), a decision-maker can choose a preferred solution (i.e., a preferred descriptor subset) according to his/her own requirements. Here, we select 29 descriptors in order to keep a good balance between *N* and (1 - *MCC*).

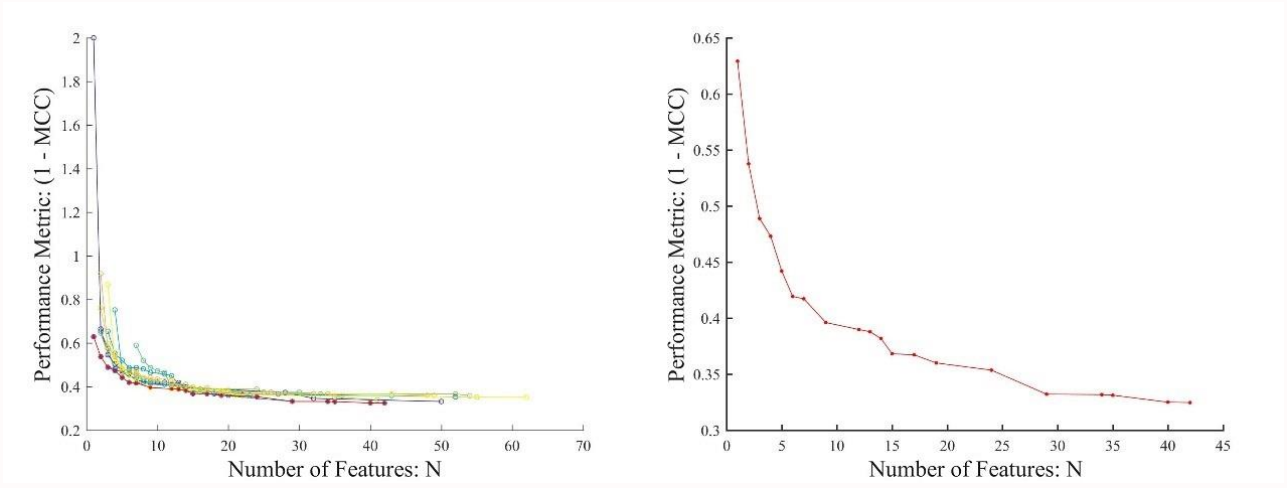


Figure 5.3 The non-dominated descriptor subsets

6) The selected 29 descriptors are 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. In addition, 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 the 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

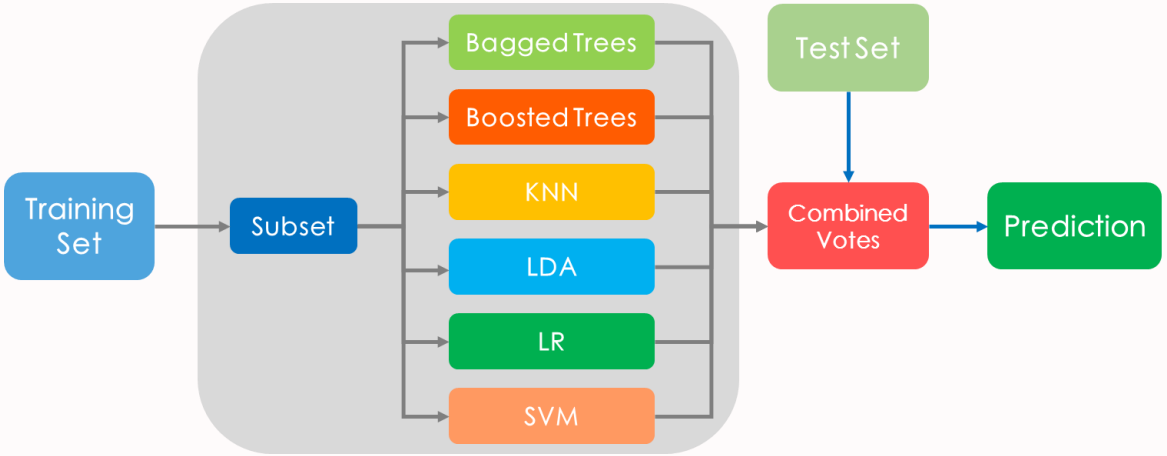


Figure 5.4 Ensemble Model

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 [131].

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 [118][119]. Partition coefficient [120] 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 [120-123] and a crucial part of Lipinski's rule of five [124][125], 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 [126][127]. Thus, the distribution coefficient, which takes account of ionization, may be a more reliable measurement for the lipophilicity at physiological pH [128][129]. 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 [130] 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, and the other 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 [131]. Herein, all molecules were divided into two parts, namely the training set and the test set, using the Kennard–Stone method [132][133] 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 [117]. 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

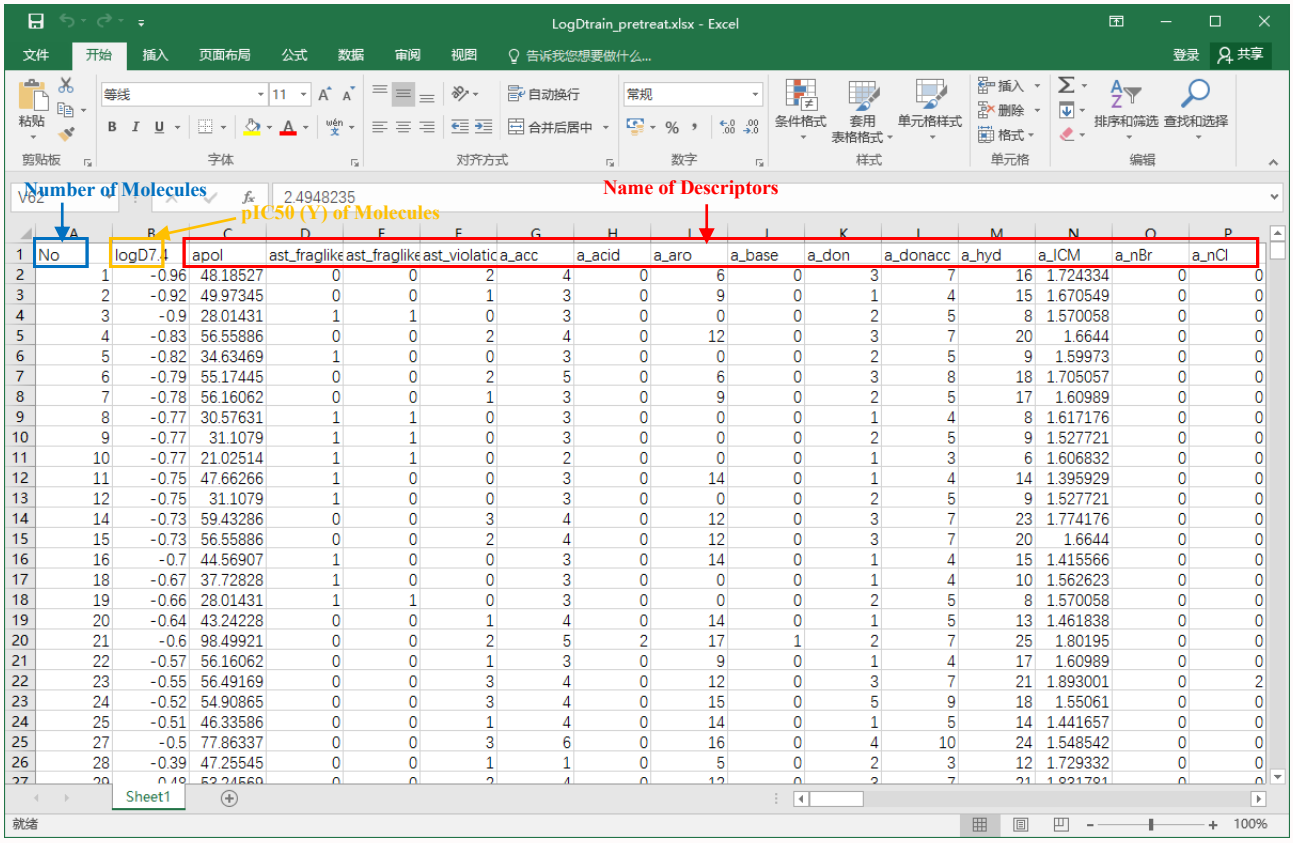
5.3.3.1 ECoFFeS Operation

Step 1 Preparation of Dataset

The pretreated training set: LogDtrain_pretreat.xlsx (904 molecules with 127 descriptors)

