The Manual for RMiner

(Version 1.0)

Intelligent and Distributed Computing Laboratory
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, Hubei 430074, China
http://idc.hust.edu.cn/rminer/

Ruixuan Li
Wei Wang
Huaqing Li
Xiaopu Ma
Wei Ye

January 10, 2013

Table of Contents

Chapter 1

Introduction

The goal of role engineering is to configure a Role-Based Access Control (RBAC) system. Usually, role engineering process is to generate roles, assign roles to users and update role state. Role engineering has two basic approaches top down and bottom-up. The top-down uses descriptions of business processes, security policies, and other business information to configure an RBAC system. The bottom-up uses data mining techniques to discover roles from the existing user-permission assignments and other business information. The bottom-up approach is commonly called role mining.

The role mining algorithm discovers roles from the real world data set which is usually with noise. Thus it is difficult to generate a role state that is accurately satisfied the security needs for an RBAC system. Researches use the ratio of the number of generated roles exactly matching the original role set to the number of original role set to evaluate a role mining algorithm. When the original role set is not existed, the WSC [7, 8] which sums up the number of relationships in an RBAC state is often used to evaluate them. We think role mining algorithm will not completely solve the role configuring problem, sometimes we need administrators to correct some unsuitable roles generated by role mining algorithm. In short, we need implementation of role mining algorithms to generated roles and interactive tools to update the role state.

RMiner is such a tool set that provides implementation of some classic role mining algorithms such as ORCA [9], HierarchicalMiner[10], FastMiner[11] for generating roles and a role state editor for updating role state. In addition, RMiner offers some abstract interfaces that user can add new algorithms based on them. Users will easily discover a new algorithm's difference from others using the visualization tools. All these design of tool set are learned from WEKA [1-6].

Many existing algorithms are learning though from the data mining field. However, the workbench environment for data mining cannot do role mining well. We take WEKA [1-6] for an example to show the reasons. Firstly, the input of WEKA is a set of instances that describes the relationship between attributes. In contrast, the input of role mining algorithm contains the user to permission assignments and some business information. An instance for WEKA indicates values on every attributes. This is same with the description for business information in role mining field. However, if we use instances to indicate the assignment, we must list all the permissions but every value for them is just 0 or 1, it is quit redundant and easy to mix assignment and attribute. Secondly, the output for role mining is a set of associated permissions. This is similar to association a rule mining in WEKA, but the evaluating criteria in role mining is different from support and confidence that used in association rules. The rules with low frequency will be a role in role mining field. Thirdly, there is lack of a commonly implementation of role mining algorithms, thus researches are hard to do a comparison experiment. In summary, the role of

mining algorithms can't be migrated to WEKA directly, so we need to do same changes on WEKA to adapt to the characteristics of role mining.

At last, RMiner is implemented by Java that can cross platforms. The interaction of RMiner is friendly. Users can easily to operate on it to build an RBAC system. The hierarchy of various modules in RMiner is clear and the interfaces are strong scalability. Users can develop a new algorithm on it rapidly.

Chapter 2

Data and Framework

2.1 Data

An ass file is an ASCII text file that describes the relationship of users and permissions. An ur file describes the relationship of users and roles. An rp file describes the relationship of roles and permissions.

These files have two distinct sections. The first section is the header information, which is followed the data information. The headers of these files contain the names of the relations and the contents included by the relationship' dimension. An example header on the standard ass dataset looks like this:

@assignment upa
@dimension user {Ann,Bob,Carl,Doro,Ed,Fay}
@dimension permission {rAcc,wAcc,cdAcc,cTrans,rTrans}

Where, the upa is the name of the assignment relationship, from the above relationship, we can know the assignment is a two dimensions including user set and permission set. The users contain Ann,Bob, Carl,Doro,Ed and Fay and the permissions contain rAcc,wAcc,cdAcc,cTrans and rTrans.

The data of these files looks as follows:

@matrix
1,1,0,1,0
1,1,0,1,0
1,0,1,0,1
1,0,1,0,1
1,1,0,1,0

1,1,1,1,1

The two dimension relationship usually use matrix to describe. In the role mining field, because user has some permission completely or hasn't some permission. So the assignment matrix is a Boolean matrix whose elements are one or zero.

In the ass file, there may be many attributes about users or permissions. If there is the users' attribute information, it looks like this:

@attributes of user
@attribute post {U, UH, GTG, G, GT, GR, UHU, GRG}
@data
U
UHU
U
UH
GTG
GT
GRG
GR
G
GTG

In this example, we can see the user has the position information which is the user's attribute. The set of the post is showed in post field.

If there is the permissions' attribute information in the ass file, it looks as follows:

@attributes of permission
@ATTRIBUTE sepallength REAL
@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE petallength REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}
@DATA
5.1,3.5,1.4,0.2, Iris-setosa
4.9,3.0,1.4,0.2, Iris-virginica
4.7,3.2,1.3,0.2, Iris-versicolor
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2, Iris-versicolor
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3, Iris-virginica

We can see that the permission has five fields and four of them are real type and one is class type. The names of fields are displayed in ATTRIBUTE field, for example, sepallength. The last attribute field called class has three values. You can see the value from @DATA domain.

At last, please note first the @assignment, @dimension and @matrix declarations are case insensitive, and then the format of the rp and ur files are same with the ass file. So reader should can infer from the ass file when you see the ur and rp files.

2.2 The framework of RMiner

We want to design a tool to help researchers or administrator do a role engineer work. The tool also can do some comparison experiments to let them understand characteristics of algorithms clearly. The framework of RMiner is shown in Figure 1. The framework includes three parts, which are user, user interface and implementation of algorithms.

Firstly, user must transform the other file format to required file format that has been introduced in Section 2.1. Generally speaking, if the input includes the predefined set of roles, the accuracy of algorithm selected by user will be calculated. After that, users can remove the redundancy or noise in the input. RMiner loads the input data set and uses the selected algorithm to generate roles. The output of selected algorithm contains: ROLES (set of roles), UA (user to role assignments) and PA (role to permission assignments). Of course, if these outputs are not satisfied with user's security need, user can update them by a visualization tool: role editor. At last, you can save these results into a specified file.

