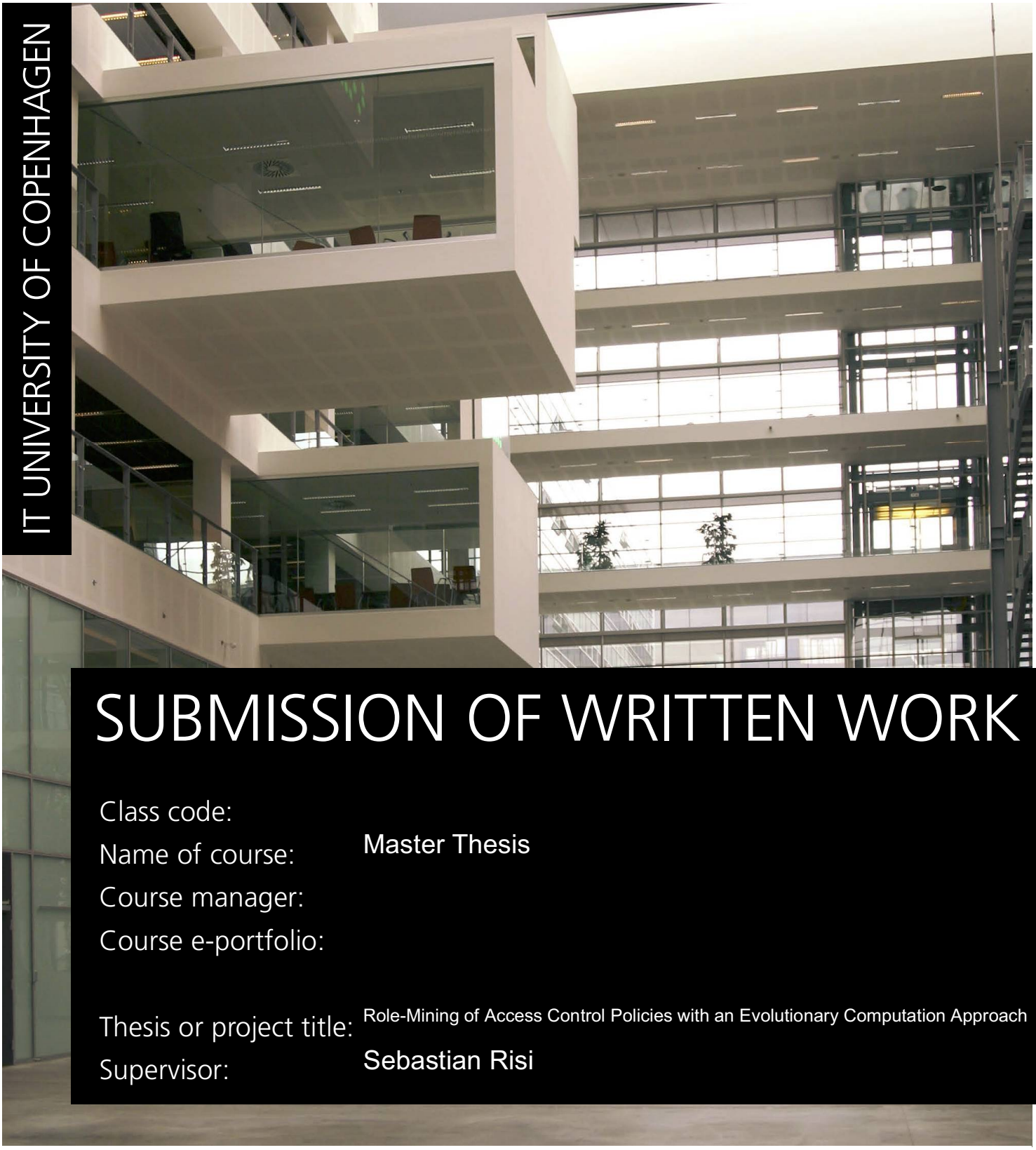
IT UNIVERSITY OF COPENHAGEN



SUBMISSION OF WRITTEN WORK

Class code:

Name of course: Master Thesis

Course manager:

Course e-portfolio:

Thesis or project title: Role-Mining of Access Control Policies with an Evolutionary Computation Approach

Supervisor: Sebastian Risi

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Theresa Ragna Elisabeth Brandt von Fackh	17/10-1984	tebr@itu.dk
2.			@itu.dk
3.			@itu.dk
4.			@itu.dk
5.			@itu.dk
6.			@itu.dk
7.			@itu.dk

IT UNIVERSITY OF COPENHAGEN

MASTER THESIS

SOFTWARE DEVELOPMENT AND TECHNOLOGY

Role-Mining of Access Control Policies with an Evolutionary Computation Approach

Author:
Theresa Ragna Elisabeth
BRANDT VON FACKH

Supervisor:
Sebastian
RISI

1. December 2015

Abstract

Role-based access control (RBAC) is the most used access control model nowadays due to the advantage to lower the complexity and costs of access control administration. One major task in RBAC is the role engineering, the creation of a Role Model. Different algorithms are applied on current assignments of users to permissions in order to obtain a role model in reasonable time. From this the term Role-Mining was derived. While research in role mining is mainly based on data mining approaches, very little research has been done on role mining with an evolutionary computation approach. In this thesis an evolutionary approach to evolve an RBAC role model from a given access control policy towards an optimal solution is investigated. An evolutionary algorithm for role mining based on a simple evolutionary algorithm and on a multi-objective evolutionary algorithm is developed. Several role mining objectives are investigated and tested on the developed algorithms.

Contents

Chapter 1

Introduction

Access control policies are maintaining security policies as direct assignments from access rights (permissions) to identities (users, computers or other principals). In order to lower the complexity and cost of access control administration, access control models were introduced. The most used access control model in the recent years in enterprise identity management systems is the Role-based Access Control (RBAC)[23]. RBAC security policies are grouping permissions into roles, which then are assigned to identities. Permissions can be contained in several roles and users can be assigned to several roles.

One major task of RBAC is the role engineering, the creation of a role model, which determines which permissions are grouped together to roles and to whom are they assigned. Furthermore how many roles and assignments are needed, to cover the current access control policies. In enterprises the amounts of users and permissions can be enormous, which made the usage of data mining or artificial intelligence techniques for this task popular. The aim is to discover a good role model, which covers the current access control policies and effectively lowers the complexity and cost of access control administration. These processes of obtaining role models with the help of data mining or artificial intelligence techniques is called role mining.

In this thesis the use of an evolutionary computation approach for role mining is investigated.

1.1 Motivation

Role mining is an optimization problem, which seeks for an optimal role model. Selecting the criteria for its optimization is still unsolved. There are several quality measures for an RBAC model, which can be dependent on the enterprises policy. Clearly the role mining optimization is following several objectives, for example minimizing the number of roles and minimizing the number of confidentiality and availability violations (over- and underentitlements), which can be in conflict with each other.

With multiobjective evolutionary algorithms pareto-optimal solutions, a set of optimal trade-off solutions, can be found in a single run instead of several runs, which is necessary for some of the conventional stochastic processes, like simulated annealing and tabu search[1]. Previous research in the role mining field are using scalarisation, where several objectives are weighted and summarized in one optimization function, like the Weighted Structural Complexity (WSC) in Molloy et al.[28] or the fitness functions in Saenko & Kotenko[35]. The trade-offs between the objectives are not apparent in these approaches.

There are constraining policies existing, such as Separation of duties (SoD), which should prevent that certain permissions are assigned to a user at the same time to avoid fraud. Role mining algorithms might not take this into account and the resulting role model is corrected in a post-process by domain experts. Evolutionary algorithms can include constraint handling by for example penalize role model solutions, which do not comply to the policies.

The resulting role models of role mining often do not have any business meaning in that sense that there is no logical business description for the roles. Consequently, the role model does not get adopted by managers and system administrators, which are responsible for the correct assignment of these roles. Several workshops with domain experts are needed to transform the mined role model into a realistic role model, which is accepted by the business. One approach is to include business information into the role mining algorithm. But first relevant and usable business information has to be identified and collected. Furthermore a measure for the ”meaningfulness” of roles has to be identified.

Another approach could be to include human expert knowledge to a role mining

algorithm. Interactive evolutionary computation (IEC)[38] could accomplish human feedback to a role mining algorithm based on evolutionary computation to navigate the algorithm to an optimal result, accepted by the business.

Another challenge in RBAC is the maintaining of the role model. Permissions, users and business information are evolving over time and the initially created role model might become suboptimal over time. There is a practical need for periodic quality assessment of the resulting role models[23]. A complete re-modelling would not be a feasible solution, since the system administrators and managers would not accept if the role model is regularly restructured. A solution which slowly evolves the role model with the occurring changes has to be designed.

Evolutionary algorithms can adapt solutions to changing circumstances[11]. A current role model state can be used as starting population and the evolutionary algorithm will optimize towards the formulated objectives.

There are several arguments, which are motivating to use an evolutionary computation approach for role mining, but so far this approach has not been researched very widely. Only few research (Saenko & Kotenko [34][35]) could be found, which started to investigate to mine a role model with an evolutionary algorithm.

In this thesis the suggested evolutionary algorithm by Saenko & Kotenko is implemented as a starting point, where the not mentioned parts of the algorithm are intuitively filled. The algorithm is tested on synthetic and real datasets commonly used in role mining research. Furthermore multi-objective evolutionary algorithms are implemented and tested. Challenges of the approach are pointed out. Another topic investigated is to include business information data into the role mining algorithm.

1.2 Thesis question

How can a role model be mined from scratch with an evolutionary computation approach and how adequate are the results?

1.3 Roadmap

The remainder of this thesis report is organized as follows. The thesis starts with an introduction to the domain of RBAC and Role Mining (Chapter 2). Afterwards an introduction to evolutionary algorithms is given, where the basic concepts are explained and a description of algorithms is given (Chapter 3). With the background information to the domain and evolutionary algorithms in mind the related work is described in Chapter 4. In Chapter 5 the approach and implementation is described and decisions are discussed. The experiments and according results can be read in Chapter 6. A discussion on the results and the approach is given in Chapter 7. Chapter 8 gives an outlook for future work. At last a conclusion is given in Chapter 9, where the work is summarized and the most important findings are highlighted.

Chapter 2

Domain and Problem

In this section a basic introduction to the domain is given. It is important that the motivation and concepts of the domain are known in order to scope the problem definition and for guidance in the implementation of the approach.

Proof-read chapter 2

2.1 Access Control Models

Access control ensures through policy definitions and enforcement that users[1] can only access resources (e.g. files, applications, networks) they are authorized to. The policy definitions describe which user is authorized to which permission. A permission describes who has access to resource and which action (e.g. read, write, execute) can be done on a resource.

There are business, security and regulatory drivers for managing access control policies [32]. The interest of the business is to lower the costs for managing the permissions for employees and quickly equip employees with necessary permissions such that they can efficiently fulfill their tasks. The security driver is to ensure information confidentiality, integrity, and availability to prevent accidental and intentional security breaches. But also to be compliant with regularities, such as the Data Protection

[1] For simplicity the term ”User” is used throughout the thesis although the term ”subject” would be more general to also include e.g. service accounts.

Directive in Europe (Directive 95/46/EC)[2], Basel II[3], Sarbanes–Oxley Act (SOX)[4] or company regulations, have an impact on the access control policies.

The National Institute of Standards and Technology (NIST) [19] describes various logical access control models, which provide a policy framework that specifies how permissions are managed and who, under what circumstances, is entitled to which permissions and enforce these access control decisions. Some of these logical access control models are briefly described in the following to enable the reader to differentiate between the concepts.

Identity-based Access Control (IBAC)

In IBAC a user gets a certain access to a resource by being assigned directly to a permission, which is connected to a resource. On a low-level Access Control Lists (ACLs) are implementations of IBAC[5]. While IBAC may be manageable in companies with a small amount of users and permissions, the maintenance can be quickly overwhelming in consideration of satisfying all drivers of access control policies (see above).

Role-based Access Control (RBAC)

In RBAC permissions are bundled to roles, which then are assigned to users. A user gets a certain access to a resource by being assigned to a role, which contains the corresponding permission to that access. In other words users inherit all permissions of the roles they are assigned to. The motivation of this extra abstraction layer of a role is to easier maintain the access control of users. Even more abstraction can be introduced by role hierarchies (see section 2.2.2). The RBAC model, which contains the roles, the user assignments to roles, the permission assignments to roles and the role-hierarchy, needs to be defined before the access control mechanism can enforce the access control. The RBAC model can be an ease of administration in comparison to direct assignments in IBAC. The degree of the benefit is dependent on the RBAC

[2] `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML` (07-10-2015)

[3] `https://en.wikipedia.org/wiki/Basel_II` (07-10-2015)

[4] `https://en.wikipedia.org/wiki/Sarbanes-Oxley_Act` (27-11-2015)

[5] In Windows OS the ACLs can also contain groups

model (see section 2.2).

