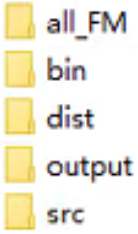
This archive contains feature models and source codes used in the paper "Looking For Novelty in Search-based Software Product Line Testing", by Yi Xiang, Han Huang, Senior Member, IEEE, Miqing Li, Sizhe Li, and Xiaowei Yang.

1. Introduction to archive structures

There are five folders in the released archive. Here is a brief introduction to each of them.



- The *all_FM* folder contains all feature models used in the paper.

- The *bin* folder contains all binary files after compiling.

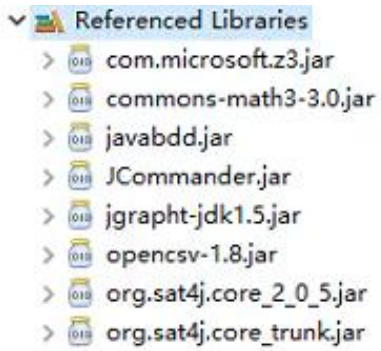
- The *dist* folder contains some JAR libraries needed to run the experiments.

- The *output* folder stores all outputs of the experiments.

- The *src* folder contains all Java source codes.

2. How to run this program?

Step 1: Load the project into Eclipse platform. Note that all JARs in the *dist* folder should be added into the "**Referenced Libraries**" in Eclipse (see below).



Step 2: Open *src/spl/SPL.java*, and locate at the *main*() function, the only entry of this program.

Step 3: Run different experiments as follows:

- To find correlations between t-wise coverage and novelty scores, use the following function in *main*()

  ```
  SPL.getInstance().fitnessRelateCoverage();
  ```

  It will automatically generate MATLAB scripts to perform correlation analysis. For example, we have generated the following piece of scripts for the axTLS model (See Fig. 1). These scripts are stored in *FM_axTLS*.m. After this, you can run these MATLAB scripts to get the Pearson's correlation coefficient *r* and *p*-values. This is achieved by calling the *corrcoef* function: *[R,P] = corrcoef(data).*

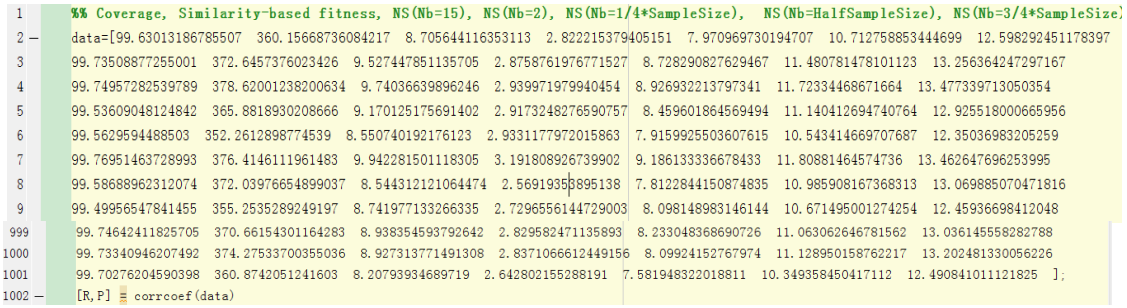


Figure 1. An example of MATLAB scripts for correlation analysis

- To run Unpredictable, NS, GA, SamplingDown and Henard's GA, please use the following functions, respectively. These procedures will automatically save results into files.

```
SPL.getInstance().samplingProductsSAT4JUnpredictable;//Unpredictable
SPL.getInstance().findProductsNSR1; //NS
SPL.getInstance().findProductsGAR1; //GA
SPL.getInstance().samplingDownProductsR1;  //SamplingDown
SPL.getInstance().findProductsGA; //Henard's GA
```

Step 4: Get t-wise coverage or fault detection rate, using *spl.GenerateFromExistingResultsMain.java*. More precisely,

--To compute t-wise coverage, use *computeTwiseCoverage()* in the *main*() function;
--To compute fault detection rate, use *computeFaultRate()* in the *main*() function.