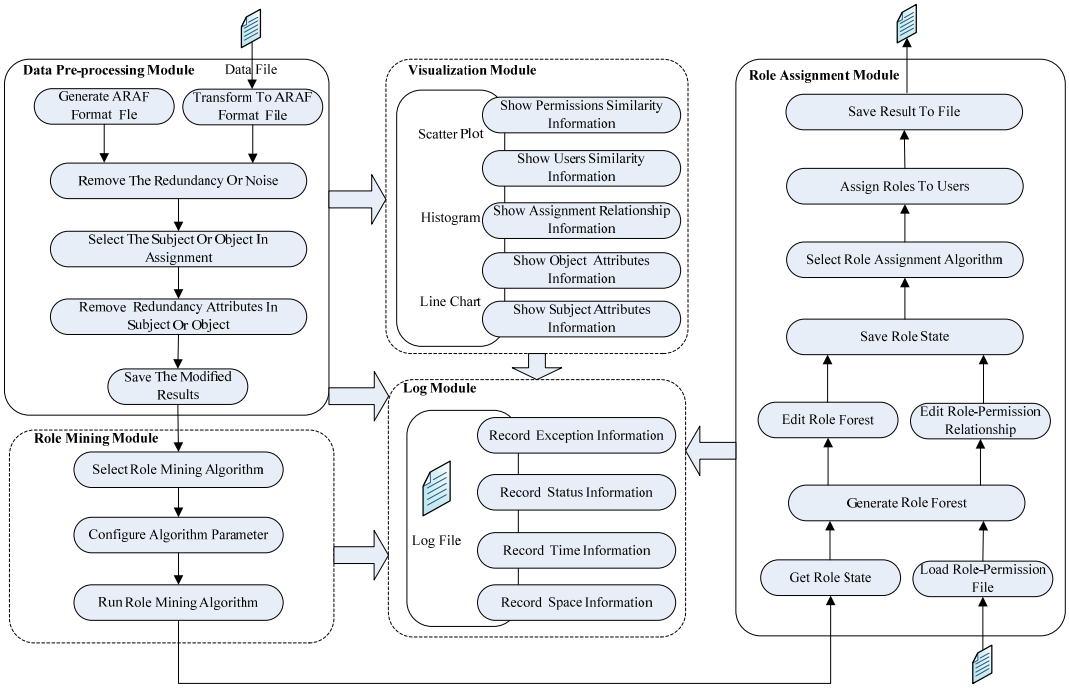


Figure 1: The framework of RMiner

Chapter 3

The Graphical User Interface

3.1 Launching RMiner

When you get the RMiner.zip files, you can decompression it. You will get a lot of files. Among them some files are important, for example, RMiner.jar and datasets folder. The RMiner.jar file is a boot file for the RMiner. The RMiner-src.jar file is source code file of the RMiner project, from it, you can see the source code, and you can modify it according to the way you want or you like. At last, datasets folder is sample data set which will be used in the project.

In a word, if you want to launch the RMiner, you click the RMiner.jar file twice. After doing so, you will see the graphical user interface shown in Figure 2.

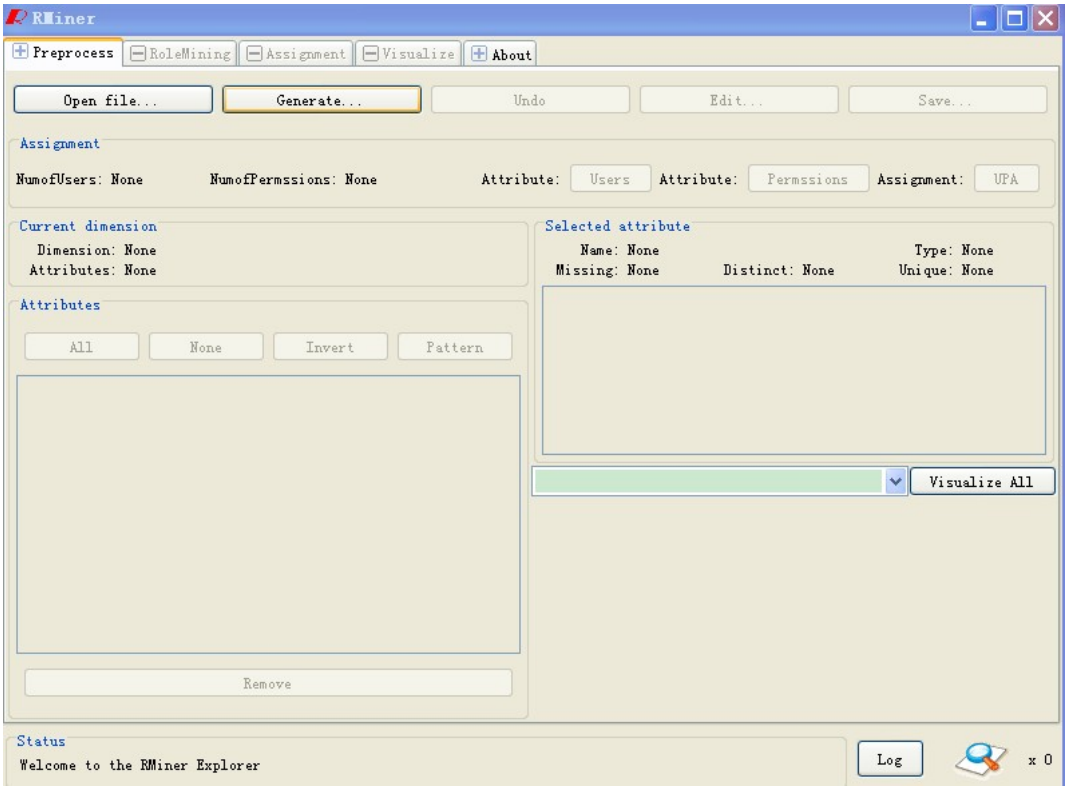


Figure 2: The main interface of RMiner

You must find that the interface is similar with WEKA's interface. That is right! Our objective is to let the system be more easily used, while the best way of achieving the objective is to use the familiar interface for most people at present. So, we design the interface based on the WEKA's interface.

