**实验前提：**

（1）已知某个项目的小版本数据集文件。

1. 运用各种缺陷预测模型(rf/lm/lr/svm)+不同的预测类型(prob+num+prob)所得到的预测结果文件已知。
2. 已知N种工作量策略
3. 已知测试工作量覆盖的模块Top X (X==1代表全测,X=0.2代表只测20%)
4. 已知剩余缺陷率R1,R2

（6）用版本V i预测V i+1

（7）合并版本v0~v5来预测版本v6

**实验结果:**

**版本1：按照原文作者的思路来实现，缺陷发现模型是GO模型。我画出了策略A1~A11,B1~B4,C1~C4，以及最优模型O的预测结果图。横坐标为“当前测试工作量/实际测试工作量×100%”，纵坐标为“model在分配策略S下能发现的缺陷总数 / 总的实际缺陷总数 × 100%"。**

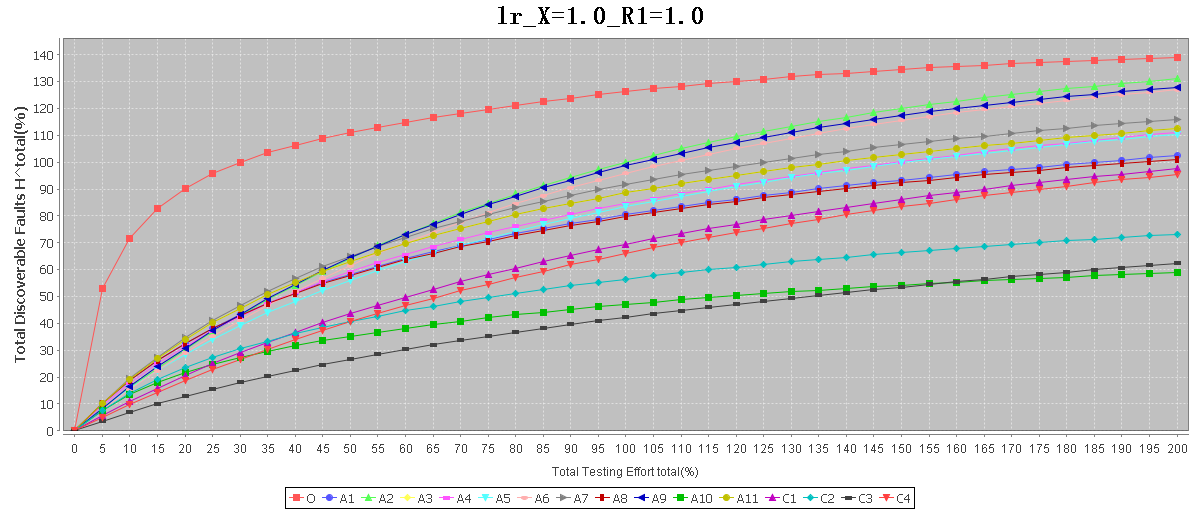
1. 版本1的结果都在这个目录下：

QQ截图20151128212558

1. 文件目录结构说明：

例：

1. GO/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.png 是预测结果图，如下：



其中，lang是项目名，prob是预测类型，8\_by\_all 是指实验结果是基于缺陷预测时V2->v1的结果进行的，lr表示预测模型是lr,X=1.0代表全部模块都分配测试工作量，如果X=0.5代表只把测试工作量分给50%的模块，R1代表剩余缺陷率。

2. GO/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.xls是图中的每个点对应的具体坐标。

3. GO/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.doc是每个策略要达到实际bug时所需的test数目与实际数目的百分比。结果如下：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prediction type | Modeling technique | Required testing effort total(%) | | | | | | | | | | | | | | |
| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | C1 | C2 | C3 | C4 |
| prob | lr | 186% | 100% | 138% | 150% | 151% | 109% | 126% | 193% | 103% |  | 138% | 214% | 669% | 572% | 224% |