Attribute-based Access Control (ABAC)

In Hu et al.[19] the authors try to guide to a standard definition of ABAC, since there seems to be no consensus. The basic concept of ABAC is to introduce an additional abstraction layer in form of ABAC policies, which are basically complex boolean rule sets that can evaluate different kinds of attributes. These attributes could be user information (e.g. department, job title), resource information (e.g. threat level) or environment information (e.g. current time, current location). A user gets a certain access to a resource if the rule set is satisfiable. The rule sets need to be defined before the access control mechanism can enforce the access control. Compared to RBAC ABAC seems more flexible, but it also comes with challenges regarding risk and auditing[4].

The scope of this thesis is to research an approach which outputs an RBAC model. The RBAC model is often used in bigger organizations[32] and is leveraged by many Identity- and Access Management (IAM) systems. Some ideas of the ABAC model are exploited to some extent in the implementation of the thesis approach. In praxis many companies use a combination of ABAC and RBAC.

2.2 RBAC Reference Model

There are different functional capabilities of RBAC models, which are used to distinguish different RBAC models. In Sandhu et al.[37] four conceptual RBAC models are distinguished, which will be introduced in the following.

2.2.1 $RBAC_0$ - Base Model

The base model describes the minimum requirement for an RBAC model. The minimum requirement is that roles are bundles of permissions and users are assigned to these roles in order to get the according permissions. A user can be assigned to several roles and roles can be assigned to several users. The same many-to-many relation exists between roles and permissions. Therefore a user can get the same permission

several times via different roles. Figure 2.1 demonstrates the relation between users, permissions and roles.

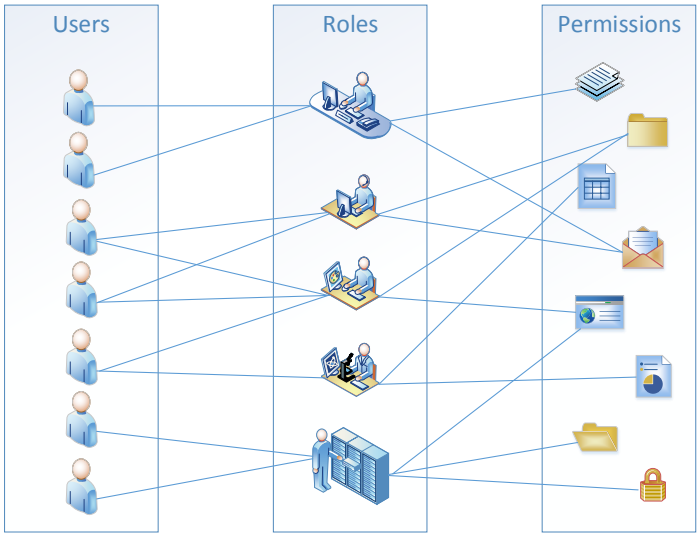


Figure 2.1: The relation between users, permission and roles in RBAC.

2.2.2 $RBAC_1$ - Role Hierarchies

The $RBAC_1$ model is based on $RBAC_0$ and introduces the concept of role hierarchies. In a role hierarchy roles can be assigned to roles and inherit their permissions to the roles they have been assigned to.Generally a role hierarchy is a directed acyclic graph, where a role inherits the permissions from any other junior-role, which is assigned to it. It is also possible to limit inheritance in role hierarchies to control the power of impact of roles. In figure 2.2 shows one example of a role hierarchy. In Moffett[27] role hierarchies and how they can be in conflict with some control principles, such as the Separation-of-Duty (SoD) principle, decentralisation, supervision and review, are discussed.

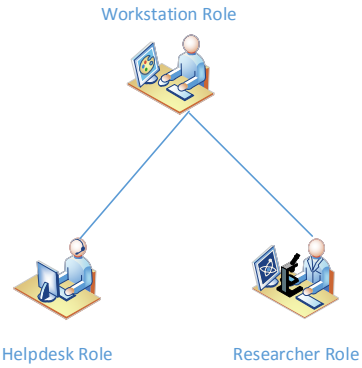


Figure 2.2: Example of a role hierarchy: The Helpdesk Role and the Researcher Role are inheriting all the permissions from the Workstation Role and have some additional permissions.

2.2.3 $RBAC_2$ - Constraints

The $RBAC_2$ model is based on $RBAC_0$ and introduces the concept of constraints. With the help of constraints Separation-of-Duty (SoD), also known as Segregation-of-duty, can be achieved. SoD is a security principle for preventing fraud and errors by splitting tasks and associated permissions among multiple users. For example a user who has the permission to request the purchase of goods or services should not have the permission of approving the purchase. Roles in an RBAC model should therefore not have permissions bundled, which violate each other due to SoD. But also user assignments to multiple roles should not violate the SoD constraints.

2.2.4 $RBAC_3$ - Consolidated Model

The consolidated model combines $RBAC_1$ and $RBAC_2$ and therefore supports role hierarchies and constraints. Bringing these two functional capabilities together come with additional functionality. Constraints can also be applied on the role hierarchy. For example, the number of parent roles for child roles can be limited or different child roles can be constrained to have different parent roles. This type of SoD is static. A dynamic SoD prevents the availability of permissions to a user, who is currently using

a permission, which is not allowed to use in combination with the other permissions at the same time.

Models	Hierarchies	Constraints
$RBAC_0$	No	No
$RBAC_1$	Yes	No
$RBAC_2$	No	Yes
$RBAC_3$	Yes	Yes

Figure 2.3: RBAC Definitions[39]

The relation of the four introduced models can be seen in figure 2.3. In this thesis the main focus will be on the $RBAC_0$ model since it forms the basis of the other models. The $RBAC_0$ model is often a starting point for many companies. Like in Vaidya et al.[41] and Frank et al.[15] sessions will not be considered for simplicity reasons. The NIST RBAC model [36] distinguished between four levels of RBAC models: Flat, Hierarchical, Constrained and Symmetric RBAC. Each level introduces an additional functional capability to the previous level. In terms of the NIST RBAC model the Flat RBAC model will be the focus of the thesis.

2.3 Role Model Quality

A basic role model consists of users, permissions, roles, user-role assignments and role-permission assignments. A role model is most desirable if it has a state, which best leverages the RBAC model by satisfying the business, security and regulatory drivers: Lowering the costs for the administration of access control and equip employees with necessary permissions such that they can efficiently fulfill their tasks while keeping security requirements and being compliant.

How this goal is measured in the role model has been described by several quality criteria[23][16]. The criteria concern the role model state, individual roles or both. In this thesis the different criteria are summarized into the categories: Completeness, complexity and comprehension.

2.3.1 Completeness

The completeness measure is the mostly used measure in role mining[23], for example it is used in Xu & Stoller[44] and Vaidya et al.[41]. Completeness is the degree to which the role model fulfills the existing user-permission assignments. When the user-role and the permission-role assignments of the role model are resolved, it should cover all current user-permission assignments. In figure 2.4 an example is shown where user-permission assignments are split into user-role and the permission-role assignments. Resolving an role model would be the other way around - from user-role and permission-role assignments to user-permission assignment.

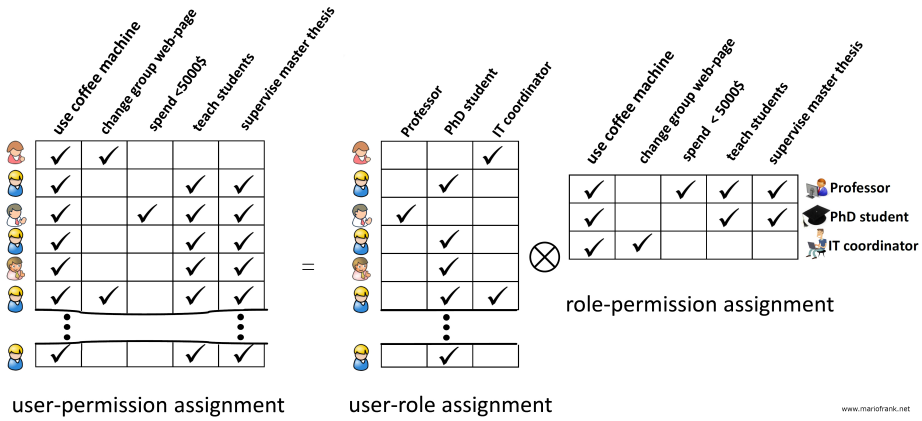


Figure 2.4: University toy example taken from Frank[14], which demonstrates the transformation from user-permission assignment(UPA) to user-role assignments(UA) and permission-role assignments(PA)

It is assumed that the current user-permission assignments are in a state, where users get the necessary access to efficiently perform their tasks and no security or compliance regulations are violated. This of course is an ideal situation, but in reality the current state often has quite some noise, especially when no IAM solution has been in place before. Users tend to have more permissions than they actually need, since it is unlikely that someone will claim that he has too many permissions. The measure of completeness of the role model state is therefore in accordance to the initial access control configuration.

A certain amount of overentitlements (users get too many permissions) or underen-

titlements (users get too few permissions) in the access control policies can be acceptable, if the least privilege principle is too costly to implement in practice. In figure 2.5 an example of over- and underentitlements can be seen. Further it should be noted that the measure of completeness is only a formal approach with no semantic meaning.

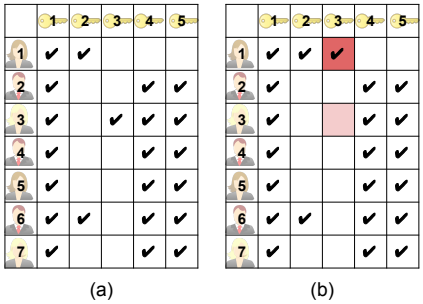


Figure 2.5: (a) Original user-permission assignments. There is a user for each row and a permission for each column. A check-mark represents the assigment of a permission to a user. (b) Resolved role model into user-permission assignments. In comparison to the original user-permission assignments one overentitlement (User 1 - Permission 3) and one underentitlement (User 3 - Permission 3) can be determined.

The combinations of permissions within roles or the combination of user-role assignments might violate security regulations such as SoD. This measure is therefore not only taking individual roles but also the role model state into account.

Constraints could also be ensured in a mechanism posterior to the role model, where individual permissions are detracted from users again, if it violates a constraint. Allowing constraint violations in the role model increases not only the processing and reliance on the posterior mechanism but also the auditing effort.

2.3.2 Complexity

The complexity of a role model is measured in its number of roles and the number of user-role and permission-role assignments. It is often connected to the maintenance costs of a role model. For example in figure 2.5 three roles, eight user-role assignments and nine role-permission assignments can be counted.

The more roles the role model has, the more maintenance effort is expected. Hence, a minimal set of roles is preferable. Furthermore it should be obvious that the total number of roles should be smaller than the total number of users or total number of permissions. Otherwise there would be no use of the advantage of having RBAC in comparison to IBAC in terms of administration costs. When each user has its individual role with the according individual bundle of permissions, the abstraction layer of the role becomes obsolete.

Also the more user-role and permission-role assignments are needed, the more maintenance effort is expected. Large roles with many permissions may reduce the number of user-role assignments, but may lead to more confidentiality/accessibility violations (conflicting Completeness). The same applies for if each role is used by many users. Small roles with few permissions on the other hand can lead to more administration effort as mentioned above, since many roles are necessary for achieving completeness. The same applies if each role is only used by very few users.

A role model can consists of very general large roles (e.g. a role for an organizational unit) and specialized small roles (e.g. a project-specific role). Determining a fix boundary of how many permissions or users can be assigned to a role requires knowledge of the role model or is given by company or security regulations.

2.3.3 Comprehension

A trending topic is to respect the ”meaningfulness” of roles in a role model[44][16]. It is important that the administrators, who are maintaining the role model, can logically understand the role model for maintaining the roles and assignments confidently. Otherwise it might happen that they avoid to work with the role model since they feel not confident to stay in line with security and compliance regulations. Or it will cause them extra effort and costs to work with the role model. The roles should be therefore comprehensive, which can be achieved by giving them a meaning close to business roles, e.g. a role ”Employee”, which contains all permissions every employee will get and is assigned to every employee-user.