The upper part of the main interface consists of five buttons, one for each of the five major RMiner applications, and five menus. These buttons can be used to start the following applications:

• **Preprocess** It is used for loading required data. In our project, it is three types of files: the *ass* file, the *ur* file and the *rp* file. These are put in the dataset folder, so if you add your own data, please put it into the folder.

• **RoleMining** Under the button, we implement eleven role mining algorithms. Eight algorithms among them are proposed by the counterparts and the remainders are our own algorithms. In the interface, you can select one of them to mine role set from the dataset selected by you in the first step.

• **Assignment** The interface provides role-assignment with these role set mined in second step and the use-permission assignment relation offered in the first step. Until now, we only implement one role assignment way called greed. That .is not only because the problem of role assignment is NP-hard and the research is few.

• **Visualization** Where, we use a visual interface for illustrating the users' similarity and the permissions' similarity. From the permissions' similarity, we can obtain an approximate role number in the dataset before the role mining. From the uses' similarity, we can preliminarily see the assignment relation on the uses' permissions.

• **About** When you click the menu, you will see a paper about the tool. In the paper, we at first give the intention of developing the tool, and then we also talk the data format processed by the tool, and at last we roughly write three parts of the tool.

The detailed process of each part is showed as follows.

3.2 Data Pre-processing

When you press down the preprocess button, you will get a file options dialog where you can select your processing file. In our project, you can pick up a file in the dataset directory. Of course, you can add a file according to the format requirement. The file type selected by you will influence the next behavior. If you select the ass file, next you can mine role and visualize. From the permissions' visualization, you will get the approximate number of the role set and then you can mine the role according to following step. If you pick up the *rp* file and *ru* file, it illustrates role set have already existed, so you don't need to mine the role set. And if you select the *rp* file, you need to assign the role permissions and edit role as you require (it will be discussed in the

following). At last, if you select the *ur* file, you don't need do the assignment work (it also is mentioned next). The only work is to assign every role permissions.

In the step, you can also see the users' attributes and permissions' attributes. The process is showed in the Figure 3. It is the result of the permissions' attributes by clicking the permissions button. Similarly, you can click the users button to get the users' attribute.

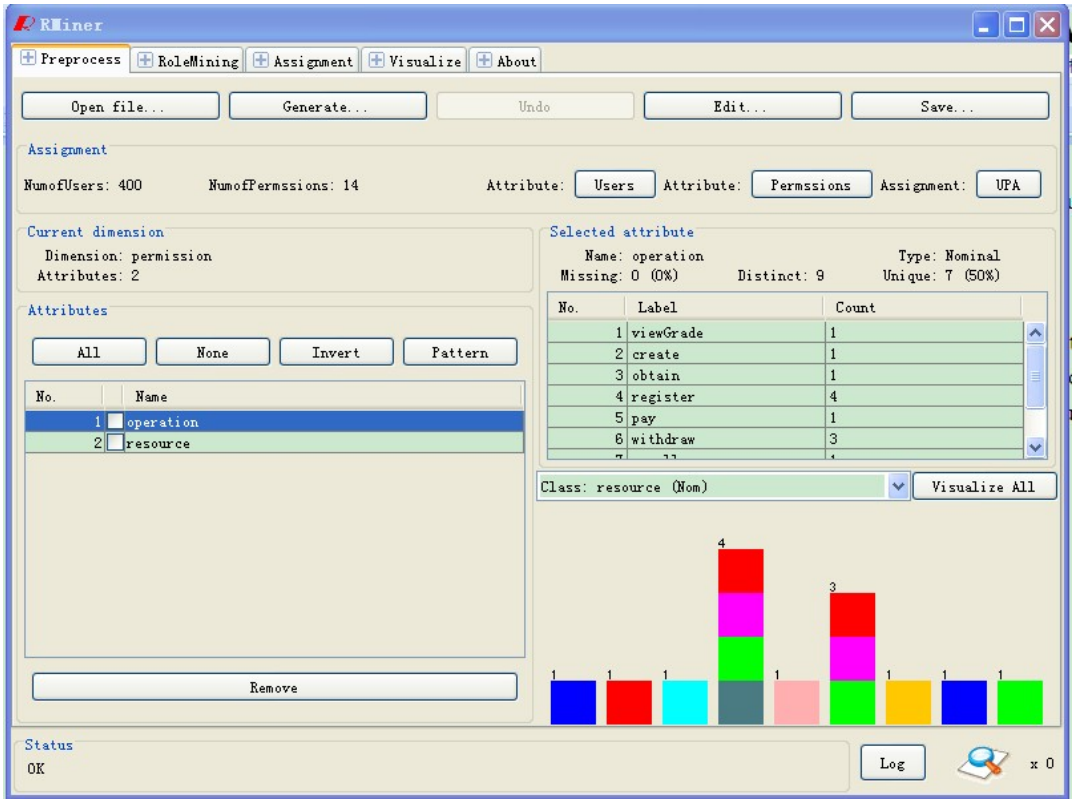


Figure 3: The permissions' attribute

From the figure, we know permission has two attributes which are the operation and the resource. That is to say, the permission can split into the operation and the resource, for example the see_book permission can split into the see-operation and the book-resource. The Figure 3 shows the result of the university_runningexample.ass. If we don't declare specifically, we will also use the file for illustrating the next function of the tool. Because the file has all not only these data which is needed in the next function and it displays very well.

In this step, it has many other functions. If you want to know the assignment information, you can click the UPA button. If you don't have user-permission assignment information, you can click the Generate button for generating random assignment information where you can appoint the user's number and the permission's number. If you want to modify the assignment information, you can press down the edit button. At last, you can save the assignment information which you just modify.