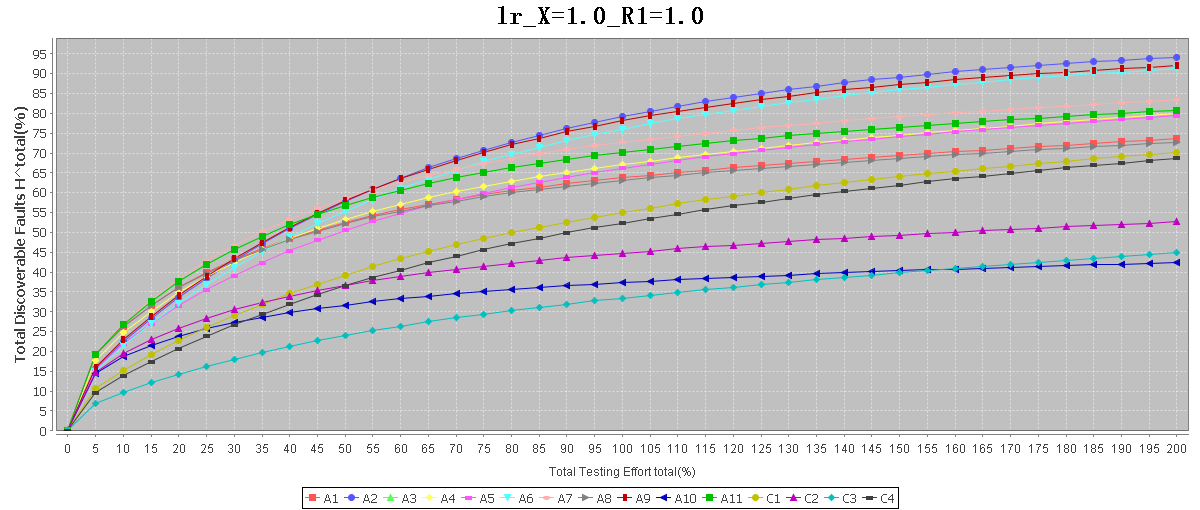
**版本2：与版本1不同的是，横坐标为“当前测试工作量/实际测试工作量×100%”，纵坐标改成了"model在分配策略S下能发现的缺陷总数 / 在最优分配策略O下能发现的缺陷总数 ×100%“。**

1. 版本2的结果都在这个目录下：

QQ截图20151202172605

1. 文件目录结构说明(同上)

GO\_2/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.png:



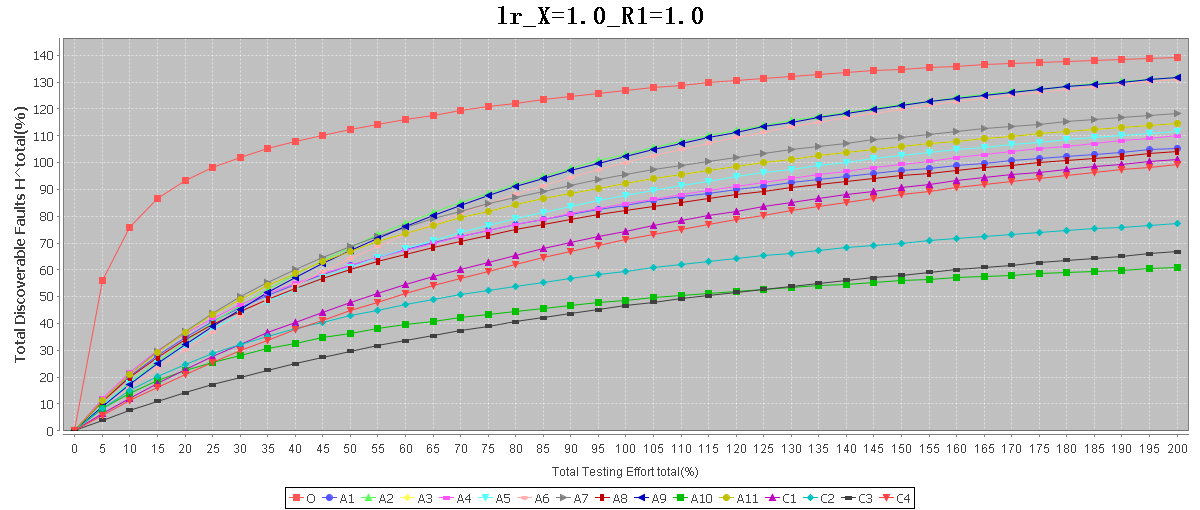
**版本3：缺陷发现模型仍然是Go模型，但是把方程中的Sloc换成了圈复杂度wmc。横坐标为“当前测试工作量/实际测试工作量×100%”，纵坐标为"model在分配策略S下能发现的缺陷总数 / 总的实际缺陷总数 × 100%“。**