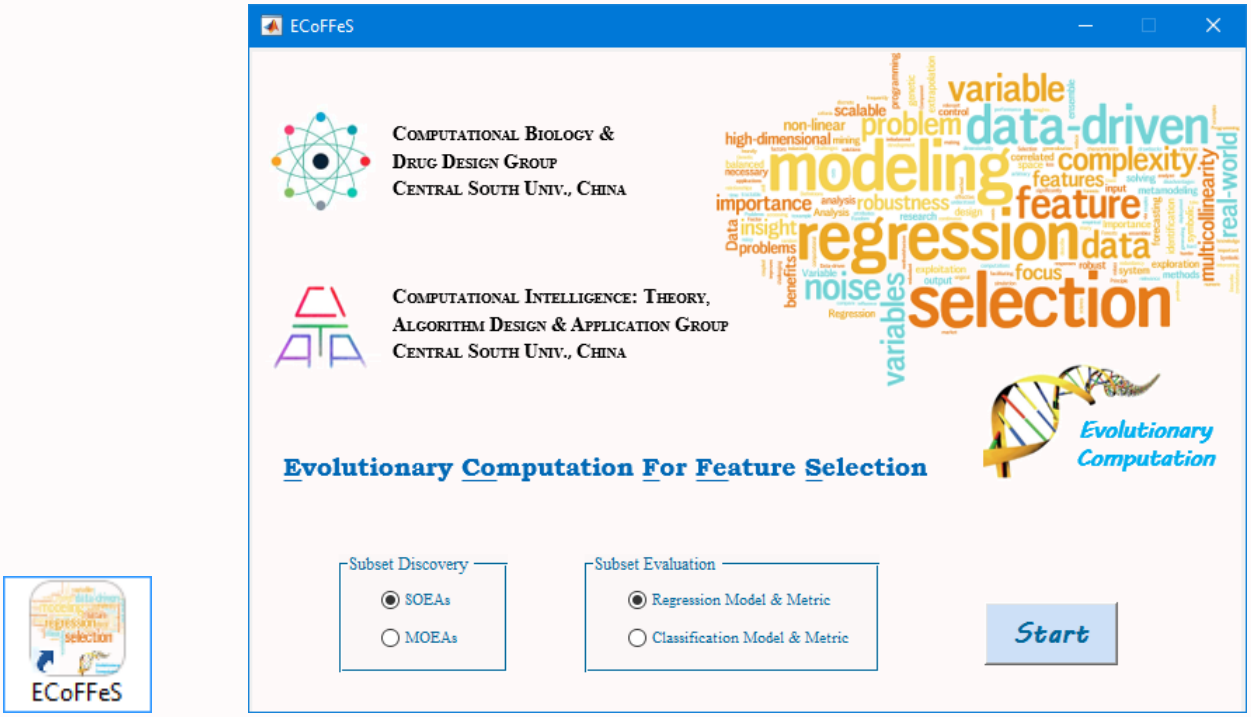
https://github.com/Jiaweihuang/ECoFFeS

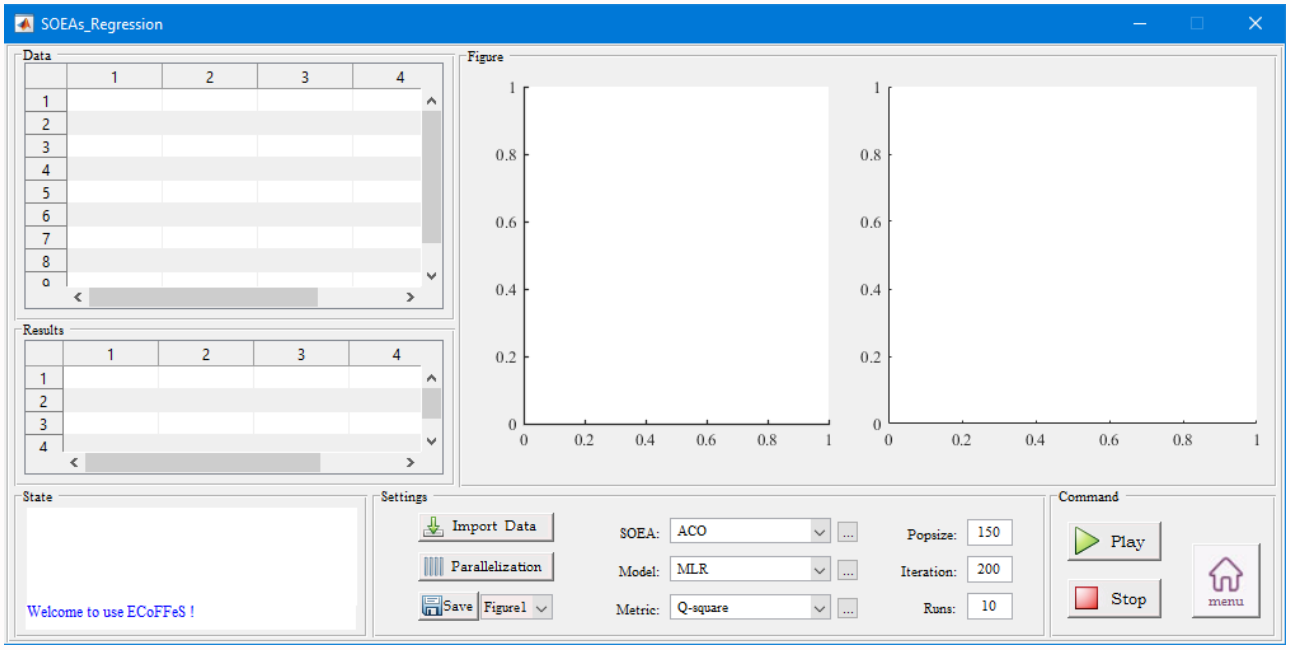
Note: The data format: the first column is Number of Molecules, the second column is $LogD_{7.4}$ (Y) of Molecules and the remain columns are Descriptor Values of Molecules