3.3 Role Mining Platform

This is a main function in the RMiner. When you select the ass format file in first step, you can mine role set in this step. If you press the RoleMining button, you can select any one role mining algorithm by clicking the choose button. In this place, we list thirteen role mining algorithms. Mining constraints in Role-Based Access Control called Anti-apriori [13] also been considered. Eight algorithms among them are proposed by the counterparts and the remainders are our own algorithms. Our algorithms are FeatureMiner [14], WeightedRoleMining [15], CCRMimplement ThesisAlgorithmImplement and YeHRMiner shown in Figure 4. After you pick up a role mining algorithm, you can click the start button to get the role set. Of course, whenever you want to stop the algorithm, you can press the stop button. The result of mining the role set on the dataset picked up in first step is showed in Figure 5. In the Figure 5, we select first the ORCA and then Hierarchical Miner. The main interface shows the result of the Hierarchical Miner.

From the figure, we can know all roles got by the Hierarchical Miner. We can get the number of the roles, the number of the edges and the number of the hierarchy. They are ten, four hundred and twenty-three (four hundred and five add seventeen) and eleven, respectively. So the final wsc (the weighted structural complexity) is four hundred and forty-three. Besides the above information, we can get the permission set and user set of every role.

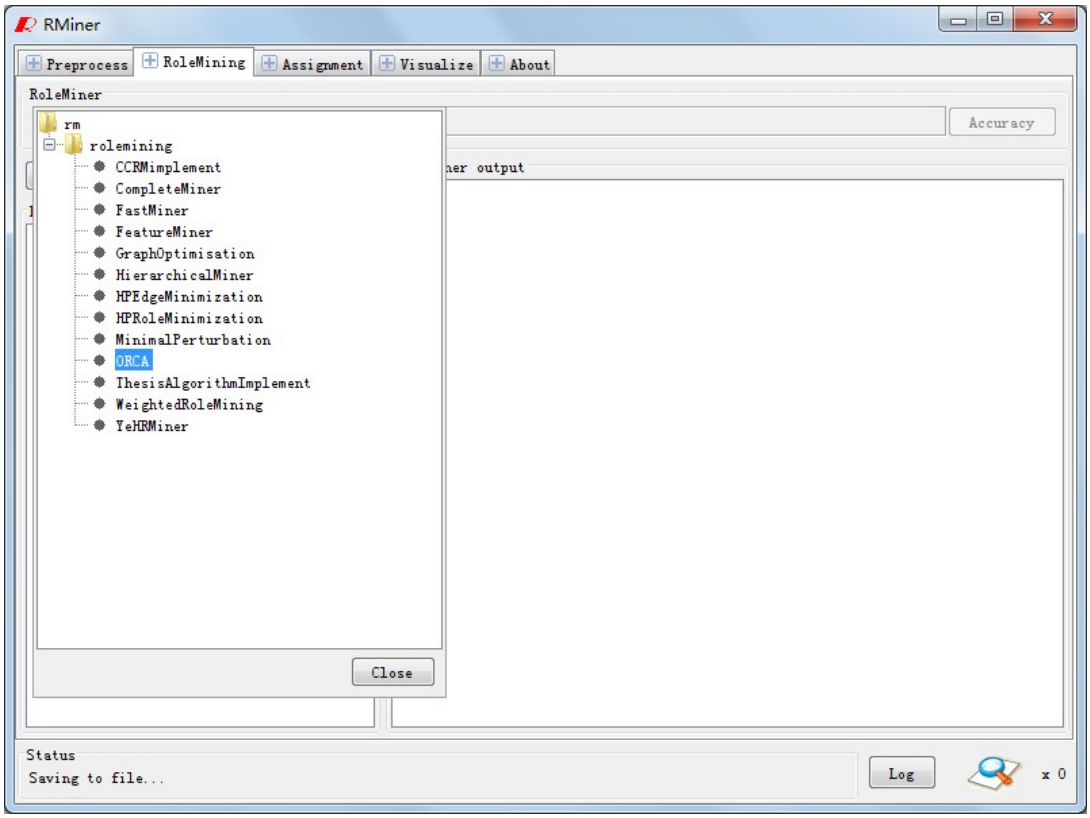


Figure 4: pick up role mining algorithm

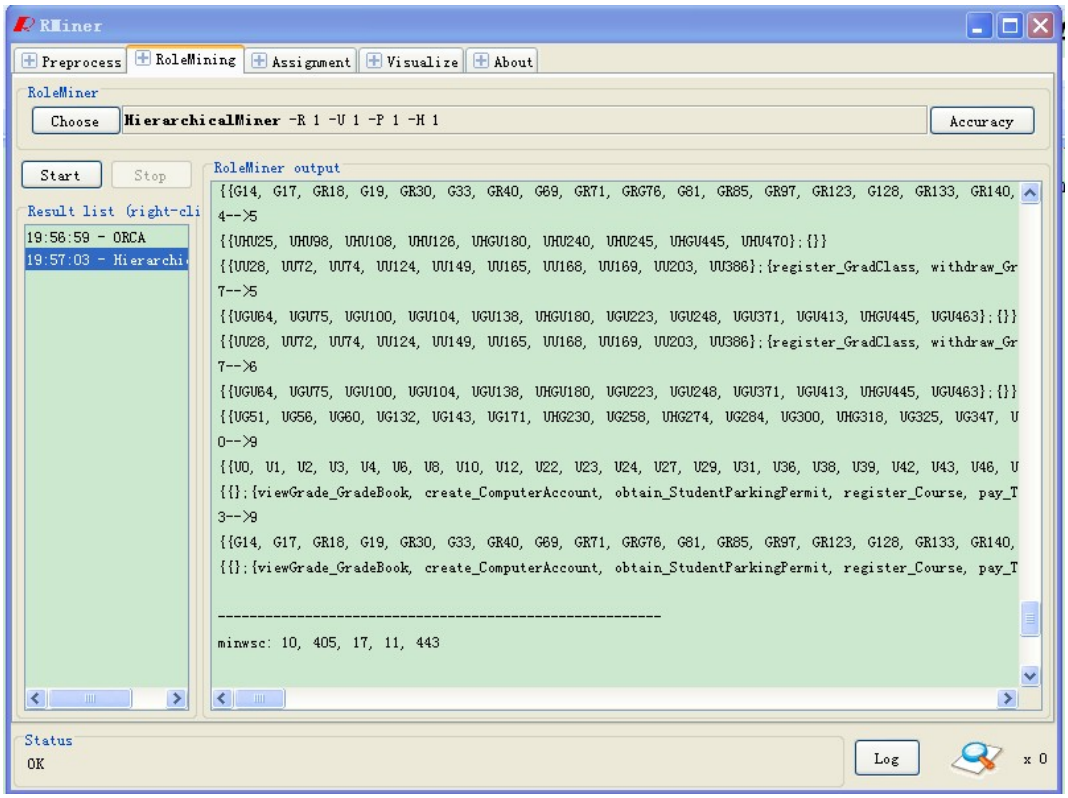


Figure 5: The result mining role set

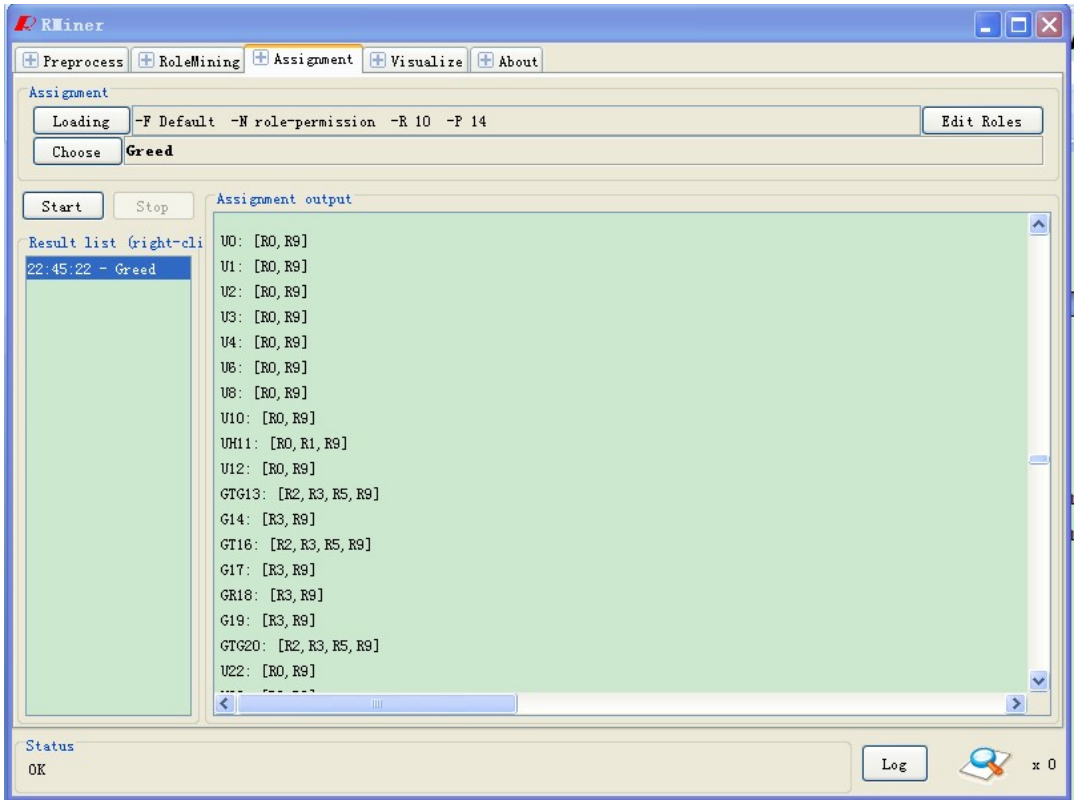


Figure 6: The assignment relation