This criteria can loosen some of the other criteria, which rather concentrate on the compression of the access control information [15]. For achieving more intuitive mean-

ingful roles it might be necessary to allow more roles than the minimal number of roles resulting from compression. More roles might result in more assignments, which are necessary to keep the role model more comprehensive. Lowering costs by having a more comprehensive role model may contradict lowering costs by having a less complex role model.

2.4 Role Engineering

Role engineering [5] describes the process to create a role model for RBAC. This task is proved to be very difficult in large enterprises.

There are three different approaches to conquer the goal of finding the right role model: Top-Down, Bottom-Up and the Hybrid approach[5][16].

Top-Down Approach

One approach is to do a top-down analysis, where the roles are built out of business information. Business processes, business roles and security policies are analysed to build a suitable role model. The resulting roles contain high-level permissions, which make sense to the business, but have not necessarily a technical complement. Therefore the high-level permissions need to be mapped to technical permissions that are used by IT systems. The roles are easy to understand as they are derived from business concepts. The analysis of the business information by experts to a high-level role model and the mapping into low-level accesses by IT Specialists are very time-consuming and costly. Furthermore the resulting role model could lead to a different access control configuration than the current one. It is likely that users get less permissions than they used to have, which might prevent them from doing their tasks efficiently.

Bottom-Up Approach

The bottom-up approach exploits the current user-permission assignments and tries to gather a role model out of it. Since this approach often uses Data Mining Techniques, the method is also called Role Mining[22]. This approach on the other hand

is often failing in generating a comprehensive role model, which is accepted by the administrators[15]. Therefore it is necessary that the business is checking the semantic meaning of the roles generated before they are used in production.

Hybrid approach

Since the advantages and disadvantages of the top-down and the bottom-up approach are mirrored, hybrid approaches have been suggested [16][26]. In these approaches the business information is leveraged to guide the computational generation of a role model out of user-permission assignments.

A role model is evolving over time and needs constant maintenance[31]. Administrators and Re-attestation processes are changing the user-permission assignments by adding and removing users and permissions to and from roles. New users are joining and others get removed, same is valid for applications and their permissions. Separate access control configurations might need to be migrated to one access control configuration. The task of role engineering is therefore not a one time task, but an on-going process (role life-cycle management[20]). An over-the-time suboptimal role model can increase the administration costs and has to be optimized again to benefit from the advantages of RBAC.

2.5 Role Mining

Role Mining is a bottom-up role engineering approach. The basic input of role mining are users, permissions and the current assignments to each other. The input can be expressed as tuple $< U, P, UPA >$, where

- U is a set of users
- P is a set of permissions
- $UPA \subseteq U \times P$ are user-permission assignments

On top additional information can be taken as input like constraints, top-down information (business information) and already existing roles. The output of role mining is the role model. A basic role model state can be described as tuple $< U, P, R, UA, PA >$,

where

- U is a set of users
- P is a set of permissions
- R is a set of roles
- $UA \subseteq U \times R$ are user-role assignments
- $PA \subseteq R \times P$ are permission-role assignments

Optionally a role hierarchy is also part of the role model. In some papers, such as [28] and [8], direct user-role assignments ($DUPA \subseteq U \times P$) are added to the role model tuple in order to provide more flexibility to handle anomalous permissions which do not fit into the role structure. Alternatively a new role for each of the individual direct assignments of DUPA can be defined, although such roles are not desirable in a role model. By adding DUPA to the tuple it can be distinguished between DUPA and undiserable roles. Figure 2.6 shows the inputs and outputs of the role mining process. A specific example in matrix representation is shown in figure 2.4.

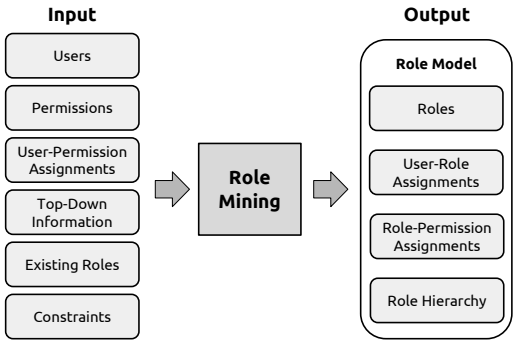


Figure 2.6: The inputs and outputs of the role mining process

2.5.1 Role Mining Problem Definitions

There are several role-mining problem (RMP) definitions existing. The problem definitions, which will be referred to in this thesis, are introduced in the following.

- **The Basic Role-Mining Problem**[41]
  "Given a set of users U, a set of permissions P and a user-permission assignment

UPA, find an RBAC configuration RC that minimizes the number of roles k and does not deviate from UPA."

The Basic RMP is the most commonly researched RMP. The minimal set of roles is probably different from the most useful set of roles, but it is argued that a minimal set of roles is appropriate when semantic information (i.e. human expert knowledge) is not given and that it is already helping security administrators by knowing the minimum required to accurately describe the current access policy configurations.

- **The δ-approx. Role-Mining Problem**[41]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, find an RBAC configuration RC that minimizes the number of roles k and deviates from UPA with less than δ assignments."
  Compared to the Basic RMP, the δ-approx. RMP bounds the degree of availability violations (underentitlements). The Basic RMP is a subproblem of the δ-approx. RMP where δ is 0. It is argued that the strict Basic RMP might not be good for the dynamic environment of users and permissions, which can change over time.

- **The Min-Noise Role-Mining Problem**[41]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, and the number of roles k, find an RBAC configuration RC with k roles, minimizing the deviation between UPA and RC."
  Compared to the δ-approx. RMP, the Min-Noise RMP bounds the number of roles, and minimizes the δ-approximation.

- **The Min-Edge Role-Mining Problem**[42]
  "Given a set of users U, a set of permissions P and a user-permission assignment UPA, find an RBAC configuration RC that is consistent with UPA and minimizes the number user-role assignments and role-permission assignments."
  Compared to the Basic RMP, the Min-Edge RMP is not minimizing the number of roles, but the number of *UA* and *PA* assignments. It is argued that minimizing the assignments also minimizes the administrative load. Furthermore the Min-Edge RMP discovers redundant roles that are heavily used[24].

- **The Interference Role-Mining Problem**[15]
  "Let a set of users U, a set of permissions P, a user-permission relation UPA,

and, optionally, part of the top-down information TDI be given. Under Assumption 1-3 (defined below), infer the unknown RBAC configuration $RC* = (R*, UA*, PA*)$.
Assumptions:
1. $RC*$ *generated* UPA
2. $RC*$ *reflects top-down information (TDI).*
3. Exceptions (errors) might exist." Compared to the other RMPs, the Interference RMP is not only compressing user-permission assignments. Algorithms, which follow the Interference RMP, set the focus on predictions of roles. The inference problem respects the hybrid role mining scenario, where additional business information is available.

In Vaidya et al.[41] it is shown that the Basic RMP, the δ-approx. RMP and the Min-Noise RMP are NP-complete problems. The problems are mapped to the database tiling and the discrete basis problem, which are in the area of data mining and data analysis.

2.5.2 Related Problem: Boolean Matrix Decomposition Problem

The role mining problem is related to the Boolean Matrix Decomposition (BMD) Problem. In Lu et al.[24] the RMP is first modelled as BMD, where a binary matrix is decomposed into two matrices. The combination of these matrices results into the original matrix. Several different decompositions can exist for a matrix.

- **Boolean Matrix Multiplication Definition:**
  $C = A \times B$ is the boolean matrix multiplication between boolean matrices $A \in \{0,1\}^{m \times k}$ and $B \in \{0,1\}^{k \times n}$. The product C is a boolean matrix with $m \times n$ dimension and is calculated with $c_{ij} = \bigvee_{l=1}^{k} (a_{ij} \wedge b_{ij})$
- **Boolean Matrix Decomposition Problem Definition:**
  The BMD problem is an optimization problem where given a boolean matrix $C \in \{0,1\}^{m \times n}$, the boolean matrices $A \in \{0,1\}^{m \times k}$ and $B \in \{0,1\}^{k \times n}$ have to be found, such that $C = A \times B$ and k is minimal[40]. $A \times B$ is called a decomposition of C.

For RMP a user-permission assignment matrix (C) needs to be decomposed into a user-role assignments matrix (A) and a role-permission assignments matrix (B). Variable k then represents the number of roles, variable m and n the number of users and permissions accordingly. Therefore finding the minimal k in the BMD problem corresponds to finding the minimal number of roles in the basic RMP. The notion of BMD for the RMP is also used in this thesis and has been already introduced in previous sections (see Figure 2.4 and 2.5).

In this thesis the role mining problem is tackled with evolutionary algorithms. An introduction to evolutionary algorithms is given in the next chapter.

Chapter 3

Introduction to Evolutionary algorithms

Evolutionary algorithms (EAs) are stochastic search algorithms inspired by the concept of the natural evolution. The general idea is to have a population of individuals, which is evolving over generations to create fitter individuals by natural selection (survival of the fittest). There are different variants of EAs: Genetic algorithms (GA), evolution strategies (ES), evolutionary programming (EP) and genetic programming (GP). All variants follow the same common concept described in the following with differences in technical details such as the representation of individuals[9].

Figure 3.1 on the next page shows a flow chart of an evolutionary algorithm process, which will be described in the following.

Proof-read chapter 3

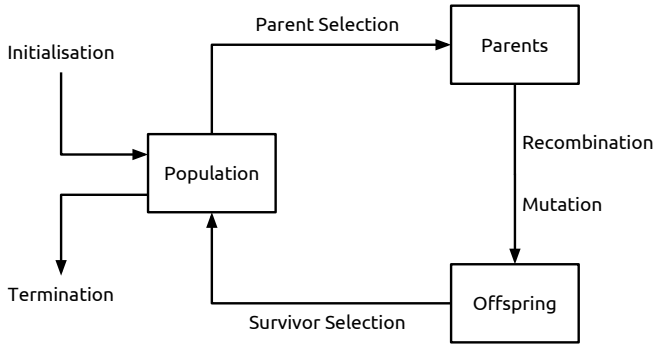


Figure 3.1: The general scheme of an evolutionary algorithm as a flow-chart [9]

The individuals in a population represent candidate solutions for the problem. A fitness function is evaluating each candidate solution in the population. Candidate solutions with a high-rated fitness are more likely to be chosen to seed the next generation than candidate solutions with low-rated fitness. The next generation is generated by applying variation operators like recombination and mutation on the selected candidate solutions. Recombination is applied to two or more candidate solutions (parents) and result in one or more new candidate solutions (children). During the recombination new candidate solutions are generated by merging random parts of the parents. The mutation operator is applied on only one candidate solution. The output is a copy of the candidate solution, where a random part is changed. The new candidate solutions, which are the output of the variation operators, are called the offspring and compete with the candidate solutions in the current population for a spot in the next generation of the population. In a survival selection the candidate solutions for the next generation are chosen. This process is repeated until a sufficient candidate solution is found or a previously set limit (e.g. number of generations) is reached.

The driving forces of evolutionary algorithms are the variation operators, which are generating new diverse candidate solutions (novelty), and the natural selection, which is guiding to better candidate solutions (quality)[9]. By preserving the possibility that candidate solutions with a lower fitness can be selected as seed for the offspring, the chance of getting stuck in a local optimum should be minimized like in other metaheuristics.

3.1 Components of Evolutionary Algorithms

An EA is defined by its components, procedures and operators, which will be introduced in the following.

3.1.1 Individuals

The individuals of a population in EA represent candidate solutions. The candidate solutions within the original problem context are called phenotypes, while their representation in the EA are called genotypes or chromosomes. A building block of a chromosome is called a gene, where the possible values for a gene are called alleles. The mapping from the phenotype to the genotype is called encoding and the inverse mapping is called decoding. A representation could be for example a binary string or a string with real values. Both also more complex structures could be the right representation. Finding the right representation of the phenotypes is problem-specific and a difficult task. It is crucial for the success of the EA. The representation influences the variation operators.

3.1.2 Population

The population is a list of individuals (genotypes), which can occur several times within the list. The individuals are immutable objects, which do not change. The population is dynamic, which will change over time due to the exchange of individuals. The parent selection of the individuals, which will be the seed for the offspring, is mostly carried out on the whole population. In most EAs the population size remains constant. The diversity of a population describes the measure of how many different individuals are present. The measure can be based on the fitness values, the phenotypes or the genotypes of the individuals. For example two individuals can represent two different phenotypes, but are evaluated with the same fitness value.