Step 2 Start the SOEAs_Regression

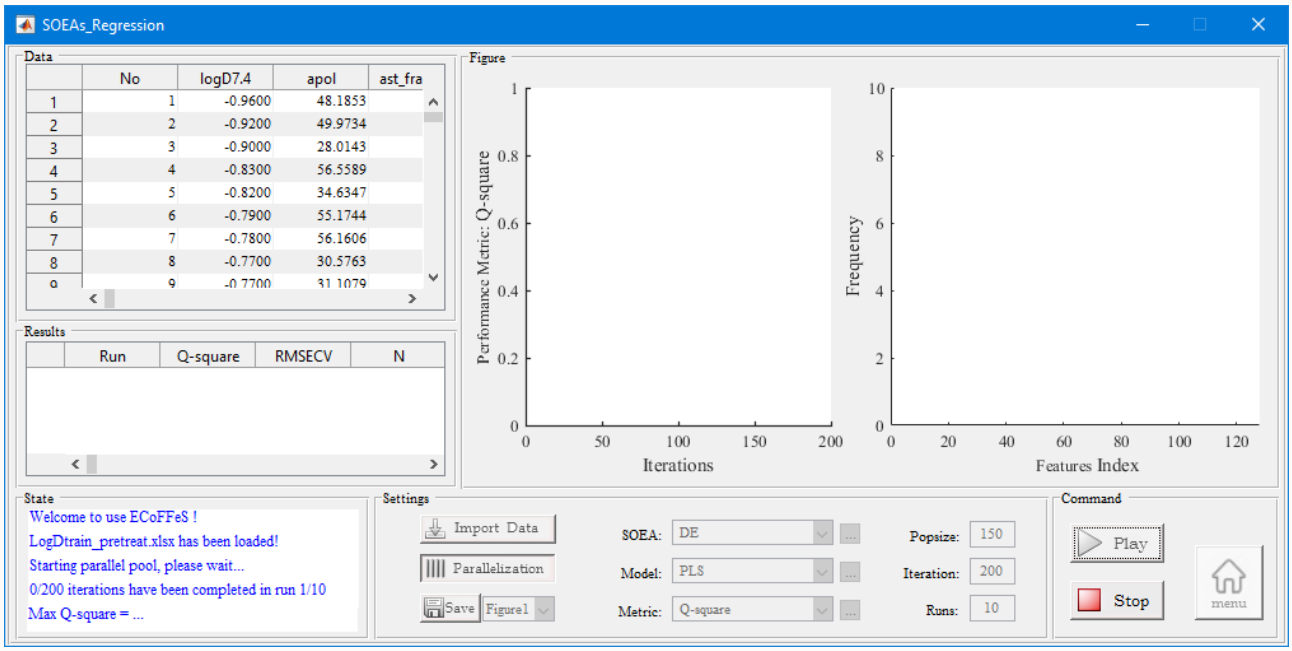
Double-click the ECoFFeS to start the main interface. In main interface, 'Subset Discovery' chooses SOEAs, 'Subset Evaluation' chooses Regression Model & Metric, press Start, then the secondary interface (SOEAs_Regression) appears.





Step 3 Set Parameters and Play

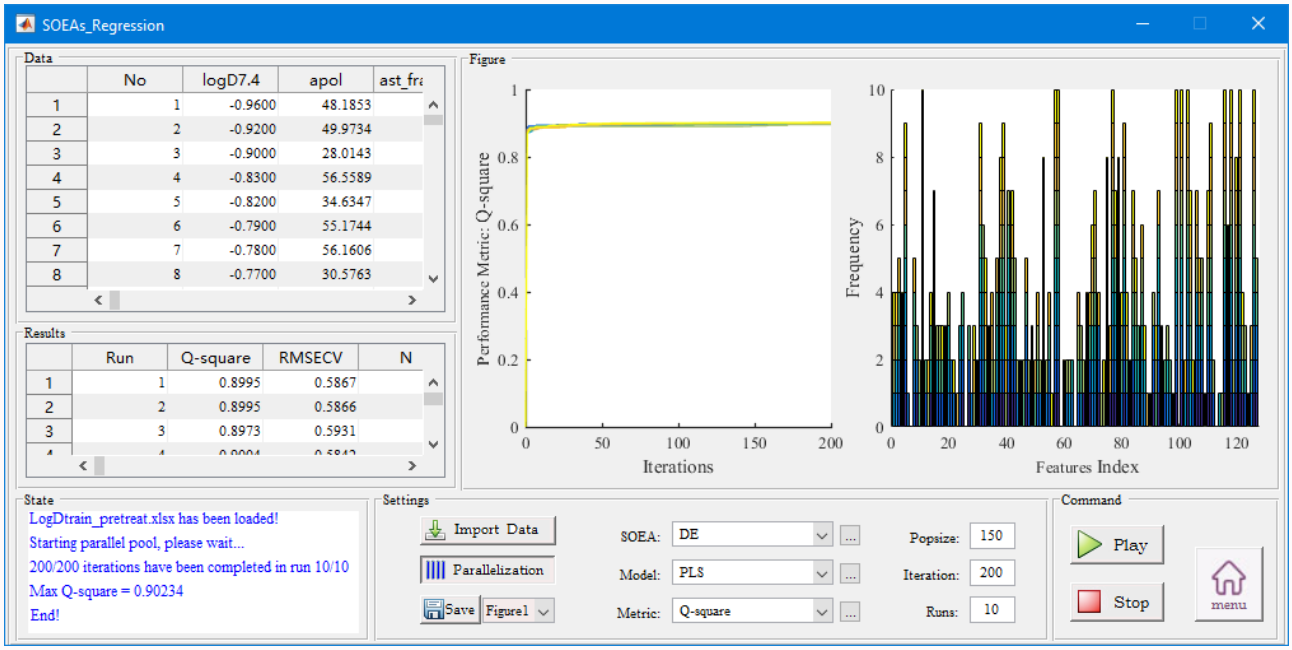
1) Import Data: LogDtrain_pretreat.xlsx;
2) Parallelization;
3) SOEA: DE; Model: PLS; Metric: *Q-square*;
4) SOEA_parameter: Default parameters
5) Model_parameter: 40
6) Metric_parameter: Default parameters
7) Popsize: 150; Iteration: 200; Runs: 10;
8) Press Play;



Step 4 Save Figures and Results

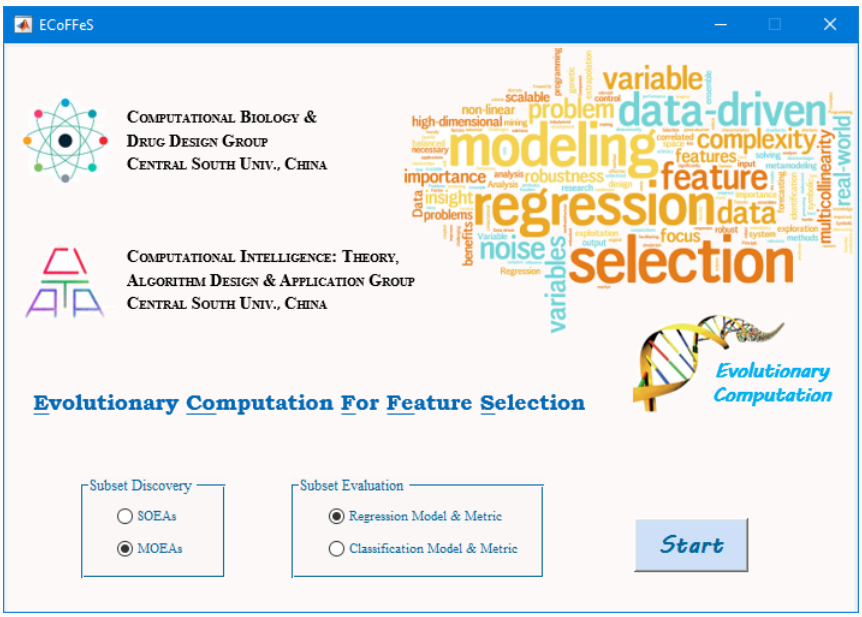
When calculation has been completed, press Save Figure1, Figure2, or Results in the drop-down menu.