In the left part of the Figure, we know that we run two algorithms which are ORCA and Hierarchical Miner to mine role set. If you click the highlight part with the right button of the mouse, you will see a menu which contains five items. You can save the role set found by any one algorithm as a file with last one of these menu items. With the interface, you can also calculate the accuracy if the ass file selected by you in first step contains presupposed roles. This process is finished through clicking the accuracy button. As you see, the bottom part of the interface records the work's state completely. If an error has occurred in your program, you can check the log and correct it with the log.

3.4 Role Assignment Tool

When you enter into the third property page, you have got all role set in the second step. You can assign every user some roles under the conditions that let the permissions of the roles just cover the permissions of user as much as possible. If there are many covered ways, we pick up the way that the number of the roles is the minimum. Because the problem itself is a NP-hard problem (the problem is equivalent to the problem that subset covers) and the research is little, we only implement a simply greed algorithms. Of course, you can add the assignment algorithms. When you select an assignment algorithm, you can click the start button to get the assignment way. The Figure 6 shows the assignment result with the role set found in second step and assignment relation in first step.

From the figure, we can clearly see the role set owned by every user. Besides you can use the role set mined with the way that is introduced in second step, you can also load the rp file directly when you click the Loading button. The right part of the Loading button shows the current relation which is role-permission relation and the number of roles and the number of permissions.