The first population consists of randomly generated individuals or of chosen individuals with higher fitness.

3.1.3 Evaluation Function (or Fitness function)

The evaluation function evaluates the individuals of a population and assigned the fitness of a candidate solution. Since the fitness influences the selection of an individual, the evaluation function encourages improvements. The goal could be to minimize or to maximize the fitness of individuals. Mathematically a minimization function can be easily transformed to a maximization problem and vice versa. The evaluation function in an EA is constructed from the objective function in the phenotype space.

3.1.4 Parent Selection

The parent selection is the selection of individuals, which will be the seed for the next generation. Typically individuals with a better fitness value get more often selected to further improve the individuals. But also individuals with a low quality fitness get a chance to pass on their genes into the next generation. This ensures that the search is not too greedy and get into a local optimum. The balance between parents with a high-quality fitness and parents with a low-quality fitness is often probabilistic.

A well-known selection mechanism is the tournament selection, where μ individuals from the input population are selected using μ tournaments. A tournament of tournament size k is selecting k random individuals from the input population and outputs the best individual (deterministic tournaments)[9]. The selection pressure can be adjusted by changing k: If the tournament size is larger, low-fitness individuals have a smaller chance to be selected.

3.1.5 Variation Operators (Recombination and Mutation)

The variation operators are responsible for discovering the search space by creating new individuals on the bases of existing individuals in the current population. All newly created individuals of a generation is called the offspring.

There are different types of variation operators. While the mutation operator only takes one individual as input, the recombination (or crossover) operator takes at least two individuals as input. The mutation operator creates a new individual (mutant),

which slightly differs from the input-individual (original). The change from the original to the mutant is chosen randomly. If the change is not random but rather guided, it is defined as an heuristic operator. The recombination operation is creating one or more new individuals (children) from its parent individuals by mixing randomly genes of these. A child might have a lower, equal or higher fitness value than its parents depending on the combination of genes.

Variation operators are depending on the representation of the phenotypes.

3.1.6 Survivor Selection (Replacement)

The survivor selection is the replacement strategy of the population μ and happens after the creation of the offspring λ. The current population is replaced by a new population - the new generation - which is mostly the same size as the population, which is being replaced. Like in the parent selection, the selection is based on the fitness values of the individuals and is often deterministic. In a fitness-biased selection for example the individuals of the population and the offspring are sorted by their fitness values and then the top segment is selected for the next population generation. In an age-biased selection only the individuals from the offspring are considered for the new population generation. If the current fittest individuals of a population is kept in the next population, the concept of elitism is used.

3.1.7 Termination Condition

The EA is either terminated when one individual is reaching a known optimal fitness value or when predefined computation conditions are met. These conditions can be for example the number of generations, number of fitness evaluations, maximum allowed CPU time or a threshold under which the population diversity has to fall.

3.2 Multiobjective Evolutionary Algorithms

The solution of an optimization problem might not only be measured regarding one objective but several, possibly conflicting objectives. These problems are called mul-

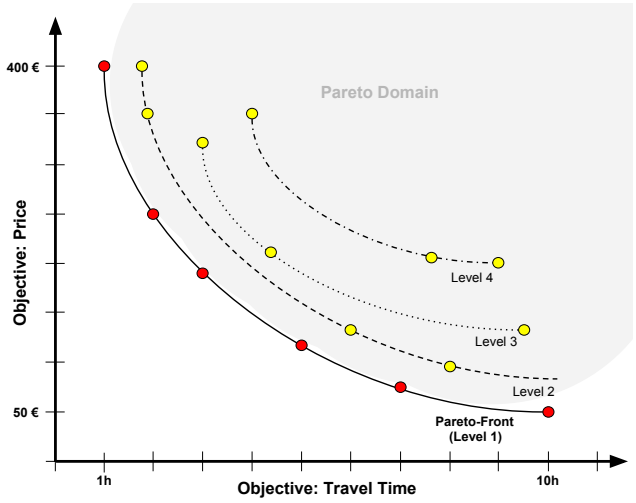


Figure 3.2: MOP Example: Buying an airplane ticket for a flight from X to Y with the objectives price and travel-time. A dot represents an offer from an airline. The red dots represents the optimal offers regarding the objectives, while the yellow dots are suboptimal offers.

tiobjective problems (MOPs). It is unlikely that there is one optimal solution for such kind of problems. It is rather likely to have a set of optimal solutions also known as Pareto-optimal solutions.

An example of a MOP could be buying an airplane ticket for a flight from city X to destination Y, where the buyer takes several factors into account. For simplification lets say the factors are price and travel time, where the rule is: The shorter the travel time, the higher the price. The buyer wants the cheapest airplane ticket and the shortest travel time, so he probably has to make a compromise between these two factors. There is a set of solutions for the two objective problem: From cheap long-travel to expensive short-travel solution.

The graph in figure 3.2 demonstrates the different options and outlines the set of pareto-optimal solutions on the so-called Pareto Front. The buyer can pick one of the cheaper solutions, but has to make a trade-off regarding the travel-time.

The fitness values of several objectives can also be combined in one single objective function, where the fitness values for each objective are weighted. This approach is also called scalarisation[9]. The weights have to be pre-determined and adjust the

preference of an objective. Sometimes these preferences are not known beforehand.

The goal in multiobjective optimization is to find all pareto-optimal solutions. The strength of EAs can be seen in solving MOPs. An EA can find several pareto-optimal solutions in one single run, since it works with a population of solutions simultaneously[7].

3.2.1 Non-dominated Sorting Genetic Algorithm II

A specific MOEA is the widely used Non-dominated Sorting Genetic Algorithm, version II (NSGA-II)[7], which is based on the concept of Pareto dominance.

The definition of pareto dominance describes a type of dominance relation between two individuals, where an individual x_1 dominates another individual x_2 if and only if 1) x_1 is strictly better than x_2 with respect to at least one objective 2) x_1 is not worse than x_2 in any of the objectives considered[2]. A pareto-optimal solution is therefore pareto-dominant over any non-pareto-optimal solution, while pareto-optimal solutions on the pareto-front are non-dominant to each other.

NSGA-II is using an elitist approach, where the selection of survivors for the next generation is based not only on the current offspring population, but also on best individuals of previous generations. Therefore the survivor selection is executed on population $P_R = P_P \cup P_O$, the combination of parent population P_P and offspring population P_O. In the first generation no previous best individuals are known, so that actually $P_R = P_P$.

NSGA-II is then sorting the population P_R of size $2N$ into different non-domination levels F_i (see section 3.2.2). The individuals in a non-domination level can be represented in a front, where the first front contains all pareto-optimal individuals of the population in respect to the objectives. The individuals in the second front are only dominated by the individuals in the first front and so on. In figure 3.2 several non-domination levels can be seen. Each individual in each front get a fitness assigned according to in which non-domination level F_i they are. Since the first level (fitness=1) is the one to achieve, the goal is to minimize the non-domination level fitness value for the individuals.

Once the non-dominated sort is complete, each individual gets a crowding distance

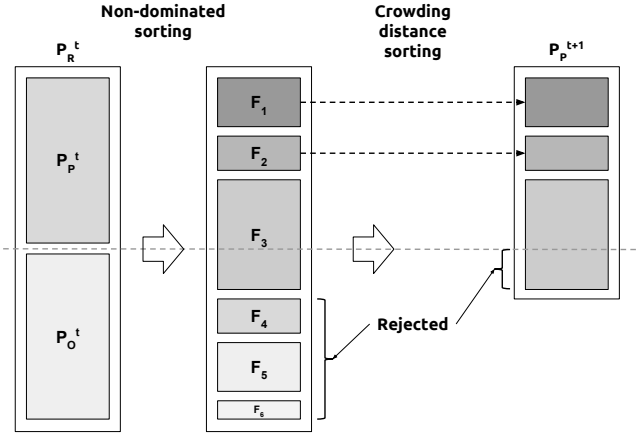


Figure 3.3: NSGA-II procedure[7]

value assigned. The crowding distance is a measure of how close an individual is to its neighbors (see section 3.2.3). Only individuals of the same non-domination level are compared to each other for the distance measure.

The new parent population P_P^{t+1} of size N in generation t is formed by the individuals in P_R with the lowest non-domination level assigned, starting with the individuals from F_1 then from F_2 and so on. The individuals of a non-domination level F_i are only added to P_P^{t+1}, if the resulting population size is not exceeding N. If the addition of the individuals in F_i is violating this condition, only the individuals in F_i with the best crowding distance are added to P_P^{t+1}. Figure 3.3 visualizes the construction of the new parent population P_P^{t+1}.

After the new parent population P_P^{t+1} is formed, the offspring population P_O^{t+1} of size N is created by using binary tournament selection for parent selection (see section 3.1.4), recombination and mutation operators. The parent selection is based on a crowded-comparison-operator $\prec_n$, in which an individual x_1 is better than an individual x_2 if 1) the non-domination level (”rank”) of x_1 is lower than the one from x_2 or if 2) x_1 and x_2 belong to the same non-domination level and the crowding distance of x_1 is higher than the one from x_2.

The full NSGA-II can be seen in algorithm 1.

The NSGA-II has relatively low computing and storage complexities. The stan-

Algorithm 1 NSGA-II by Deb et al.[7]

1: **procedure** NSGA2($popSize$, $NGEN$)
2: $t = 0$
3: P_P^t = generate-population($popSize$)
4: $P_O^t = \emptyset$
5: **while** $t <= NGEN$ **do**
6: $P_R^t = P_P^t \cup P_O^t$
7: F = fast-non-dominated-sort(P_R^t)
8: P_P^{t+1} = $\emptyset$ and i = 1
9: **while** $|P_P^{t+1}| + |F_i| <= N$ **do**
10: crowding-distance-assignment(F_i)
11: $P_P^{t+1} = P_P^{t+1} \cup F_i$
12: $i = i + 1$
13: **end while**
14: Sort($F_i, \prec_n$)
15: $P_P^{t+1} = P_P^{t+1} \cup F_i[1 : (N - |P_P^{t+1}|)]$
16: P_O^{t+1} = make-new-pop(P_P^{t+1})
17: $t = t + 1$
18: **end while**
19: **end procedure**

dard implementation has a complexity of $O(MN^2)$, which is due to the non-dominated sorting[7] (see section 3.2.2). In Fortin et al.[12] the authors are improving the NSGA-II using a more efficient divide and conquer approach, such that the complexity is $O(Nlog^{M-1}N)$.

3.2.2 Fast Non-dominated Sorting

The sorting of individuals into according non-domination levels in Deb et al.[7] is as follows. For each individual i the number of individuals, which are dominating the considered individual i, are calculated in a domination count n_i. Furthermore a set of individuals S_i, which the considered individual i is dominating, is collected. All individuals, which have a domination count zero, are in the first non-domination level F_1. For each individual i in the set S_i of each individual in F_1 the domination count n_i is decremented by one. The individuals, which now reach a domination count n_i of zero are in the next non-domination level F_2. This procedure is continued till all individuals are in a non-domination level. When N is the population size and M the number of objectives, $O(MN^2)$ comparisons are needed to identify domination counts n_i and sets of dominated individuals S_i.

3.2.3 Crowding Distance

The crowding distance is a measure of how close an individual is to its neighbors. In NSGA-II it is used to sort individuals of a non-domination level. As it can be seen in figure 3.3 only the individuals with the best crowding distance of the non-domination level, which would exceed the population size of the new parent population, make it into the new parent population ("last front selection"[13]). Furthermore the crowding distance is used for the crowded-comparison-operator $\prec_n$ in the binary tournament selection.

The crowding distance of an individual with respect to one objective m is calculated by the distance of the left and right neighbour of the individual according to their normalized fitness values. Individuals in the boundary of a non-dominated level (individuals with smallest and largest fitness values according to n objective) get an infinite distance value assigned. The crowding distance of an individual with respect to all

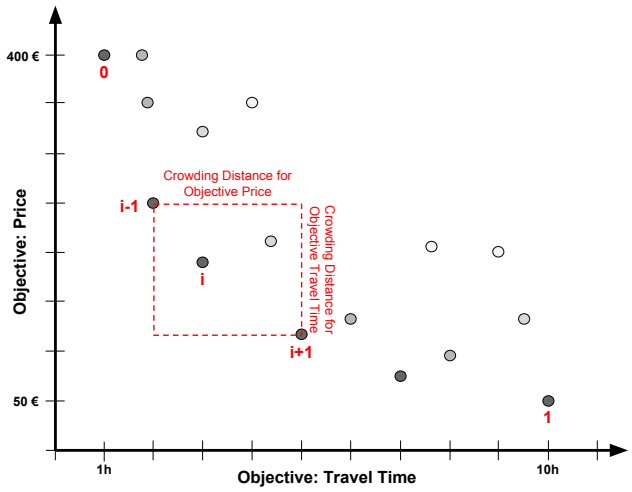


Figure 3.4: Crowding-distance calculation of an individual i corresponding to two objectives. Points of the same color represent the individuals of a non-domination level.

objectives, is the sum of the distance values for each objective of an individual. Figure 3.4 demonstrates the calculation of an individual crowding distance value according to two objectives.