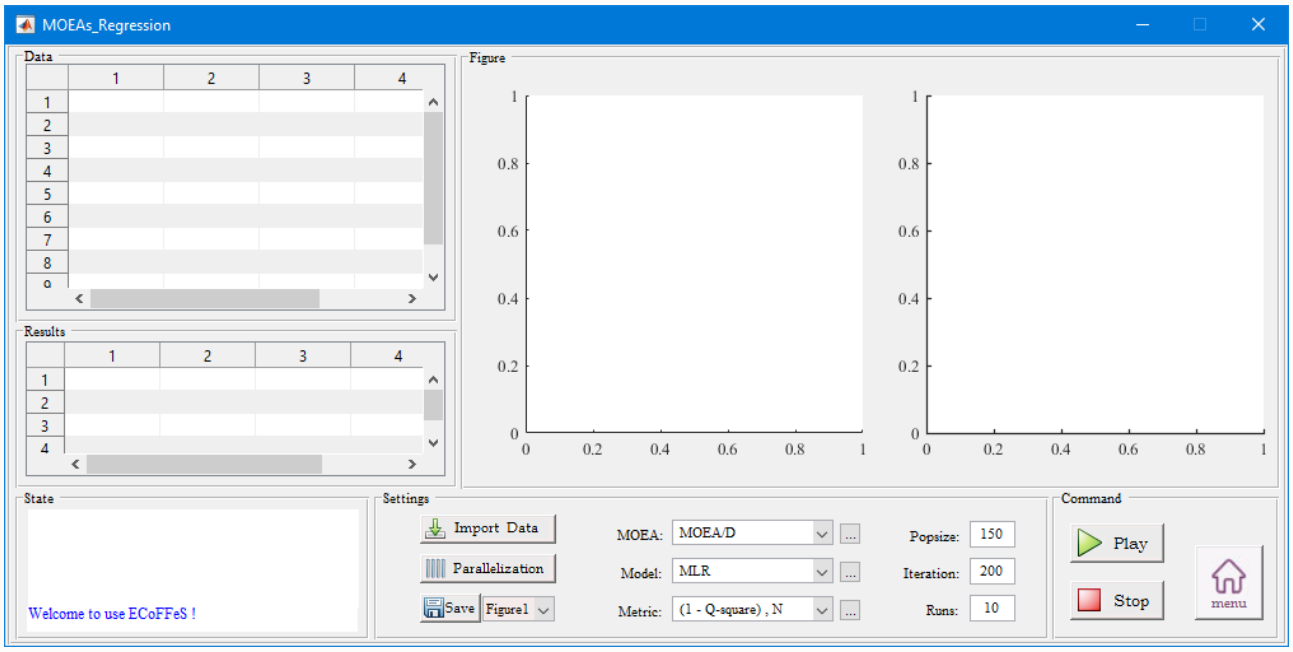
1) Save Figure1: Iteration Figure.jpg
2) Save Figure2: Frequency Figure.jpg
3) Save Results: Results.xlsx



Step 5 Switch to MOEAs_Regression

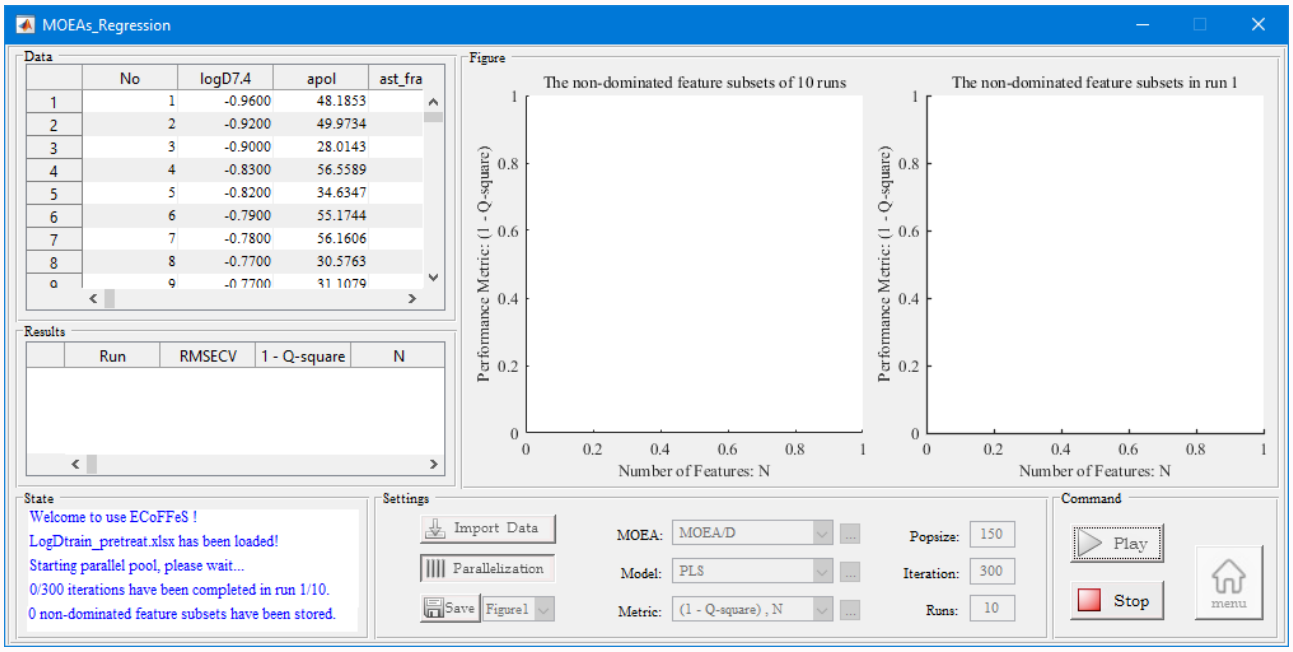
Click menu to return to the main interface. In main interface, 'Subset Discovery' chooses MOEAs, 'Subset Evaluation' chooses Regression Model & Metric, press Start, then the secondary interface (MOEAs_Regression) appears.





Step 6 Set Parameters and Play

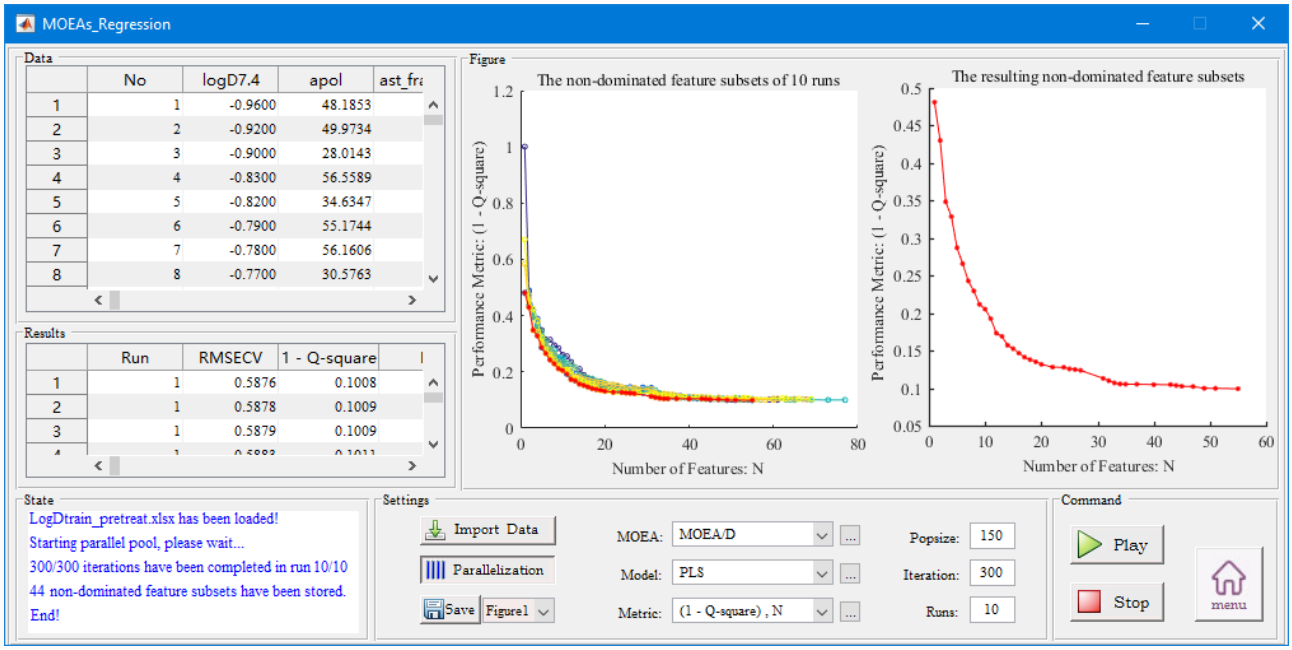
1) Import Data: LogDtrain_pretreat.xlsx;
2) Parallelization;
3) MOEA: MOEA/D; Model: PLS; Metric: (1 – *Q-square*), *N*;
4) MOEAs_parameter: Default parameters;
5) Models_parameter: 40;
6) Metrics_parameter: Default parameters;
7) Popsize: 150; Iteration: 300; Runs: 10;
8) Press Play



Step 7 Save Figures and Results

When calculation has been completed, press Save Figure1, Figure2, or Results in the drop-down menu.