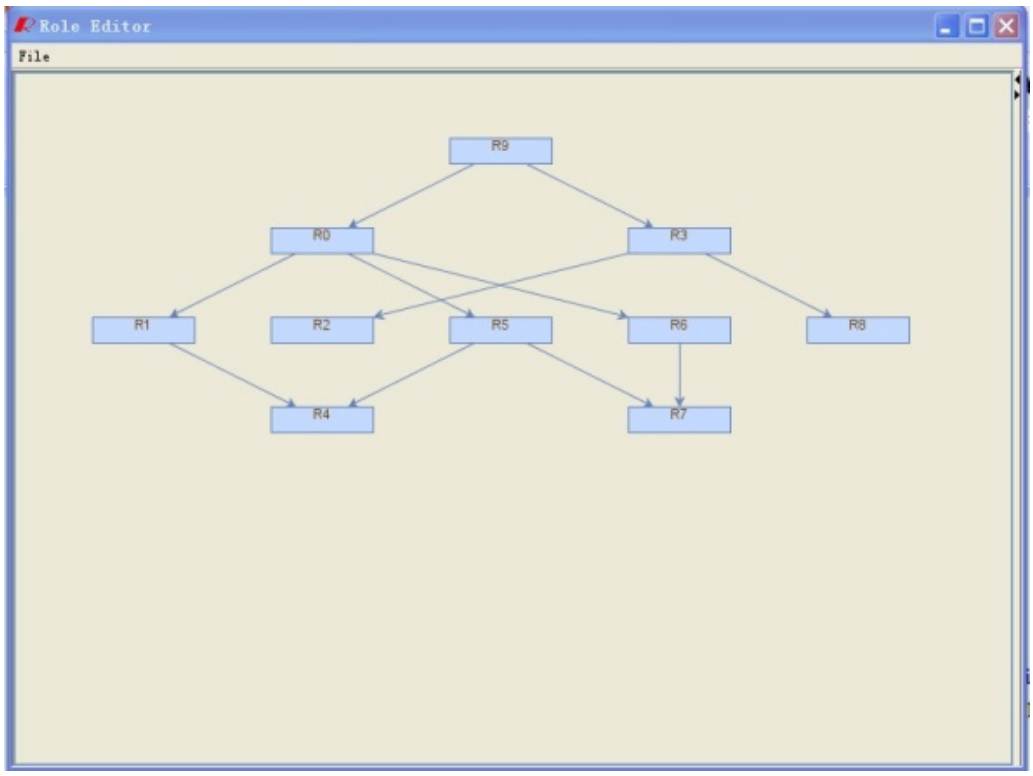
3.5 Role Editing Platform

If you aren't satisfied with the result of the role set which is got in second step [16], you can press down the Edit Roles button in the Figure 6. When you click the button, you can see the Figure 7. Figure 7 shows the result got in second step.

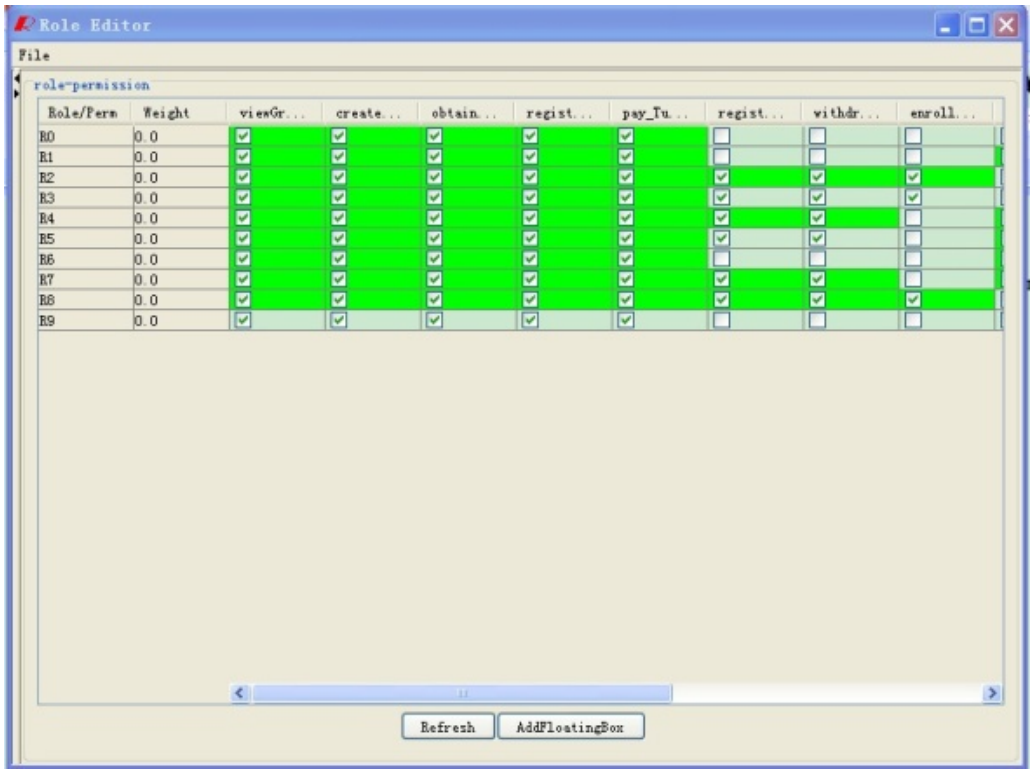
Figure 7 is divided into two parts, but they are together in fact. You can click the middle button to maximize (a) or (b). You can also drag the scroll to increase one of two parts. (a) is the hierarchy of the roles set found by RoleMing step. Thought this visual interface, you can clearly know the hierarchy which maps the superior-subordinate relationship in real unit. (b) shows the permission set of every role and the green part is permissions inherited from his parents. The white part is its own permissions. If you feel that the role set got in second step doesn't meet your needs, you can modify the checked box of every permission's selected state. When you finish your modification, you can click the refresh button in the bottom. After pressing down the refresh button, you can see the hierarchy changes immediately in (a).

The tool is very useful for the researchers working in the field of the RoleMing and is also

addition part comparing to the WEKA.



(a)



(b)

Figure 7: The role editing interface

3.6 Visualization Tool

The fourth property page is about the visualization. The aim of the tool is to evaluate the similarity between uses and the similarity between permissions before mining roles. Especially, through the similarity of permissions, we can get the approximate number of roles before mining role. If you only want to know the number of roles and it needs too long time to mine roles, you can only use the function and don't need to use the tool of the RoleMining tool mentioned in second stage. The result of permissions' similarity is showed in Figure 8.