The proposed algorithm for the calculation of the crowded distance in Deb et al.[7] has a complexity of $O(MNlogN)$, where N is the population size and M the number of objectives. The crowding distance ensures diversity in NSGA-II. The larger the average crowding distance, the better the diversity in a population.

The crowding distance calculation in the standard NSGA-II[7] has an instability. The individuals of a non-domination level can share the same fitness values and therefore get a crowding distance value of zero. Individuals sharing the same fitness do not necessarily share the same genotype. These individuals can only be rated better in the crowded-comparison-operation $\prec_n$, if the other individual is in another non-domination level. In Fortin et al.[13] the calculation of the crowding distance is improved to counteract this instability. The authors suggest to calculate the crowding distance on basis of unique fitnesses instead of individual fitnesses. The results show that the convergence and diversity of the developed NSGA-II*R* is improved in comparison to NSGA-II. The diversity is only improved in some cases of their conducted experiments. The time complexity of the NSGA-II*R* is the same as NSGA-II.

3.2.4 Weighting objectives in NSGA-II

In the scalarisation approach fitness values of several objectives are weighted and combined in one single objective function. How much an objective is taken into account can be therefore adjusted by the weights. For multi-objective algorithms like NSGA-II the goal is to find all possible trade-offs between objectives. Including weights in the fitness functions of the objectives would not have an effect how much a objective is taken into account, since in multi-objective algorithms does not put the fitness values of different objectives into relation, but rather looks at the Cartesian product of the objectives fitness values.

In Clune et al.[2] the authors are using NSGA-II with a stochastic version of Pareto dominance. The selection pressure on a second objective is adjusted by a probability p. The lower p, the lower is the selection pressure.

The definition of stochastically pareto dominance according to Clune et al.[2] describes a type of dominance relation between two individuals, where two objectives are considered. Let r be a random number in [0; 1] and p the probability for taking the second objective into account. An individual x_1 dominates another individual x_2 if and only if 1) $r > p$ and x_1 is strictly better than x_2 with respect to the first objective 2) $r <= p$ and x_1 is not worse than x_2 with respect to either objective and x_1 is better than x_2 with respect to at least one objective.

The stochastic pareto dominance replaces the pareto dominance in NSGA-II and is therefore used in the parent selection and when the individuals are sorted into non-domination levels.

In the next chapter other research on role-ming with evolutionary computation approaches are introduced.

Chapter 4

Related Work

Role Mining has been first coined in Kuhlmann et al.[22]. In the following years several researchers have analyzed Role Mining further and defined several Role Mining problems, quality measures, cleaning techniques and algorithms. In the following the related work to this thesis is introduced.

Proof-read chapter 4

4.1 Role Mining of ”Meaningful Roles”

In the recent years several researchers are investigating the problem finding ”meaningful” roles, since the classic Role Mining approaches output RBAC models, which are often not accepted in practice. In Frank et al.[16] a probabilistic role mining approach is combined with business information to a hybrid role mining algorithm. Where as in Xu & Stoller[44] a role mining algorithm based on bottom-up data is creating candidate roles, which then get post-processed with business information to determine meaningful roles.

There are two algorithms introduced by Xu & Stoller[44], which both consists of three phases. Both algorithms start with the same phase of candidate role generation. In the first algorithm, the ”Elimination Algorithm”, the generated candidate roles build the initial role model, which then gets reduced by role elimination. At last a role restoration phase is executed. In the second algorithm, the ”Selection Algorithm”, on

the other hand the initial role model is empty and is then filled by role selection from the candidate roles. The algorithm ends with a role pruning phase.

For the candidate role generation the authors extend the role mining algorithm in Vaidya et al. (CompleteMiner)[43], which is mining roles by subset enumeration, with inheritance relation of roles. The output is therefore a candidate role hierarchy. But the authors mention that also other approaches can be used to determine a candidate role hierarchy, like e.g. frequent pattern tree (FP-tree[18]) approach.

In the role elimination phase of the elimination algorithm roles get repeatedly removed from the initial role model as long the removal causes no violations according to the given access policy configuration and improves the quality of the role model. The role with the worst role quality gets removed first. With every removal the role hierarchy and the roles get adjusted, such that no violations occur. In the third phase of the elimination algorithm the previously removed roles get restored in the same order if the role model quality gets improved. Also here the role hierarchy and roles get adjusted accordingly.

The role model quality gets measured by the Weighted Structural Complexity (WSC) and role model interpretability. The WSC is a sum of the count of roles, user-role assignments, role-permission assignments and role-role assignments. Each summand has a weight, which can be adjusted according to the importance. The interpretability of a role model measures how well the users in the roles can be expressed by user attribute expressions. User attribute expressions are basically IF-THEN-rules, for example "$Department = Sales \wedge Location = Denmark$". The count of users, who do not fulfill this user attribute expression, is called attribute mismatch. For each role and its users the minimum attribute mismatch is chosen among all attribute mismatches of all possible combinations of user attribute expressions. The interpretability of a role model is the sum of the minimum attribute mismatch of each role in the role model.

For the role quality the authors suggest three measures: Clustered Size, Attribute fitness and Redundancy. The clustered size of a role measures the number of access policy configurations covered by the role in relation to the number of users of the role. The attribute fitness of a role is based on the minimum attribute mismatch of the role. The redundancy of a role is measured by how many other roles cover the same fraction of the access policy configurations.

The Selection Algorithm works in the opposite way as the elimination based algorithm: Candidate roles are repeatedly added to the role model by choosing the role with the best role quality first. This is continued until the role model does not violate the given access policy configuration. In the third phase the roles in the role model get checked in the reverse order that they have been added. The check determines if a role can be removed, such that the given access policy configuration is not violated, and if the removal would improve the role model quality. If the check is positive the role gets removed from the rolemodel.

The experiments are executed on public available datasets, which initially only provide access policy configuration information, but have no user attribute information. The authors generate synthetic user attribute information for the public available datasets by advanced computation.

The authors compare both algorithms and show that the elimination algorithm performs better than the selection algorithm. Furthermore the elimination algorithm is compared with other role mining algorithm, i.e. HierarchicalMiner[29]. The results show that equal or better results than previously proposed algorithms can be achieved according to WSC and Interpretability. For future work the authors suggest to consider other interpretability measures, which consider heterogeneity of users in different roles as well as homogeneity of users in the same role.

In Du & Chang[8] the authors exchange the Elimination Algorithm with a GA-based algorithm and the Selection algorithm with an Ant-Colony-Optimization (ACO)-based algorithm. The authors compare both algorithms and show that the GA-based approach produces better results than the ACO-based approach in consideration of the objectives. Furthermore the results are compared with the results in Xu & Stoller[44] and it is shown that the proposed algorithms achieve better performance in consideration of the objectives.

The created EA in this thesis is tested on the same datasets used by the papers. In contrast to the introduced papers, the approach in this thesis is to involve the interpretability measure when generating a role model from scratch, instead of optimizing a pre-mined set of candidate roles. Furthermore a different interpretability measure is introduced as used in Xu & Stoller[44] and Du & Chang[8], which is not only counting the attribute mismatch, but uses other evaluation measures from data mining. The provided synthetic user attribute information in Xu & Stoller[44] is restricted by the

given public datasets. To be able to construct input data, which contains access policy configurations and user attribute information, with various size and complexity a data generator is created. With an own data generator synthetic datasets in different sizes can be generated to test this thesis' approach.

4.2 Role Mining with Bio-inspired Techniques

The work of Saenko & Kotenko [34] [35] is the only known approach of solving the Basic RMP and the Min-Edge RMP with a genetic algorithm (GA), where a role model should be generated from scratch.

For the evaluation function of the GA the authors combine several objectives in one single objective maximization function, where the fitness values for each objective are weighted. The objectives for the Basic RMP are the minimization of roles, confidentiality violations (overentitlements) and availability violations (underentitlements). A confidentiality violation expresses a situation where a user would get an access right due to the generated role model, which he or she did not had in the current access control configuration. An availability violation on the other hand expresses if a user gets an access right less than he or she did had in the current access control configuration. For the Min-Edge RMP[24] the objective of minimizing roles is exchanged with minimizing the number of user-role- and role-permission assignments.

In a first version of the approach the authors choose a representation of role models as individuals consisting of three chromosomes: A chromosome X, which represents the UA-Matrix, a chromosome Y, which represents the PA-Matrix, and a control-chromosome Z, which controls if a role is active or passive. With the control-chromosome they influence how many roles a role model has and make it possible to vary the amount with variation operators of the GA. The authors suggest a crossover function in three phases, which is particularly designed for the suggested multi-chromosomal representation. The drawback of this approach are unnecessary passive genes. The chromosomes X and Y, respectively the UA- and PA-Matrix, contain roles, which are passive (controlled by chromosome Z).

Due to this drawback the authors suggest an improved representation where they combine the X and Y chromosome into one variable-length chromosome and remove

the control-chromosome Z. A role model is now representation as chromosome where a gene represents a role. A gene (role) consists of a user-list (L_X) and a permission-list (L_Y). Due to the change of representation the crossover method is changed accordingly.

The first approach is only tested on randomly generated data with three different dimensions[34]. The evaluation is only on the number of generations needed to obtain a solution which meet the suggested evaluation function. The performance of the first and the improved GA are compared with the result that the improved GA has a better performance in all dimensions of the synthetic input UPA, especially in greater dimensions[35]. No further evaluations on the resulting role models in the suggested approaches are shown. In the recent paper [21] the authors focusing on the multi-chromosomal approach again.

In this thesis' approach the suggested improved version of the EA of Saenko & Kotenko[35] is used as starting point. The same representation of individuals, fitness functions and crossover operation is re-implemented. Other parts of the EA, like mutation operators or selection strategy, are not mentioned in the paper and are therefore guided by standards or intuitive ideas. Furthermore a MOEA is used in this thesis, which does not require to combine several objectives to one fitness function by scalarization. The re-implemented EA and the MOEA are tested on datasets, which are commonly used in the role mining research, and analysis the results on several objectives.

In the next chapter this thesis' approach of using an evolutionary algorithm as role mining algorithm is presented.

Chapter 5

Approach

The role mining problems (see section 2.5.1) have been approached with different problem formulations, techniques and algorithms. In this thesis the approach of solving role mining with an evolutionary computation approach is investigated. Evolutionary computation might be a new interesting approach for Role Mining.

Since there has not been a lot of research done up till now regarding Role Mining with evolutionary computation (see Chapter 4 for related work), the research in this thesis starts with a basic set up of the Basic-RMP and Min-Edge-RMP as evolutionary computation problem. The focus is on building an $RBAC_0$ model from scratch, where the input is an access policy configuration as UPA-Matrix. Furthermore a measure for the role interpretability is suggested in order to measure the quality of a role model and to solve the Interference-RMP with the same approach.

In the following sections the evolutionary algorithms, the representation of an individual, variation operators, selection mechanisms and fitness functions used in this approach are described. For handling constraints for an $RBAC_2$ role model, a penalty function is suggested.

Proof-read chapter 5

5.1 Evo-RoleMiner and Evo-RoleMiner*M*

In this thesis a single- and a multi-objective EA for Role Mining has been constructed, where the objectives can be exchanged in regard to the considered Role Mining Problem. In the following the algorithms will be called Evo-RoleMiner and Evo-RoleMiner*M* accordingly.

The Evo-RoleMiner is based on the evolutionary algorithm process described in chapter 3. A pseudo-code is outlined in Algorithm 2 in the Appendix.

The Evo-RoleMiner*M* is based on the NSGA-II algorithm described in section 3.2.1. A second version of the Evo-RoleMiner*M* is based on the improved NSGA-II*R* algorithm from Fortin et al.[13] (see also section 3.2.3). In a third version of the Evo-RoleMiner*M* the second of two objectives can be relaxed by setting a weight (see also section 3.2.4). The third version builds on version two of the Evo-RoleMiner*M* and is based on NSGA-II*R* with weights.