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.

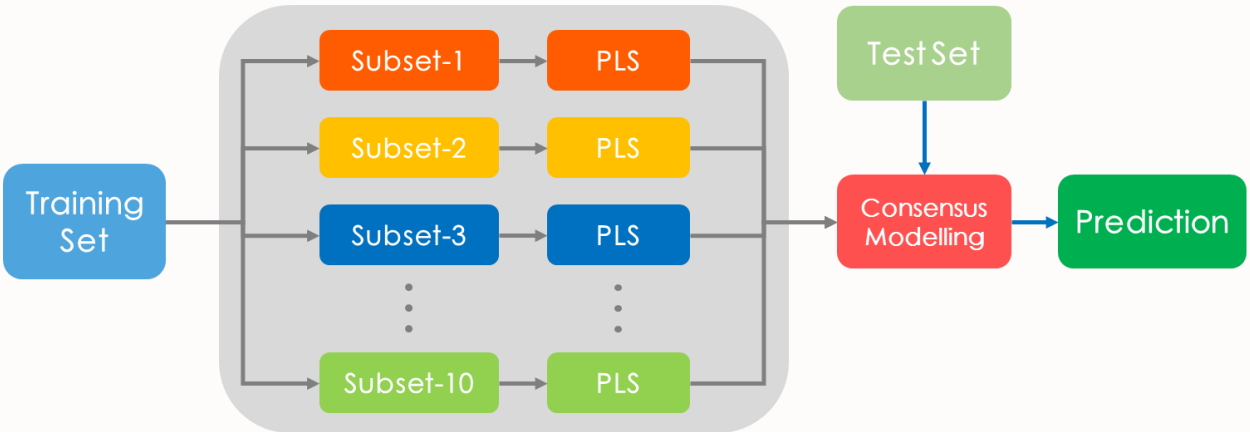


Figure 5.5 the workflow of Consensus Modelling (PLS)

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

model	subset size	training				test		
		R^2_{adj}	Q^2	MAE_{train}	$RMSE_{train}$	Q^2_{ext}	MAE_{test}	$RMSE_{test}$
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 are showed in Figure 5.6.

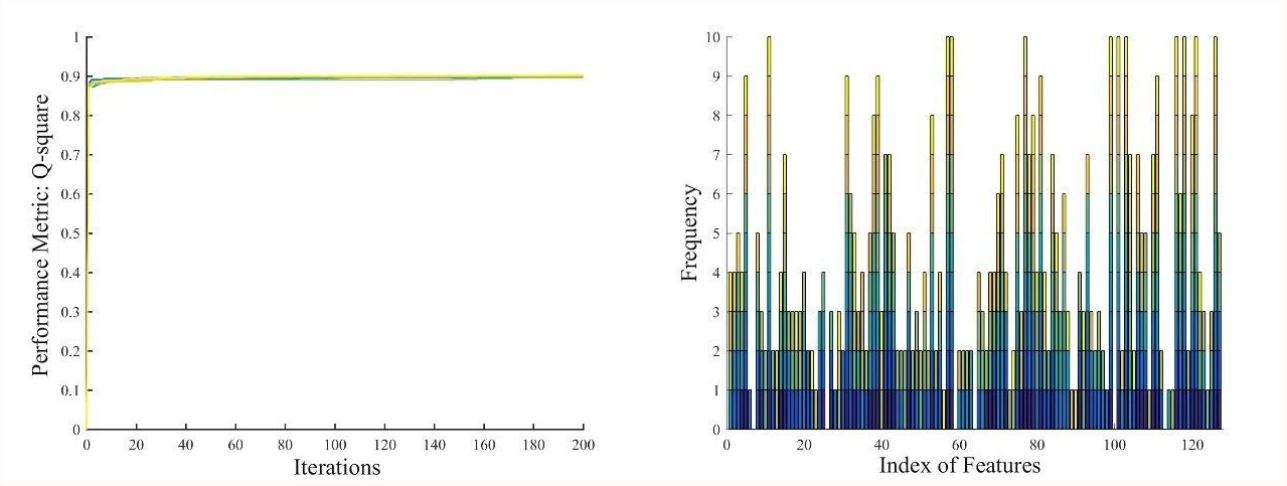


Figure 5.6 The iteration figure and the frequency figure

4) According to the frequency figure, we can acquire the importance of the molecular descriptors and the important descriptors are 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 have 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 (non-dominated) 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. According to the obtained Pareto optimal solutions (i.e., a set of descriptor subsets), a decision-maker can choose a preferred solution (i.e., a preferred descriptor subset) according to his/her own requirements. Here, we select 20 descriptors in order to keep a good balance between *N* and (1 – *Q-square*).

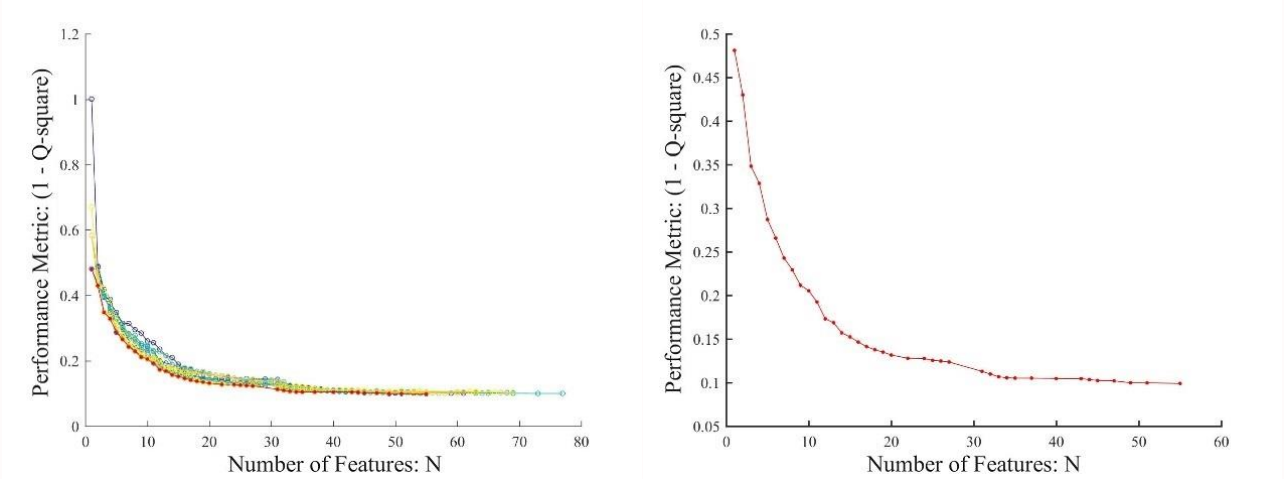


Figure 5.7 The non-dominated descriptor subsets

6) The selected 20 descriptors are 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 Table 5.8.