Step 5: Generate Latex tables, reporting means of indictors, e.g., Coverage and fault detection rate. In *SPL/GenerateTablesMain.java*, we have implemented procedures to automatically generate Latex tables. Note that there are two ways to generate tables.

```
generateLatexTables(false) // Without showing Mann-Whitney U test results
generateLatexTables(true) /* Showing Mann-Whitney U test results, and this requires to get the
*.tr files (see Section 3.1 for more details) */
```

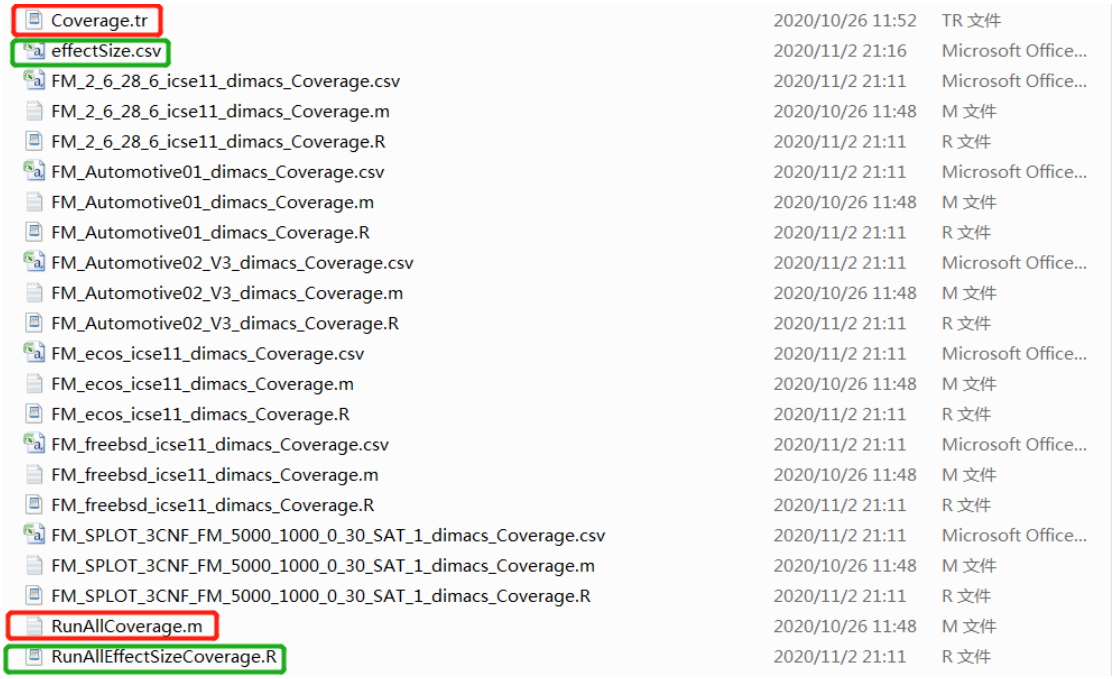
Note that, to further make reproducibility easy, a three-minute video (*HowToRun*.mp4) was provide to demonstrate how to run this program step by step.

3. Ad-hoc procedures

Here are some useful ad-hoc procedures.

3.1 Statistical Analysis

In *jmetal.myutils.datacolletion.CollectionDataForTest.java*, we have implemented procedures to automatically generate MATLAB scripts for the Mann-Whitney U test, and R scripts for computing effect size. After configuring and running *jmetal.myutils.datacolletion.CollectionDataForTestMain.java*, we can get a folder with the following contents.



Then, you just need to run *RunAllCoverage*.m in MATLAB to get *Coverage*.tr file, which stores the Mann-Whitney U test results. They are represented by three symbols '+', '-' and '=', stating that the first algorithm, i.e., the new proposal, performs significantly better than, worse than and equivalently to each of the peer algorithms, respectively. Also, you need to run *RunAllEffectSizeCoverage*.R in R platform to get the effectSize.csv file. This file stores values of the effect size, along with their magnitudes. Note that, to present Mann-Whitney U test results stored in the *Coverage*.tr file, you need to use this line `generateLatexTables(true)` in *GenerateTablesMain.java*. Note also that, in our paper, '+', '-' and '=' are presented as '•', '◦' and '‡', respectively. Of course, any symbols you prefer to can be chosen.

3.2 **Problem-related Utils**

In the following package, we provided tools for finding core and dead features (*coreAndDeadFeatures*.java), and for generating artificial faults (*generateArtificalfaults*.java).