5.2 Representation of a Role Model as Individual

The representation of individuals in an EA is crucial for the success of the EA and influences the variation operators. Finding a good representation of role models as individuals for a Role Mining EA is a challenging task, since the number of roles for a role model is not necessarily predefined and part of the search. After considering a bit-string representation and the multi-chromosomal representation proposed by Saenko & Kotenko[34] (see Appendix A.1), the improved representation proposed by Saenko & Kotenko[35] has been chosen as representation of individuals. One of the drawbacks of the other representations are unnecessary information (unused roles). Another disadvantage of the representations are complex crossover operations.

The improved representation eliminates these drawbacks by having one chromosome for an individual, where no unnecessary information occur. The chromosome consists of a list of roles, which contain a user-list and a permission-list (see figure 5.1). For the Evo-RoleMiner and Evo-RoleMiner*M* the user- and permission lists are implemented as sets.

L^X	L^Y
3	1, 3, 4, 5
2, 4, 5, 6, 7	1, 4, 5
1, 6	1, 2

Figure 5.1: Example of a role model representation suggested in Saenko & Kotenko[35]. A row represents a role (gene). In the column L^X are user lists and in the column L^Y are permission lists.

5.3 Variation Operators on Role Model Representation

Dependent on the choice of the representation of individuals in the previous section, according mutation and crossover methods are created. The choice of variation operators and how often they are executed influences how much of the search space is discovered during the evolution. Since the improved representation of Saenko & Kotenko[35] has been selected for this thesis, the following variation operators are suited for this representation.

5.3.1 Mutation methods

Since no further instructions for the mutation are given in Saenko & Kotenko[35], the mutation operations have been chosen intuitively. For the mutation six different mutation types are implemented:

- Add a new role
- Add a user to a role
- Add a permission to a role
- Remove a role
- Remove a user from a role
- Remove a permission from a role

Examples can be seen in Figure 5.2. Which role, user or permission is added or removed is chosen randomly. If an individual gets mutated is determined by a fixed probability $MUTPB$. In addition, fixed probabilities for each mutation method are dictating how the individual gets mutated. The probabilities allow to influence how often a certain mutation should be executed and can therefore influence how much new solutions are discovered and how likely good solutions are lost. Furthermore the probabilities of adding or removing a role could be set to 0 for the Min-Noise-RMP (see section 2.5.1).

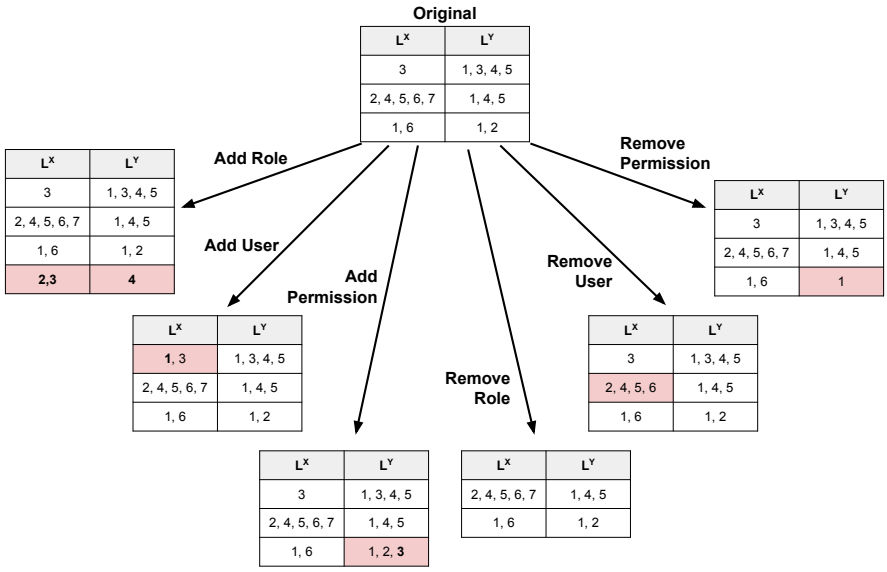


Figure 5.2: Examples for the six different mutation operators

There are two implementation variations of the mutation operations. In the first implementation, mutation towards an incomplete role model is allowed. An incomplete role model is a role model where users with no role assignments and permissions not contained in any role can occur.

The second implementation prevents that incomplete role models with missing users or permissions can occur. This is ensured by re-executing the mutation type. For example if the mutation of removing a user would lead to a role model where the user has no role assigned, the mutation is discarded and executed again, where a different

user is randomly chosen. If all users only have one role assigned, the reverse mutation (in this case adding a user) is executed. To prevent deadlocks, this reverse-switch is only allowed once. This second implementation of mutation operations restricts the search space.

5.3.2 Crossover methods

The recombination used for the Evo-RoleMiner and Evo-RoleMiner*M* is like in Saenko & Kotenko[35] executed on two individuals with a traditional one-point crossover: Both individuals are keeping their first *x* roles, while the rest of the roles get exchanged as shown in figure 5.3. The points of crossover (after *x* roles) of the two individuals are randomly chosen.

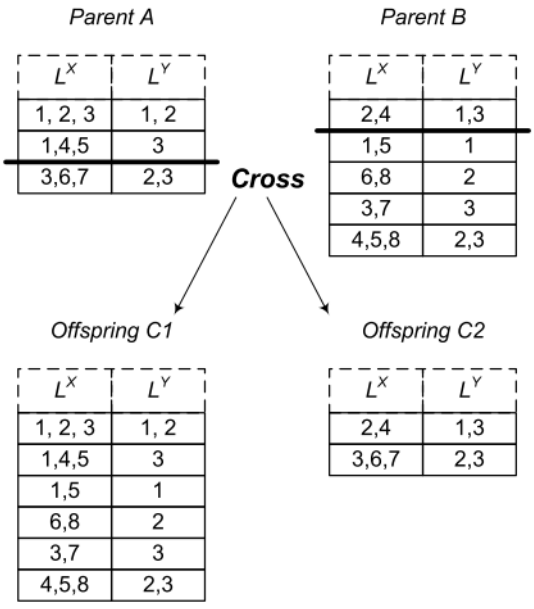


Figure 5.3: Examples of the crossover in the improved GA for the RMP by Saenko & Kotenko[35]

As for mutation there is also a fixed probability *CXPB* for crossover operations. It should be noted that the crossover operation can lead to incomplete role models, where users might not get any role assigned or permissions are not contained in any role. While the prevention of incomplete role models is straight forward in the mutation

operations, this assurance is less straight forward in the crossover operation.

5.3.3 Local optimization

A mutation and crossover is always followed by a local optimization on the individual. Like in Saenko & Kotenko[35] the individual is compressed by the rule of User combining and/or Permission combining. If there are roles with the same user sets, they get combined by merging the permissions sets (User combing). Accordingly roles get combined by merging user sets when the same permission sets are present (Permission combining). It should be noted that by this compression the number of roles in the role model can shrink. Furthermore the order of first user combining and then permission combining might not necessarily optimize the individual the best possible way (see example in Figure 5.4b).

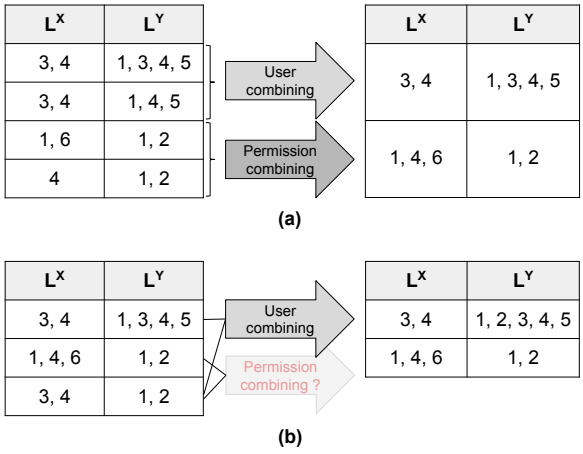


Figure 5.4: Examples of local optimizations. In (a) a case for user- and a separate case for permission-combining is detected and executed. In (b) an overlapping user- and permission-combining is detected, but only the user-combining is executed.

In the Evo-RoleMiner and Evo-RoleMiner*M* this local optimization can be switched on and off by the parameter ”optimization”.

5.4 Selection Strategy

Which selection mechanisms are chosen is dependent on if a single-objective or a multi-objective EA is applied. For the parent selection in the Evo-RoleMiner a tournament selection is chosen since it is a widely used selection strategy in EA (see section 3.1.4). With the tournament size k the selection pressure can be adjusted. The same selection strategy is chosen for the survivor selection in the Evo-RoleMiner.

For the Evo-RoleMiner*M* the selection strategy is dictated by the NSGA-II and NSGA-II*R* (see section 3.2.1 and 3.2.3), where the parent selection is based on a binary tournament selection and the survivor selection is executed on the population of the parent population P_P and the offspring population P_O.

5.5 Measuring the Interpretability of Roles

The goal is to generate a role model, which is accepted by the business, in particular the security administrators. Therefore an appropriate measure has to be found. The interpretability of a role shall express how easy a reasonable interpretation of the role can be identified, such that the role gets adopted by security administrators. An interpretable role is also called a ”meaningful” role. Another motivation for generating roles, which are interpretable by user attributes, is to predict the roles a user shall have. Furthermore the interpretable roles in RBAC can be easily migrated to ABAC.

The interpretability of a role in this thesis is measured with the help of user attributes. One or more user attributes can describe the users of a role. For example:

Let $OrganizationalUnit \in \{HumanResources, Sales, Motor\}$,

$Location \in \{Denmark, Germany, US\}$ and

$EmployeeType \in \{Internal, External\}$

be the user attributes.

The users of a role can be described by an AND-combination of the attribute values,

e.g. $Sales \wedge Denmark$ when all users in the role are in *Sales* and *Denmark*. For simplicity an OR-combination, e.g. $Sales \vee Motor$ where all users would be either in *Sales* or $Motor$, is not considered. This case can be rather represented by two roles with the same permission-set, but different user-sets. It should be noted that this requires the disabling of the permission-combining optimization (see section 5.3.3).

Two ideas have been developed in this thesis to measure how well a role can be described by the user attribute values of its user-members. In the following both ideas are introduced, where the later one has been implemented.

5.5.1 Role Cluster Approach

One suggested approach developed in this thesis is to treat roles like clusters where the users and their attribute values are objects. A role-cluster is considered good if the user-objects within the role-cluster are compact and the user-objects between role-clusters are separated. This means that the user attribute values considered describe all users within a role (compactness), but no users not in the role (separation). The compactness and separation of a role-cluster can be measured with the average silhouette coefficients of the user-objects of a role-cluster[18].

For each user u the average distance between the user and all other users in the role, the user u belongs to, has to be calculated for the compactness measure:

$$a(u) = \frac{\sum_{u' \in R_i, u \neq u'} dist(u, u')}{|R_i| - 1} \tag{5.5.1}$$

where user u is in role R_i as well as users u'. The average distance *dist* is calculated based on the attributes of a user. The separation is measured by:

$$b(u) = \min \left\{ \frac{\sum_{u' \in R_j} dist(u, u')}{|R_j|} \right\} \tag{5.5.2}$$

where user u is not in role R_j, but the users u' are. The average distance *dist* is calculated based on the attributes of a user. The silhouette coefficient of an user-object can then be calculated with:

$$s(u) = \frac{b(u) - a(u)}{\max(a(u), b(u))} \tag{5.5.3}$$

To calculate the fitness of a role-cluster, the average silhouette coefficient of all users in the role can be calculated. If the average silhouette coefficient converges 1, the role-

cluster is considered compact. A negative silhouette coefficient says that the user-object in a role is closer to user-objects of other role-clusters than to user-objects of the same role-clusters.

This measure of the average silhouette coefficients requires the role-clusters to be crisp, which means a user-object can be only in one role-cluster. But since the users can have several roles the role-clusters are fuzzy. Furthermore the measure has to be executed for every user attribute combination. Therefore an altered calculation of silhouette coefficients has to be used or a different measure has to be used to evaluate the role-clusters. To evaluate fuzzy clusters the sum of the squared error (SSE) can be used[18]. It measures how well a fuzzy clustering fits a data set. In Rawashdeh & Ralescu[33] the authors suggest an algorithm for the silhouette coefficient in fuzzy clusters.

For the role-cluster approach no working implementation has been achieved.