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

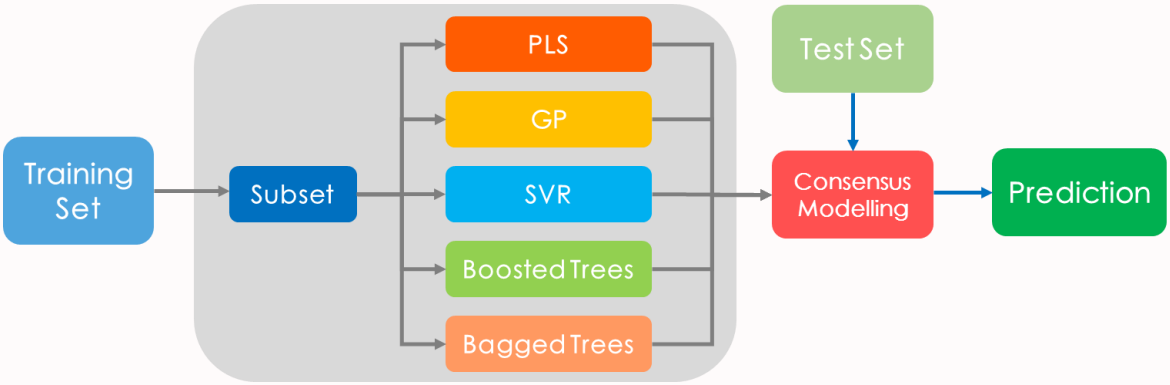


Figure 5.8 Consensus Modelling

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*, Alex F Chen, Dongsheng Cao* and Shengxiang Yang
ECoFFeS: a software for feature selection using single/multi-objective evolutionary algorithms
In submission

Yong Wang
Contact: ywang@csu.edu.cn

Dongsheng Cao
Contact: oriental-cds@163.com

6.2 License

Bibliography

[1] Saeys, Yvan, Iñaki Inza, and Pedro Larrañaga. "A review of feature selection techniques in bioinformatics." *Bioinformatics* 23.19 (2007): 2507-2517.

[2] Wang, Lipo, Yaoli Wang, and Qing Chang. "Feature selection methods for big data bioinformatics: A survey from the search perspective." *Methods* 111 (2016): 21-31.

[3] Khan, Asad U. "Descriptors and their selection methods in QSAR analysis: paradigm for drug design." *Drug Discovery Today* 21.8 (2016): 1291-1302.

[4] Eiben, Agoston E., and Jim Smith. "From evolutionary computation to the evolution of things." *Nature* 521.7553 (2015): 476-482.

[5] Cao, Dongsheng, et al. "Toward better QSAR/QSPR modeling: simultaneous outlier detection and variable selection using distribution of model features." *Journal of computer-aided molecular design* 25.1 (2011): 67-80.

[6] Dash, Manoranjan, and Huan Liu. "Feature selection for classification." *Intelligent data analysis* 1.1-4 (1997): 131-156.

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

[8] John, George H., Ron Kohavi, and Karl Pfleger. "Irrelevant features and the subset selection problem." *Machine learning: proceedings of the eleventh international conference*. 1994.

[9] Roy, Kunal, Supratik Kar, and Rudra Narayan Das. *Understanding the basics of QSAR for applications in pharmaceutical sciences and risk assessment*. Academic press, 2015.

[10] Roy, Kunal, Supratik Kar, and Rudra Narayan Das. *A primer on QSAR/QSPR modeling: Fundamental concepts*. Springer, 2015.

[11] Liu, Yong, Feng Tang, and Zhiyong Zeng. "Feature selection based on dependency margin." *IEEE Transactions on Cybernetics* 45.6 (2015): 1209-1221.

[12] Liu, Huan, and Zheng Zhao. "Manipulating data and dimension reduction methods: Feature selection." *Encyclopedia of Complexity and Systems Science*. Springer New York, 2009. 5348-5359.

[13] Liu, Huan, et al. "Feature Selection: An Ever Evolving Frontier in Data Mining." *FSDM* 10 (2010): 4-13.

[14] Liu, Huan, and Lei Yu. "Toward integrating feature selection algorithms for classification and clustering." *IEEE Transactions on knowledge and data engineering* 17.4 (2005): 491-502.

[15] Unler, Alper, and Alper Murat. "A discrete particle swarm optimization method for feature selection in binary classification problems." *European Journal of Operational Research* 206.3 (2010): 528-539.

[16] Liu, Yuanning, et al. "An improved particle swarm optimization for feature selection." *Journal of Bionic Engineering* 8.2 (2011): 191-200.

[17] Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz. "Handbook of evolutionary computation." *New York: Oxford* (1997).

[18] Fogel, David B. *Evolutionary computation: toward a new philosophy of machine intelligence*. Vol. 1. John Wiley & Sons, 2006.

[19] Xue, Bing, et al. "A survey on evolutionary computation approaches to feature selection." *IEEE*

Transactions on Evolutionary Computation 20.4 (2016): 606-626.

[20] Dorigo, Marco. "Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale." *Ph.D. dissertation*, *Politecnico di Milano*, *Italy* (1992).

[21] Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996): 29-41.

[22] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." *IEEE computational intelligence magazine* 1.4 (2006): 28-39.

[23] Chen, Bolun, Ling Chen, and Yixin Chen. "Efficient ant colony optimization for image feature selection." *Signal processing* 93.6 (2013): 1566-1576.

[24] Yu, Hualong, et al. "A modified ant colony optimization algorithm for tumor marker gene selection." *Genomics, proteomics & bioinformatics* 7.4 (2009): 200-208.

[25] Storn, Rainer, and Kenneth Price. "Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces, Berkeley." *CA: International Computer Science Institute* (1995).

[26] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." *Journal of global optimization* 11.4 (1997): 341-359.

[27] WANG, Yong, et al. "On the selection of solutions for mutation in differential evolution." (2016).

[28] Wang, Yong, Zixing Cai, and Qingfu Zhang. "Enhancing the search ability of differential evolution through orthogonal crossover." *Information Sciences* 185.1 (2012): 153-177.

[29] Pampara, Gary, Andries Petrus Engelbrecht, and Nelis Franken. "Binary differential evolution." *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 2006.

[30] Holland, John H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[31] Holland, John H. "Genetic algorithms." *Scientific american* 267.1 (1992): 66-72.

[32] Roberge, Vincent, Mohammed Tarbouchi, and Gilles Labonté. "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning." *IEEE Transactions on Industrial Informatics* 9.1 (2013): 132-141.

[33] Qiu, Meikang, et al. "Phase-change memory optimization for green cloud with genetic algorithm." *IEEE Transactions on Computers* 64.12 (2015): 3528-3540..

[34] Noraini, Mohd Razali, and John Geraghty. "Genetic algorithm performance with different selection strategies in solving TSP." (2011).

[35] Yen, Yun-Sheng, et al. "Flooding-limited and multi-constrained QoS multicast routing based on the genetic algorithm for MANETs." *Mathematical and Computer Modelling* 53.11 (2011): 2238-2250.

[36] Eberhart, Russell, and James Kennedy. "A new optimizer using particle swarm theory." *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE, 1995.

[37] Shi, Yuhui, and Russell Eberhart. "A modified particle swarm optimizer." *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998.

[38] Kennedy, James. "Particle swarm optimization." *Encyclopedia of machine learning*. Springer US, 2011. 760-766.

[39] Kennedy, James, and Russell C. Eberhart. "A discrete binary version of the particle swarm algorithm." *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*. Vol. 5. IEEE, 1997.

[40] Liu, Hui, Zixing Cai, and Yong Wang. "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization." *Applied Soft Computing* 10.2 (2010): 629-640.

[41] Lee, Sangwook, et al. "Modified binary particle swarm optimization." *Progress in Natural Science* 18.9 (2008): 1161-1166.