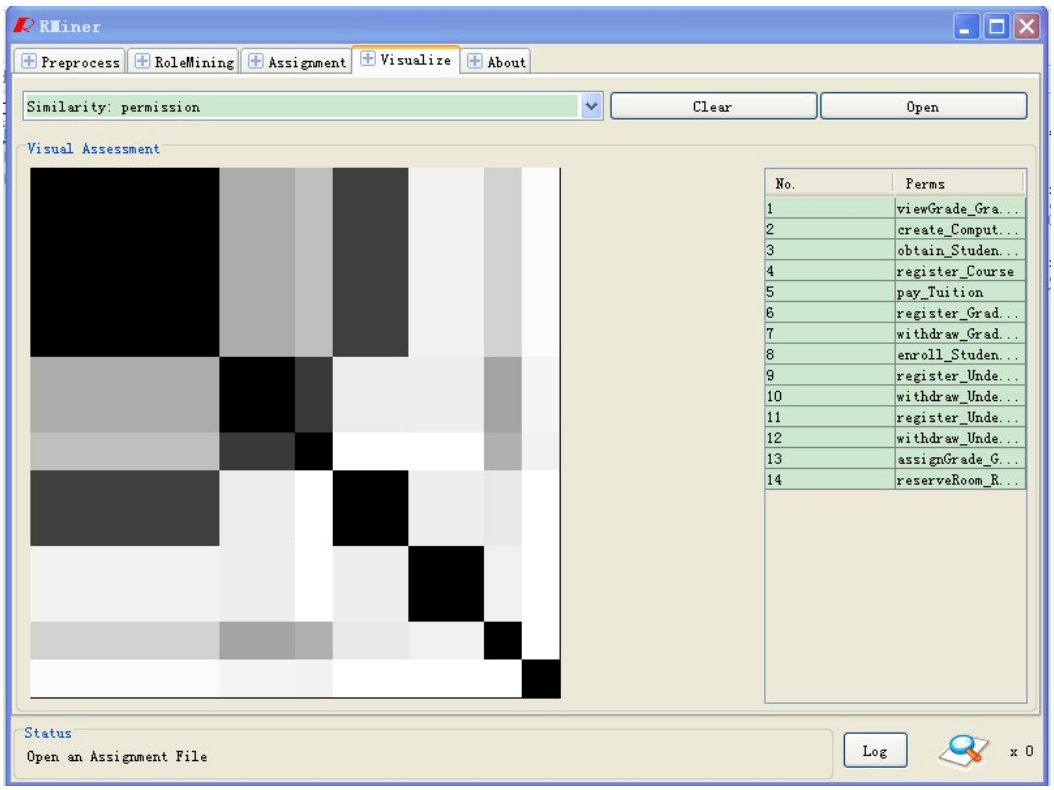


Figure 8: The similarity between permissions

Because vertical and horizontal axis both is permission, we get a symmetry graph. If you analyze the number of roles, you must use only upper or lower triangular of the graph. In the graph, each kind deep color represents a role. The deeper the color is (the more the pixels are), the more obvious the role is. For instance, the black in the upper left corner of the graph is very good role, moreover, the role has a lot of permissions. If some color is too dark, we don't think it is a role. In Figure 8, we can get nine roles. It is very close the result got by Hierarchical Miner which is ten.

The right part of the graph shows all these permissions. You can modify any one of these permissions with clicking the corresponding permission twice. After modifying, the result will be showed on the left part immediately. If you want to load other data, you can first clear the current data by pressing down the clear button, and then you can load your own data with clicking the open button. At last, you can also see the similarity between uses. The Figure 9 shows the relation. Because the users are too many in the dataset selected in first step, the graph looks very irregular. We pick up other dataset where the users are fewer. In fact, there are only six users.

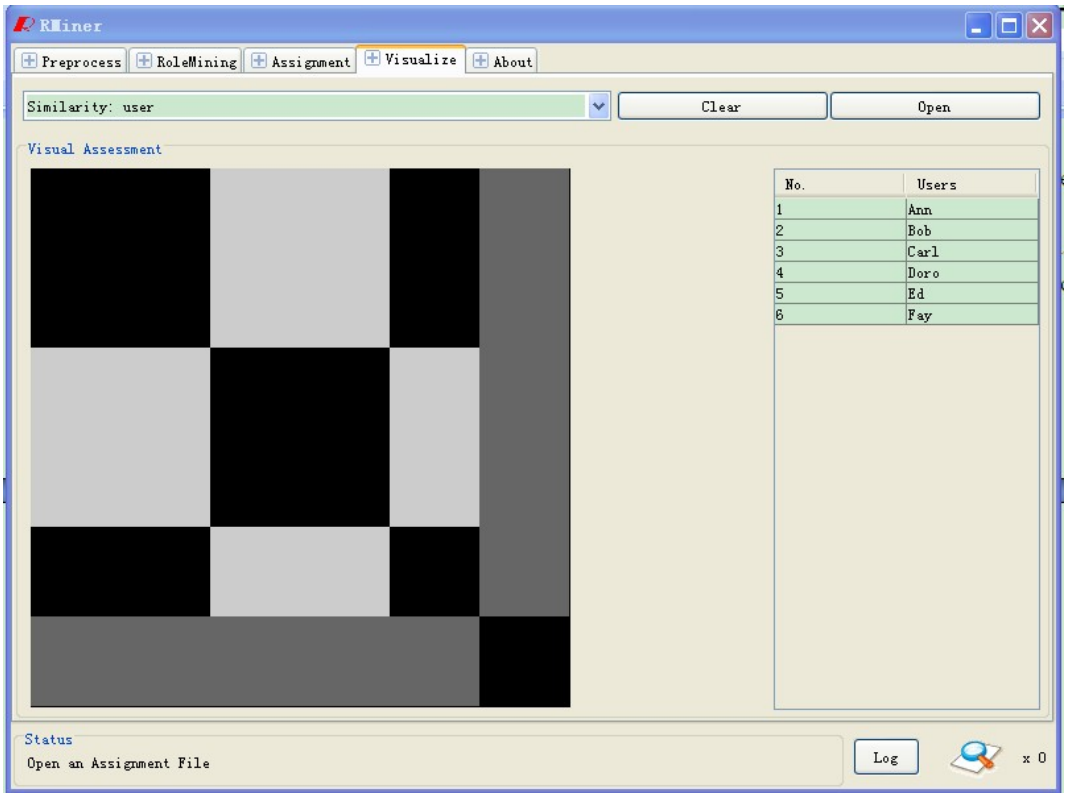


Figure 9: The similarity between uses

The graph is also symmetrical. The reason is same as the front part. From the graph, we can users' permissions' extent of the similarity. The deeper the color is, the more similar two users are.

3.7 About Information

It is a rough introduction about the product. When you click the menu, you will see a paper about the tool. In the paper, we at first give the intention of developing the tool, and then we also talk the data format processed by the tool, and at last we roughly introduce three parts of the tool.