5.5.2 Role Classifier-Rule Approach

Another suggested approach developed in this thesis is to generate classifier-rules for roles and measure the accuracy of these rules. Additionally the rule should be simple and not too complex (e.g. many conditions, which identify exactly one user). For this supervised-learning approach, all users in a role get classified as being in the role, while all other users are classified as being not in the role. Hence for each role a binary classification problem has to be solved. Rules can be extracted from a decision tree or can obtained directly using a Sequential Covering algorithm[18]. Both approaches will be introduced in the following.

- **Rules from decision trees**
  In traditional decision tree construction like ID3 or C4.5 the dataset gets repeatedly split based on the attribute which most effectively splits the dataset in one class or the other (information gain). The depth of the tree could be used as the measure of the complexity of a rule. Figure 5.5 demonstrates an example decision tree for a role.

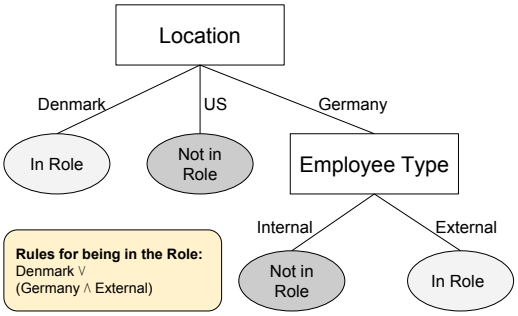


Figure 5.5: Example of decision tree for a role

The split by information gain is not necessarily wanted for the rules describing a role, since permissions originally have not been assigned to users which share the most common user attribute values, but rather because users gained legitimation through other specific attribute values. For example although users in a generated role are all located in *Denmark*, while users not in the role are located elsewhere, the reason that users of the role got assigned to the permission set the generated role is grouping could be due to the Departments they are in, which could be diverging. The information gain for the location would be higher, but not necessary the legitimation, why a user gets the permission set assigned through the role. Also the measure used for splitting the data in a decision tree is often biased towards class imbalances or multivalued attributes[18].

Furthermore classifier-rules for both classes are discovered, while for the measuring of the role only the rules for class "*In the role*" are from interest. Therefore the decision tree approach is not considered further.

- **Direct rule induction**

  The second approach is to discover the most descriptive rules for a role (class "*In the role*") by direct rule induction.

  The algorithm developed in this thesis got inspired by Sequential Covering algorithms, which directly learn rules for classification by repeatedly removing a portion of the dataset. In the following the algorithm for rule induction is described. The according pseudo-code of the algorithm can be seen in Algorithm 3 in the Appendix B.3.

  Before the algorithm is applied on a role, all users of the role in the role model

get classified as "*In the role*" or "*Not in the role*". A set of all user attribute combinations is created (see line 2, Algorithm 2). If X user attributes are considered, $2^X - 1$ combinations are created. For example with the user attributes *Department* and *Location*, the following combinations are possible:

$$(Department), (Location), (Department, Location)$$

The Figures 5.6 and 5.7 on the next pages are supporting the following further description.

The first user in the role is picked ((1) in Figure 5.6) and the smallest user attribute combination (for example $\{Department\}$) is chosen ((2) in Figures 5.6 and 5.7). A rule is generated out of the user attribute values of the user for the chosen user attributes. In the example the *Department* the user is in, creates a rule, i.e. $rule=\{()HR\}$.

Next it will be checked if users not in the role also comply with the rule ((3) in Figures 5.6 and 5.7). If too many users outside the role also comply with the rule, the rule is discarded. Basically the sensitivity of the rule according to class "*Not in the role*" is calculated and checked if it is beneath a specified threshold $t1$ (currently set to $t1=\frac{1}{3}$).

If the rule persists the check, all users inside the role, which comply with the rule, are not considered any further and are removed in the current iteration of the considered attribute combination ((4) in Figures 5.6 and 5.7). Then it is measured how many users inside the role comply with the rule. Again the sensitivity is calculated, but this time on the users in class "*In the role*". If a certain threshold $t2$ is met (currently set to $t2=\frac{2}{3}$) ((5) in Figures 5.6 and 5.7), the rule is considered as a potential classifier-rule for the role ((6) in Figures 5.6 and 5.7).

If the threshold $t2$ is not met ((5) in Figures 5.6 and 5.7), the potential rule gets eventually extended by another OR-connecting rule by calling the procedure recursively with the reduced user set of the role ((7) in Figures 5.6 and 5.7, but not applied). This can be specified by a parameter in the algorithm. By setting the parameter=2 an OR-connecting rule like $Department = HumanResources \vee Department = Sales$ can be constructed.

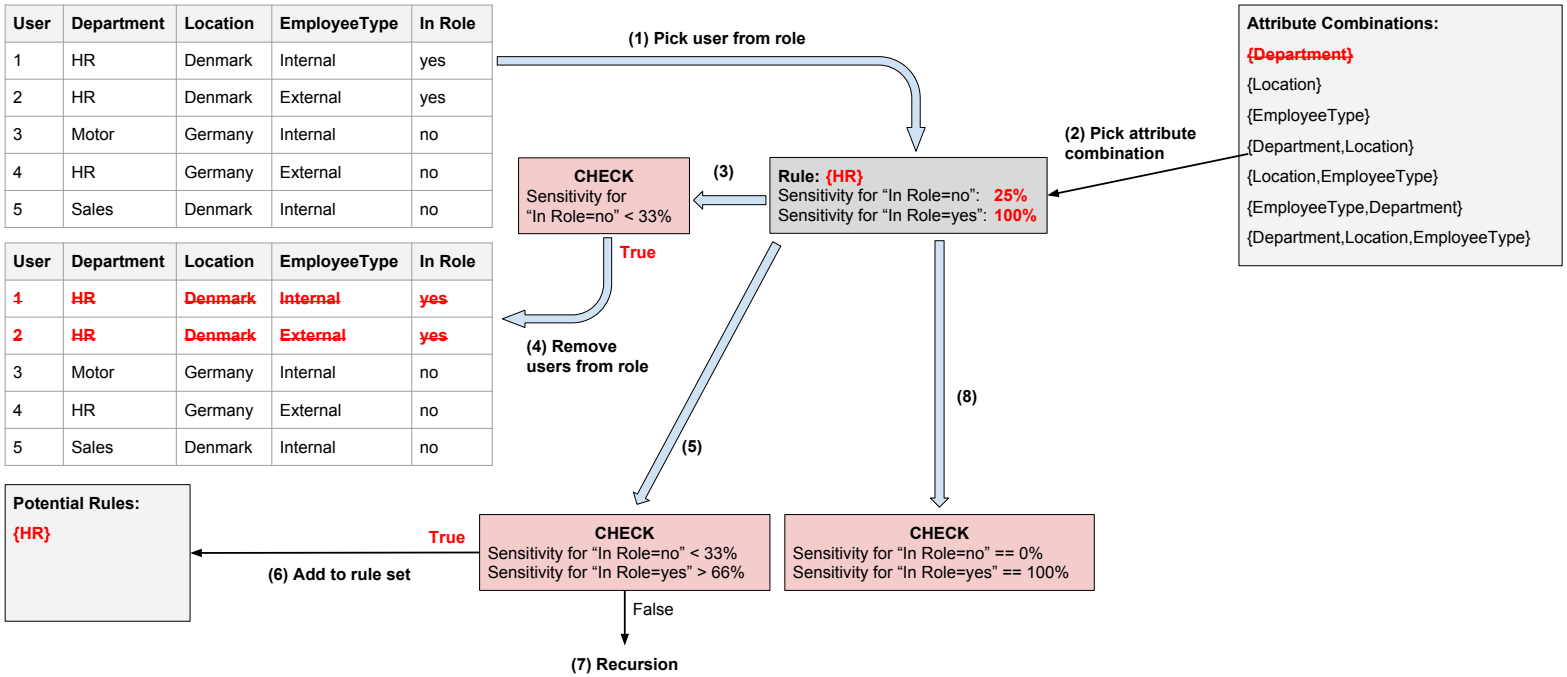


Figure 5.6: Rule Induction Extract 1: (1) A user in the considered role is picked (2) smallest user attribute combination is picked and the rule HR is created (3) Check sensitivity of rule of users with "In Role=no" (4) Remove all users inside the role, which comply with the rule (5) Check sensitivity of rule of users with "In Role=yes" (6) Save rule as a potential classifier-rule for the role (7) Recursion, where the resulting rule set is OR-connected to the current rule (not applied in this extract) (8) Check if rule is perfect potential rule

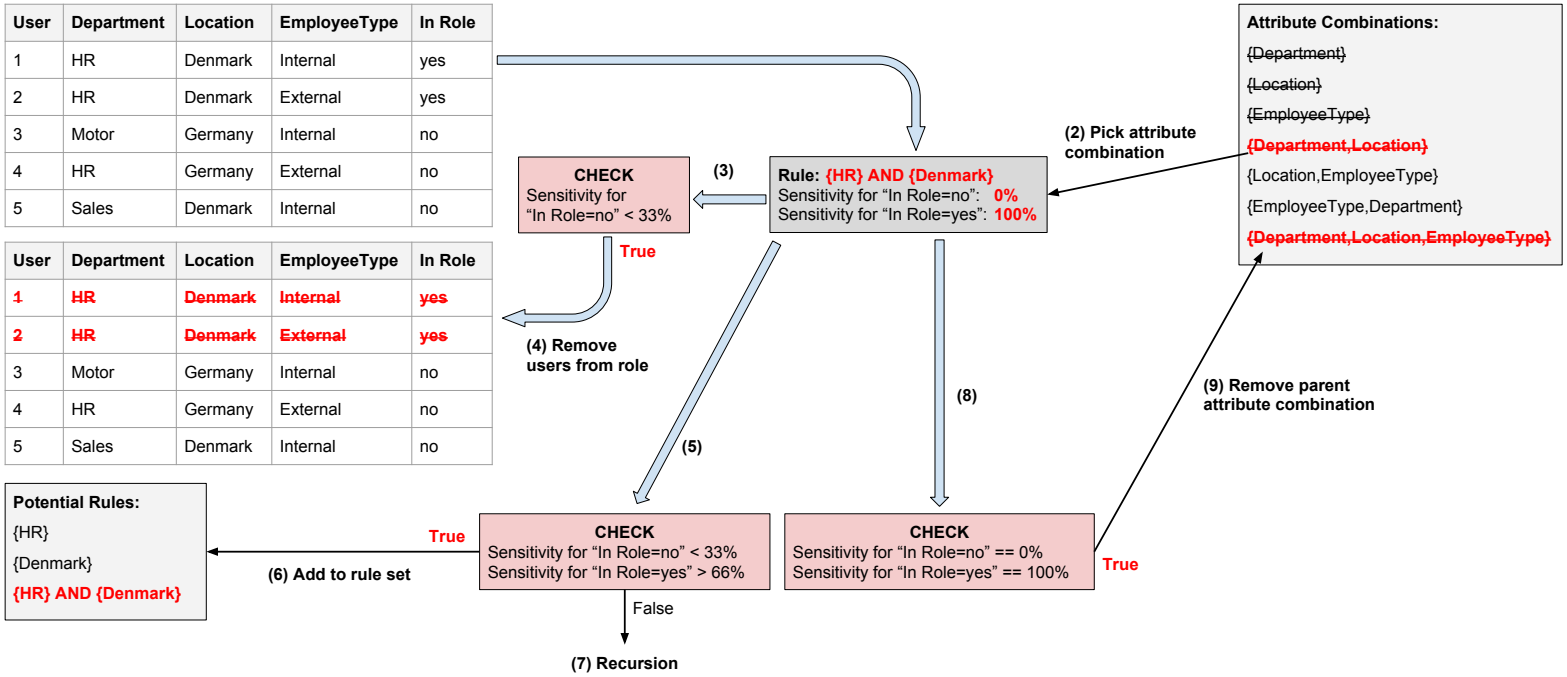


Figure 5.7: Rule Induction Extract 2: (1) A user in the considered role is picked (2) smallest user attribute combination is picked and the rule HR is created (3) Check sensitivity of rule of users with "In Role=no" (4) Remove all users inside the role, which comply with the rule (5) Check sensitivity of rule of users with "In Role=yes" (6) Save rule as a potential classifier-rule for the role (7) Recursion, where the resulting rule set is OR-connected to the current rule (not applied in this extract) (8) Check if rule is perfect potential rule (9) Remove parent attribute combinations

Certain user attribute combinations get removed when the potential rule yields a perfect potential rule, which is when the rule is complying with all users inside the role and with none of the users outside the role ((8) in Figure 5.7). The user attribute combinations, where the perfect potential rule is a subset of, get removed ((9) in Figure 5.7). If the potential rule is not perfect, a more complex user combination might yield a better potential rule and is kept as possibility in the user attribute set.