[42] Xue, Bing, Mengjie Zhang, and Will N. Browne. "Particle swarm optimization for feature selection in classification: A multi-objective approach." *IEEE transactions on cybernetics* 43.6 (2013): 1656-1671.

[43] Stadler, Wolfram. "A survey of multicriteria optimization or the vector maximum problem, part I: 1776–1960." *Journal of Optimization Theory and Applications* 29.1 (1979): 1-52.

[44] Deb, Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. Vol. 16. John Wiley & Sons, 2001.

[45] Zhou, Aimin, et al. "Multiobjective evolutionary algorithms: A survey of the state of the art." *Swarm and Evolutionary Computation* 1.1 (2011): 32-49.

[46] Zhang, Qingfu, and Hui Li. "MOEA/D: A multiobjective evolutionary algorithm based on decomposition." *IEEE Transactions on evolutionary computation* 11.6 (2007): 712-731.

[47] Li, Hui, and Qingfu Zhang. "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II." *IEEE Transactions on Evolutionary Computation* 13.2 (2009): 284-302.

[48] Miettinen, Kaisa. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 2012.

[49] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6.2 (2002): 182-197.

[50] Deb, Kalyanmoy, and Himanshu Jain. "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints." *IEEE Trans. Evolutionary Computation* 18.4 (2014): 577-601.

[51] Jain, Himanshu, and Kalyanmoy Deb. "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach." *IEEE Trans. Evolutionary Computation* 18.4 (2014): 602-622.

[52] Durillo, Juan J., et al. "Distribution of computational effort in parallel MOEA/D." *International Conference on Learning and Intelligent Optimization*. Springer Berlin Heidelberg, 2011.

[53] Gong, Yue-Jiao, et al. "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art." *Applied Soft Computing* 34 (2015): 286-300.

[54] Molinaro, Annette M., Richard Simon, and Ruth M. Pfeiffer. "Prediction error estimation: a comparison of resampling methods." *Bioinformatics* 21.15 (2005): 3301-3307.

[55] Wold, Svante. "Cross-validatory estimation of the number of components in factor and principal components models." *Technometrics* 20.4 (1978): 397-405.

[56] Wold, Svante. "Validation of QSAR's." *Quantitative Structure-Activity Relationships* 10.3 (1991): 191-193.

[57] Ghose, Arup, and Vellerkad Viswanadhan. *Combinatorial Library Design and Evaluation:*

Principles, Software, Tools, and Applications in Drug Discovery. CRC Press, 2001.

[58] Walker, John D., Lars Carlsen, and Joanna Jaworska. "Improving opportunities for regulatory acceptance of QSARs: the importance of model domain, uncertainty, validity and predictability." *Molecular Informatics* 22.3 (2003): 346-350.

[59] Afantitis, Antreas, et al. "A combined LS-SVM & MLR QSAR workflow for predicting the inhibition of CXCR3 receptor by quinazolinone analogs." *Molecular diversity* 14.2 (2010): 225-235.

[60] Fawcett, Tom. "An introduction to ROC analysis." *Pattern recognition letters* 27.8 (2006): 861-874.

[61] Matthews, Brian W. "Comparison of the predicted and observed secondary structure of T4 phage lysozyme." *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405.2 (1975): 442-451.

[62] Truchon, Jean-François, and Christopher I. Bayly. "Evaluating virtual screening methods: good and bad metrics for the "early recognition" problem." *Journal of chemical information and modeling* 47.2 (2007): 488-508.

[63] Snedecor, George W., and William G. Cochran. "Statistical methods 6th edition." *The Iowa State University* (1967).

[64] Wold, Svante, Michael Sjöström, and Lennart Eriksson. "PLS-regression: a basic tool of chemometrics." *Chemometrics and intelligent laboratory systems* 58.2 (2001): 109-130.

[65] Vapnik, Vladimir Naumovich, and Vlamimir Vapnik. *Statistical learning theory*. Vol. 1. New York: Wiley, 1998.

[66] Cristianini, Nello, and John Shawe-Taylor. "An Introduction to Support Vector Machines and Other Kernel-based Learning Methods." (2000).

[67] Schölkopf, Bernhard, and Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[68] Cover, Thomas, and Peter Hart. "Nearest neighbor pattern classification." *IEEE transactions on information theory* 13.1 (1967): 21-27.

[69] Alpaydin, Ethem. *Introduction to machine learning*. MIT press, 2014.

[70] Finley, Andrew O., and Ronald E. McRoberts. "Efficient k-nearest neighbor searches for multi-source forest attribute mapping." *Remote Sensing of Environment* 112.5 (2008): 2203-2211.

[71] Harrell, Frank. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2015.

[72] Höskuldsson, Agnar. "PLS regression methods." *Journal of chemometrics*2.3 (1988): 211-228.

[73] Barker, Matthew, and William Rayens. "Partial least squares for discrimination." *Journal of chemometrics* 17.3 (2003): 166-173.

[74] Pérez, Néstor F., Joan Ferré, and Ricard Boqué. "Calculation of the reliability of classification in discriminant partial least-squares binary classification." *Chemometrics and Intelligent Laboratory Systems* 95.2 (2009): 122-128.

[75] Ballabio, Davide, and Viviana Consonni. "Classification tools in chemistry. Part 1: linear models. PLS-DA." *Analytical Methods* 5.16 (2013): 3790-3798.

[76] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003): 1-16.

[77] Hsu, Chih-Wei, and Chih-Jen Lin. "A comparison of methods for multiclass support vector machines." *IEEE transactions on Neural Networks* 13.2 (2002): 415-425.

[78] Weston, Jason, and Chris Watkins. "Support vector machines for multi-class pattern

recognition." *ESANN*. Vol. 99. 1999.
[79] Kreßel, Ulrich H-G. "Pairwise classification and support vector machines." *Advances in kernel methods*. MIT Press, 1999.
[80] Hearst, Marti A., et al. "Support vector machines." *IEEE Intelligent Systems and their Applications* 13.4 (1998): 18-28.
[81] Valyon, József, and Gábor Horváth. "A weighted generalized ls-svm." *Periodica Polytechnica, Electrical Engineering* 47.3 (2003): 229-251.
[82] Avery, Mitchell A., et al. "Structure− Activity Relationships of the Antimalarial Agent Artemisinin. 6. The Development of Predictive In Vitro Potency Models Using CoMFA and HQSAR Methodologies." *Journal of medicinal chemistry* 45.2 (2002): 292-303.
[83] Guha, Rajarshi, and Peter C. Jurs. "Development of QSAR models to predict and interpret the biological activity of artemisinin analogues." *Journal of chemical information and computer sciences* 44.4 (2004): 1440-1449.
[84] Cao, Dong-Sheng, et al. "ChemoPy: freely available python package for computational biology and chemoinformatics." *Bioinformatics* (2013): btt105.
[85] Neaz, M. M., et al. "2D-QSAR of non-benzodiazepines to benzodiazepines receptor (BZR)." *Medicinal chemistry research* 18.2 (2009): 98-111.
[86] Haefely, W., et al. "Recent advances in the molecular pharmacology of benzodiazepine receptors and in the structure-activity relationships of their agonists and antagonists." *Advances in drug research* 14 (1985): 165-322.
[87] Selwood, David L., et al. "Structure-activity relationships of antifilarial antimycin analogs: a multivariate pattern recognition study." *Journal of medicinal chemistry* 33.1 (1990): 136-142.
[88] Nicolotti, Orazio, et al. "Multiobjective optimization in quantitative structure− activity relationships: Deriving accurate and interpretable QSARs." *Journal of Medicinal Chemistry* 45.23 (2002): 5069-5080.
[89] Wang, Yong, et al. "Incorporating PLS model information into particle swarm optimization for descriptor selection in QSAR/QSPR." *Journal of Chemometrics* 29.12 (2015): 627-636.
[90] Triggle, David J., and John B. Taylor. *Comprehensive medicinal chemistry II: editors-in-chief, John B. Taylor, David J. Triggle*. Elsevier, 2007.
[91] Cheng, Feixiong, et al. "admetSAR: a comprehensive source and free tool for assessment of chemical ADMET properties." (2012): 3099-3105.
[92] Kola, Ismail, and John Landis. "Can the pharmaceutical industry reduce attrition rates?." *Nature reviews Drug discovery* 3.8 (2004): 711-716.
[93] Merlot, Cedric. "Computational toxicology—a tool for early safety evaluation." *Drug discovery today* 15.1 (2010): 16-22.
[94] Hou, Tingjun, and Junmei Wang. "Structure–ADME relationship: still a long way to go?." *Expert opinion on drug metabolism & toxicology* 4.6 (2008): 759-770.
[95] Wang, Sichao, et al. "ADMET evaluation in drug discovery. 12. Development of binary classification models for prediction of hERG potassium channel blockage." *Molecular pharmaceutics* 9.4 (2012): 996-1010.
[96] Wang, Shuangquan, et al. "ADMET evaluation in drug discovery. 16. Predicting hERG blockers by combining multiple pharmacophores and machine learning approaches." *Molecular Pharmaceutics* 13.8 (2016): 2855-2866.
[97] Vandenberg, Jamie I., et al. "hERG K+ channels: structure, function, and clinical significance."