（1）版本3的结果都在这个目录下：

QQ截图20151203135745

（2）文件目录结构说明(同上)

GO\_WMC/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.png:

****

GO\_WMC/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.doc:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prediction type | Modeling technique | Required testing effort total(%) | | | | | | | | | | | | | | |
| A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | C1 | C2 | C3 | C4 |
| prob | lr | 167% | 94% | 125% | 153% | 140% | 100% | 114% | 176% | 96% |  | 125% | 194% | 561% | 557% | 205% |

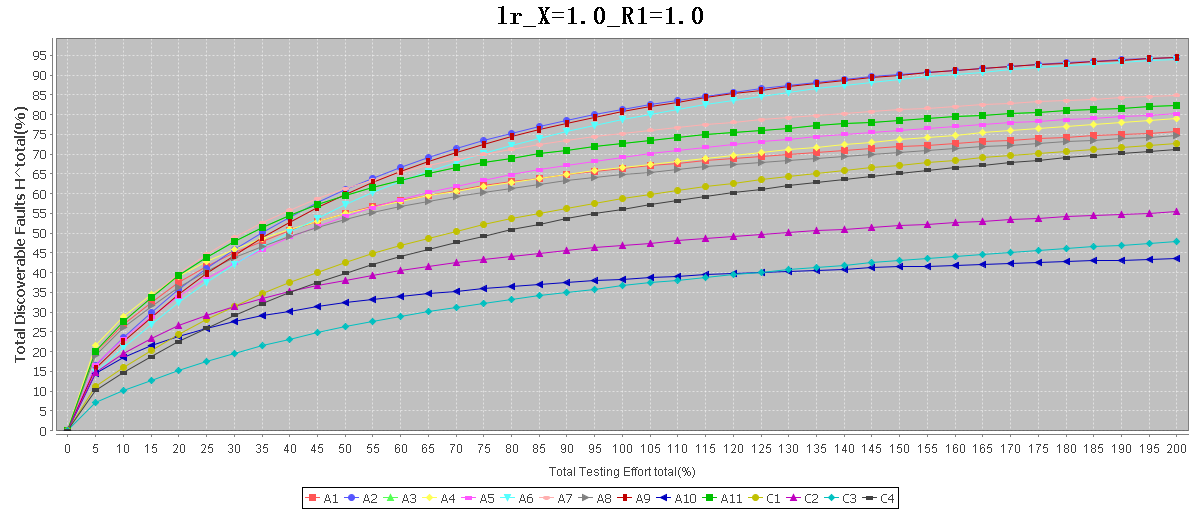
**版本4：与版本3不同的是，横坐标为“当前测试工作量/实际测试工作量×100%”，纵坐标改成了"model在分配策略S下能发现的缺陷总数 / 在最优分配策略O下能发现的缺陷总数 ×100%“。**

（1）版本4的结果都在这个目录下：

QQ截图20151203135808

（2）文件目录结构说明(同上)

GO\_WMC2/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.png:



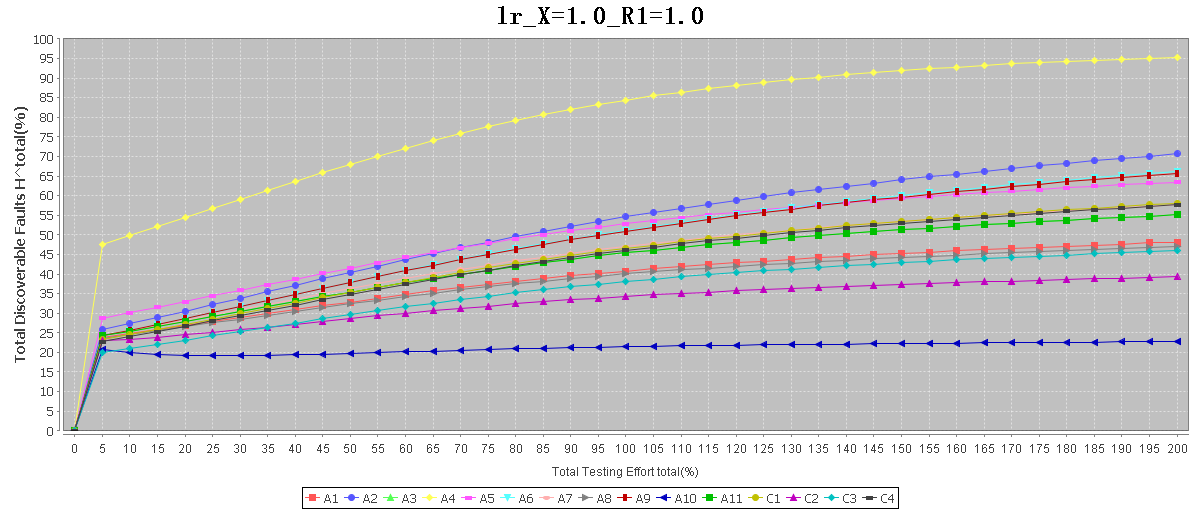
**版本5：缺陷发现模型仍然是Go模型，但是估计ai和b0的方式是通过v0来得到v1。横坐标为“当前测试工作量/实际测试工作量×100%”，纵坐标为"model在分配策略S下能发现的缺陷总数 / 在最优分配策略O下能发现的缺陷总数 ×100%“。**

（1）版本5的结果都在这个目录下：

QQ截图20151202173507

(2)文件目录结构说明(同上)

(3)样例：GO\_AIB0/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.png：

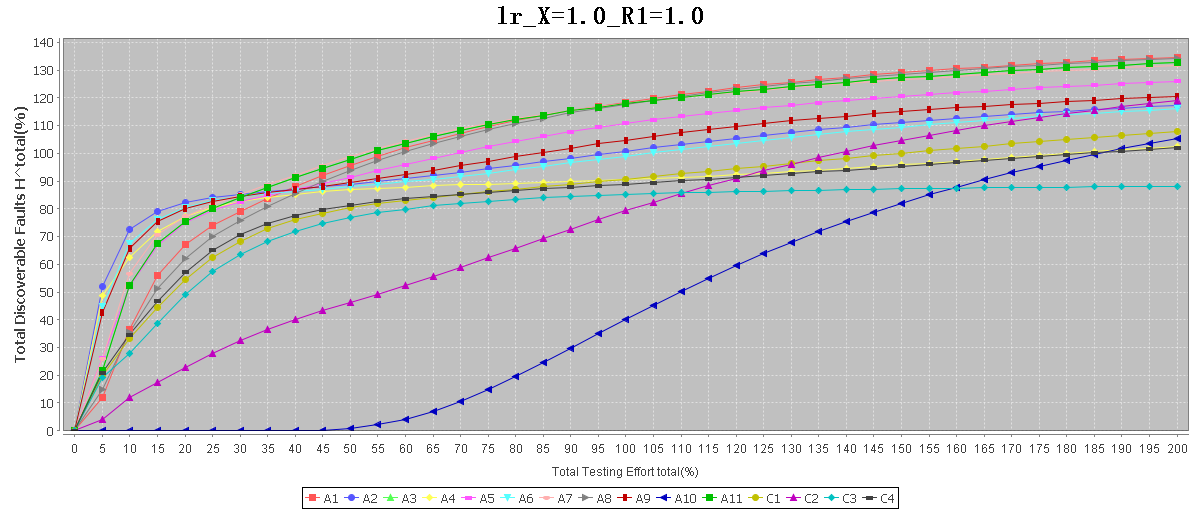


**版本6：缺陷发现模型是Rayleigh，Rayleigh无法得到最优工作量分配策略。横坐标为“当前测试工作量/实际测试工作量×100%”，纵坐标为“model在分配策略S下能发现的缺陷总数 / 总的实际缺陷总数 × 100%"。**

1. 版本6的结果都在这个目录下：

QQ截图20151202192643

1. 文件目录结构同版本1。
2. 样例：Rayleigh/lang/prob/8\_by\_all/(lang.prob.8\_by\_all)lr\_X=1.0\_R1=1.0.png:



具体实验结果：https://github.com/faultPrediction/cefprat/tree/master/new/result