Chapter 4

Adding Your Own Algorithm

4.1 Get the Source Code

If you want to know the details about these algorithms, you can get the source code by unzip the RMiner-src.jar file in the dataset directory. After importing the unzipped file to an integrated development environment, you can get the project which is showed in Figure 10.

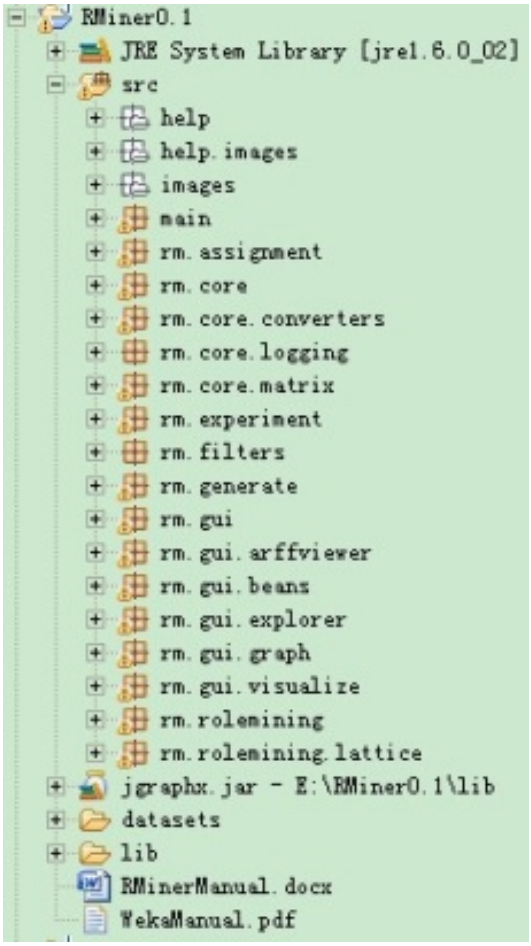


Figure 10: The project directory

In the project's src directory, there are all packages needed by our algorithms. Among these packages, those packages whose names contain core string are the core functions. Those packages whose names contain gui string are about the interfaces. In the rm.rolemining package, those are a lot of rolemining algorithms set and the rm.rolemining lattice package contain many data structures which are assisted the rm.rolemining package to implement those roleming algorithms. At last, the other packages in the src directory are used to assist other functions. When you want to launch the project, you can run the Explorer class in the rm.gui.exploer package.

4.2 Add Your Own Algorithm

If you want to add your own algorithm and do the control experiment, you can make it as follows:

Firstly, you must add your algorithm's name ending with a comma and a backslash character which looks like rm.rolemining.ORCA,\ in the GenericObjectEditor.props file which is in rm.gui package.

Secondly, you must implement your algorithm in the rm.rolemining package and put required data structures into the rm.rolemining lattice package or create a new package whose name starts with rm.rolemining to put these data structures. When you implement your own algorithm, your class

must inherit AbstractRoleminer class and implement OptionHandler interface. At last, you can run the Explorer class in the rm.gui.exploer package.

That is all. After you do these according to above three steps, you will see that your own algorithm appears in the RoleMining property page. And then you can make the control experiment with the platform.

Bibliography

[1] Remco R. Bouckaert, Eibe Frank, Mark Hall, Richard Kirkby, Peter Reutemann, Alex Seewald. and David Scuse. (2009) WEKA Manual for Version 3-7-0. University of Waikato, Hamilton, New Zealand.

[2] WEKAWiki – http://WEKA.wiki.sourceforge.net/

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten. The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter, Volume 11, Issue 1, pp.10-18, 2009.

[4] Extensions for WEKA's main GUI on WEKAWiki– http://WEKA.wiki.sourceforge.net/Extensions+for+WEKA%27s+main+GUI

[5] Adding tabs in the Explorer on WEKA Wiki –http://WEKA.wiki.sourceforge.net/Adding+tabs+in+the+Explorer

[6] Explorer visualization plugins on WEKAWiki –http://WEKA.wiki.sourceforge.net/Explorer+visualization+plugins

[7] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo. Evaluating role mining algorithms. In Proceedings of the 14th ACM Symposium on Access Control Models and Technologies, pages 95-104, June 2009.

[8] M. Frank,J. M. Buhmann, and D. Basin. On the Definition of Role Mining. In Proceedings of the 15th ACM Symposium on Access Control Models and Technologies, pages 35-44, June 2010.

[9] J. Schlegelmilch and U. Steffens. Role mining with ORCA. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT), 2005.

[10] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino,S. Calo, and J. Lobo. Mining roles with semantic meanings. In Proc. ACM Symposium on Access Control Models and Technologies (SACMAT), 2008.

[11] Jaideep Vaidya, Vijayalakshmi Atluri, Vijayalak -shmi Atluri. RoleMiner: Mining Roles using Subset Enumeration. CCS'06, October 30–November 3, 2006, Alexandria, Virginia, USA.

[12] Ruixuan Li, Huaqing Li, Wei Wang, Xiaopu Ma, Xiwu Gu. RMiner: A Tool Set for Role Mining. The 18th ACM Symposium on Access Control Models and Technologies (SACMAT 2013), Amsterdam, Netherlands, Jun 12-14, 2013

[13] Xiaopu Ma, Ruixuan Li, Zhengding Lu, Wei Wang. Mining Constraints in Role-Based Access

Control. Mathematical and Computer Modelling, Elsevier, 2012, 55(1-2): 87-96

[14] Ruixuan Li, Wei Wang, Xiaopu Ma, Xiwu Gu, Kunmei Wen. Mining Roles using Attributes of Permissions. International Journal of Innovative Computing, Information and Control, 2012, 8(11): 7909-7924

[15] Xiaopu Ma, Ruixuan Li, Zhengding Lu. Role Mining Based on Weights. The 15th ACM Symposium on Access Control Models and Technologies (SACMAT 2010), Pittsburgh, PA, USA, June 9-11, 2010, pp. 65-74

[16] Jinwei Hu, Yan Zhang, Ruixuan Li, Zhengding Lu. Role Updating for Assignments. The 15th ACM Symposium on Access Control Models and Technologies (SACMAT 2010), Pittsburgh, PA, USA, June 9-11, 2010, pp. 89-98