Physiological Reviews 92.3 (2012): 1393-1478.

[98] Recanatini, Maurizio, et al. "QT prolongation through hERG K+ channel blockade: current knowledge and strategies for the early prediction during drug development." *Medicinal research reviews* 25.2 (2005): 133-166.

[99] Villoutreix, Bruno O., and Olivier Taboureau. "Computational investigations of hERG channel blockers: New insights and current predictive models." *Advanced drug delivery reviews* 86 (2015): 72-82.

[100] Witchel, Harry J. "The hERG potassium channel as a therapeutic target." *Expert opinion on therapeutic targets* 11.3 (2007): 321-336.

[101] Brugada, Ramon, et al. "Sudden death associated with short-QT syndrome linked to mutations in HERG." *Circulation* 109.1 (2004): 30-35.

[102] Curran, Mark E., et al. "A molecular basis for cardiac arrhythmia: HERG mutations cause long QT syndrome." *Cell* 80.5 (1995): 795-803.

[103] Brown, A. M. "Drugs, hERG and sudden death." *Cell calcium* 35.6 (2004): 543-547.

[104] Aronov, Alex M. "Predictive in silico modeling for hERG channel blockers." *Drug discovery today* 10.2 (2005): 149-155.

[105] Raschi, Emanuel, et al. "The hERG K+ channel: target and antitarget strategies in drug development." *Pharmacological research* 57.3 (2008): 181-195.

[106] Bains, William, Antranig Basman, and Cat White. "HERG binding specificity and binding site structure: evidence from a fragment-based evolutionary computing SAR study." *Progress in biophysics and molecular biology* 86.2 (2004): 205-233.

[107] Roche, Olivier, et al. "A virtual screening method for prediction of the HERG potassium channel liability of compound libraries." *ChemBioChem* 3.5 (2002): 455-459.

[108] Broccatelli, Fabio, et al. "QSAR modeling and data mining link torsades de pointes risk to the interplay of extent of metabolism, active transport, and hERG liability." *Molecular pharmaceutics* 9.8 (2012): 2290-2301.

[109] Sinha, Nandita, and Srikanta Sen. "Predicting hERG activities of compounds from their 3D structures: Development and evaluation of a global descriptors based QSAR model." *European journal of medicinal chemistry* 46.2 (2011): 618-630.

[110] Wang, Sichao, et al. "Recent developments in computational prediction of HERG blockage." *Current topics in medicinal chemistry* 13.11 (2013): 1317-1326.

[111] Jing, Yankang, et al. "In silico prediction of hERG inhibition." *Future medicinal chemistry* 7.5 (2015): 571-586.

[112] Wang, Yanli, et al. "PubChem's BioAssay database." *Nucleic acids research* 40.D1 (2012): D400-D412.

[113] Aronov, Alex M., and Brian B. Goldman. "A model for identifying HERG K+ channel blockers." *Bioorganic & medicinal chemistry* 12.9 (2004): 2307-2315.

[114] Li, Qiyuan, et al. "hERG classification model based on a combination of support vector machine method and GRIND descriptors." *Molecular pharmaceutics* 5.1 (2008): 117-127.

[115] Studio, Discovery. "version 2.5." *Accelrys Inc.: San Diego, CA, USA* (2009).

[116] Waring, Michael J., and Craig Johnstone. "A quantitative assessment of hERG liability as a function of lipophilicity." *Bioorganic & medicinal chemistry letters* 17.6 (2007): 1759-1764.

[117] ChemicalComputingGroup, M. O. E. "Molecular Operating Environment." (2008).

[118] Waring, Michael J. "Lipophilicity in drug discovery." *Expert opinion on drug discovery* 5.3

(2010): 235-248.

[119] Remko, Milan, Andrej Boháč, and Lucia Kováčiková. "Molecular structure, pKa, lipophilicity, solubility, absorption, polar surface area, and blood brain barrier penetration of some antiangiogenic agents." *Structural Chemistry* 22.3 (2011): 635-648.

[120] Leo, Albert, Corwin Hansch, and David Elkins. "Partition coefficients and their uses." *Chemical reviews* 71.6 (1971): 525-616.

[121] NU, Shawahna R. Rahman. "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." *Daru* 19.2 (2011).

[122] Souza, Erica Silva, et al. "Estimating the octanol/water partition coefficient for aliphatic organic compounds using semi-empirical electrotopological index." *International journal of molecular sciences* 12.10 (2011): 7250-7264.

[123] Cao, Dong-Sheng, et al. "Prediction of aqueous solubility of druglike organic compounds using partial least squares, back-propagation network and support vector machine." *Journal of Chemometrics* 24.9 (2010): 584-595..

[124] Pollastri, Michael P. "Overview on the Rule of Five." *Current protocols in pharmacology* (2010): 9-12.

[125] Lipinski, Christopher A., et al. "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings." *Advanced drug delivery reviews* 23.1-3 (1997): 3-25.

[126] Bhal, Sanjivanjit K., et al. "The Rule of Five revisited: applying log D in place of log P in drug-likeness filters." *Molecular pharmaceutics* 4.4 (2007): 556-560.

[127] Comer, John, and Kin Tam. "Lipophilicity profiles: theory and measurement." *Pharmacokinetic optimization in drug research* (2001): 275-304.

[128] Ermondi, Giuseppe, Miriam Lorenti, and Giulia Caron. "Contribution of ionization and lipophilicity to drug binding to albumin: a preliminary step toward biodistribution prediction." *Journal of medicinal chemistry* 47.16 (2004): 3949-3961.

[129] Zhivkova, Zvetanka, and Irini Doytchinova. "Quantitative structure–clearance relationships of acidic drugs." *Molecular pharmaceutics* 10.10 (2013): 3758-3768.

[130] Kah, Melanie, and Colin D. Brown. "LogD: Lipophilicity for ionisable compounds." *Chemosphere* 72.10 (2008): 1401-1408.

[131] Wang, Jian-Bing, et al. "In silico evaluation of logD7. 4 and comparison with other prediction methods." *Journal of Chemometrics* 29.7 (2015): 389-398.

[132] Kennard, Ronald W., and Larry A. Stone. "Computer aided design of experiments." *Technometrics* 11.1 (1969): 137-148.

[133] Kocjančič, Robert, and Jure Zupan. "Modelling of the river flowrate: the influence of the training set selection." *Chemometrics and Intelligent Laboratory Systems* 54.1 (2000): 21-34.