The next smallest user attribute combination left (for example ($Location$)) is chosen and the procedure continues as described above until no user attribute combinations or users are left.

After generating the rules for a role, the rule with the best accuracy gives the measure for the role interpretability (”meaningfulness”). The accuracy of a rule is measured by

$$accuracy = \frac{TP + TN}{P + N} \tag{5.5.4}$$

where:

- TP are the true positives, meaning the number of users classified as ”*In the role*” by the rule
- TN are the true negatives, meaning the number of users classified as ”*Not in the role*” by the rule
- P are the true positives, meaning the number of actual users in the role
- N are the true negatives, meaning the number of actual users not in the role

5.6 Objective Measures

Several objective measures of the role mining problems (see section 2.5.1) are implemented for the Evo-RoleMiner and the Evo-RoleMiner*M* in order to measure the individuals of a population. The implemented objective measures introduced in the following are organized in the according role model categories: Completeness, Complexity and Comprehension (see section 2.3).

5.6.1 Completeness Measures

To measure the completeness (see section 2.3.1) different measures can be created. All of them need the original access control policies UPA for measuring how complete the current considered role model is.

- **Confidentiality Violations**
  Confidentiality violations exist when users get more permissions in the generated role model, than they had in the original access control policies (overentitlements). These violations are measured by counting the additional set positive bits ("1") in the resulting matrix of $UA \times PA$ with UPA. For minimizing confidentiality violations the worst case would be the count of all negative bits in UPA, meaning that everyone gets all permissions. The best case is to have zero confidentiality violations.

- **Availability Violations**
  In contrast to confidentiality violations, availability violations exist when users get less permissions in the generated role model, than they had in the original access control policies (underentitlements). Also here the measure is executed by comparing the resulting matrix of $UA \times PA$ with UPA, but in this case the negative bits ("0") are counted. For minimizing availability violations the worst case would be the count of all positive bits in UPA, meaning that no one gets the permissions they had in the original access control policies. The best case is to have zero availability violations.

- **Average Confidentiality Violations**
  Instead of measuring the confidentiality violations of a role model, the average confidentiality violations of roles in a role model can be measured. For each role in the role model the overentitlements they are causing in comparison with UPA are counted. In contrast to the confidentiality violations measure on the role model, the average confidentiality violations of roles is taken into account that a violation can be caused by several roles.

5.6.2 Complexity Measures

To measure the complexity of a role model (see section 2.3.2) three different objectives are considered, where no additional information is needed.

- **Role Count**
  In the often researched Basic-RMP one objective is the count of roles in the role model, since it is often connected with less administration effort. The goal is to have as little roles as possible. In the used representation of a role model as individual, the number of genes represents the number of roles. For minimizing the role count the worst case can be determined as having a role for each user or for each permission and is therefore determined as $min(|U|, |P|)$. The best case can be set to one role, although it might be an unrealistic solution.

- **User-Role-Assignment Count (UA-Count)**
  In the Min-Edge-RMP the number of User-Role-Assignments and Role-Permission-Assignments have to be minimized instead of the number of roles as in the Basic-RMP. The number of User-Role-Assignments is counted by the positive bits in UA. For minimizing the UA-Count the worst case would be if every user is assigned to every role. Since the number of roles can vary, the worst case of the number of roles is considered as number of roles. Therefore the worst case for the number of UAs is $|U| \times min(|U|, |P|)$. The best case is the number of users $|U|$, since each user has at least one permission in UPA, which needs to be represented in at least one role.

- **Role-Permission-Assignment Count (PA-Count)**
  As mentioned above one of the objectives in the Min-Edge-RMP is the number of Role-Permission-Assignments, which needs to be minimized. The number of Role-Permission-Assignments is counted by the positive bits in PA. For minimizing the PA-Count the worst case would be $|P| \times min(|U|, |P|)$, where each permission occurs in each role. The best case is the number of permissions $|P|$, since each permission should be represented in at least one role.

5.6.3 Comprehension Measures

To measure how comprehensive a role model is (see section 2.3.3), a measure has been implemented, which is measuring the interpretability (”meaningfulness”) of the roles in a role model (see section 5.5). Required is the information of user attribute values, which might not always be at hand.

- **Average Role Interpretability**
  The average role interpretability is calculated as described in section 5.5.2. For maximizing the average interpretability of roles in a role model the worst case would be a value of 0, where no descriptive rule for any role in the rolemodel can be found. The best case is a value of 1.

5.7 Fitness Functions

Several objectives are often combined by scalarisation into one function in the research of role mining. This also needs to be done for the Evo-RoleMiner, since it allows only one fitness function. In the following subsections fitness functions for the Evo-RoleMiner are derived. A selection of the objective measures (see previous section) are combined by scalarisation to a fitness function. All objective measures get normalized to a value between 0 and 1 before they get combined.

For the multi-objective Evo-RoleMiner*M* two or more fitness functions can be defined. A fitness function can be the minimization or maximization of an objective measure (see previous section). To simplify a multiobjective problem the fitness functions can be also scalarization function as introduced in the following subsections. An example for two objective fitness functions in the Evo-RoleMiner*M* could be ”Minimizing Role Count” and ”Minimizing Violations”, where the latter is a scalarization function of confidentiality and availability violations.

5.7.1 Fitness Function for the Basic-RMP

In Saenko & Kotenko[35] a weighted function is suggested which takes the number of roles, confidentiality violations and availability violations into account. The function represents the objectives of the basic RMP. The authors formalize the function as maximization function:

$$F_{basic} = (w_1 * |R| + w_2 * G^{conf} + w_3 G^{accs})^{-1} \tag{5.7.1}$$

where $|R|$ is the number of roles, G^{conf} are the number of confidentiality violations and G^{accs} the number of availability violations. With the weights w_i the influence of the single objectives can be adjusted. For the Evo-RoleMiner a minimization function is used since the math operation to turn the fitness function into a maximization function is unnecessary for the outcome of the result and has only a negative influence on the performance. The minimization function is the inverse function of maximization function 5.7.1:

$$F^{min}_{basic} = w_1 * |R| + w_2 * G^{conf} + w_3 G^{accs} \tag{5.7.2}$$

5.7.2 Fitness Function for the Min-Edge-RMP

Another commonly used objective function, the Weighted Structural Complexity (WSC) [28] [44] has been used as implementation guideline. Since a role hierarchy (see section 2.2.2) is not in the scope of this thesis, the WSC used is without the number of role-to-role inheritance relations. The WSC is therefore defined as:

$$WSC(\gamma, W) = w_1 * |R| + w_2 * |UA| + w_3 * |PA| + w_4 * |DUPA| \tag{5.7.3}$$

where γ is the role model and W a set of weights. It should be noted that $DUPA$ are the direct assignments to reduce availability violations. Since the results would lead to a lot of overentitlements, the objective of confidentiality violations is added to the function. The new function is described as:

$$WSC(\gamma, W)^* = w_1 * |R| + w_2 * |UA| + w_3 * |PA| + w_4 * G^{accs} + w_5 * G^{conf} \tag{5.7.4}$$

where γ is the role model and W a set of weights. When the function in (5.7.4) was tested with different weights it showed that the number of roles got reduced quickly over the generations, although w_1 was set to 0 or lower. By looking at the optimization function description for User-Role- and Role-Permission-Assignments (see section

5.6.2) it is obvious that the number of roles already gets its impact through the number of User-Role- and Role-Permission-Assignments. When removing the first summand from (5.7.4) the function is similar to the function in Saenko & Kotenko[34] for the Min-Edge-RMP, which is defined as maximization function:

$$F_{edge} = (w_1 * (|UA| + |PA|) + w_2 * G^{conf} + w_3 * G^{accs})^{-1} \tag{5.7.5}$$

where w_i are the weights for the single objectives. Hence the maximization function in (5.7.5) is used as guideline for a minimization function for the Min-Edge-RMP:

$$F^{min}_{edge} = w_1 * (|UA| + |PA|) + w_2 * G^{conf} + w_3 * G^{accs} \tag{5.7.6}$$

where w_i are the weights for the single objectives. The Minimization function is used for the Evo-RoleMiner for the same reason why the minimization function is chosen in the Basic-RMP: The conversion into a maximization function does not influence the result and has only a negative influence on the time performance.

5.7.3 Fitness Function for the Interference-RMP

At last the objective for comprehension is combined with the objectives of completeness and complexity. The interpretability measure is added to the fitness functions of the Basic-RMP and the Min-Edge-RMP.

$$F^{min}_{basic\_INT} = F^{min}_{basic} + w_6 * (1 - INT) \tag{5.7.7}$$

and

$$F^{min}_{edge\_INT} = F^{min}_{edge} + w_6 * (1 - INT) \tag{5.7.8}$$

where INT stands for the interpretability measure and w_6 is the weight for the interpretability objective. Since the function is a minimization function, the Interpretability value is inversed in order to reward a higher interpretability.

5.8 Initialization of the Population and Termination condition

The initial population is randomly generated by creating chromosomes (role models) of either random or fixed gene size (role size), which contain a random set of users and

permissions. The initialization of role models with a fixed role size is only used for the Min-Noise-RMP (see section 2.5.1). It is ensured that each role has at least one user and permission assigned. It should be noted that incomplete role models can be generated, where users might not get any role or permissions are not contained in any role. The termination of the EA is via a previously set generation limit.

5.9 Constraint Handling for $RBAC_2$ role models

In the $RBAC_2$ model (see section 2.2.3) constraints like Separation of Duties (SoD) are taken into account. These can be used to guide the role engineering process, such that constraint violations are not occuring in the role model. Alternatively constraints violations in the role model can be accepted in the role model and the constraints are complied in a post-mechanism.
The concept of penalties in EA seem to be a good fit to easily incorporate constraints in the Evo-RoleMiner and Evo-RoleMiner*M*. A binary penalty function would distinguish between if an individual (role model) would violate a constraint or not. Individuals (role models) are therefore penalized with a bad fitness if they are not feasible (violating a constraint).
Before the fitness of an individual is calculated the Evo-RoleMiner and Evo-RoleMiner*M* are checking if all constraints are met, if they are given. If the individual (role model) is violating one of the constraint, the fitness is set to the worst fitness value.

Next the approach presented in this chapter is tested with access control policy datasets. The evolutionary computation and the resulting role models are visualized and analyzed.

Chapter 6

Experiments

The experiments with the EA Evo-RoleMiner provide information, which is not given in Saenko & Kotenko [35]. In particular how parameters can be set and how the objectives of the role models evolve and impact other objectives. Afterwards the MOEA Evo-RoleMiner*M* is tested and compared to the Evo-RoleMiner. Also the Evo-RoleMiner and Evo-RoleMiner*M* are tested on datasets commonly used in role mining research.

This chapter describes the setup of the experiments executed. First the data sets used in the experiments are described. Then the setup and the results of the experiments are introduced. The experiments have been divided into several steps, where the results of one experiment lead to the setup of the experiments after. All experiments have been executed ten times and started with a random start-population.

In an initial series of experiments the objective measures introduced in section 5.6.1, 5.6.2 and 5.6.3 are set as fitness function in the Evo-RoleMiner. This experiment series was executed to analyse the impact of these objectives to each other. It is shown that the objectives of minimizing confidentiality violations and minimizing availability violations can be conflicting.

In the second and third set of experiments the actual fitness functions for the Basic-RMP and Min-Edge-RMP (see section 5.7) are tested in the Evo-RoleMiner. In a later set of experiments the Evo-RoleMiner*M* is tested as well as the inclusion of the

interpretability measure (see section 5.5.2) into the Evo-RoleMiner.

6.1 Data Sets

Datasets are the input access control policies for the role mining algorithm. They are provided as textfiles, which need to be parsed. They contain information on which user is assigned to which permission.

6.1.1 Real Datasets

In many research papers the same datasets are used for performance evaluation of role mining algorithms. Table 6.1 lists some of these data sets and their components. The authors of [10] obtained these datasets from Cisco firewalls and the Lotus Domino server of the Hewlett Packard (HP) networks. The healthcare dataset was collected from the US Veteran's Administration.

Dataset	$\|U\|$	$\|P\|$	$\|UPA\|$
healthcare	46	46	1486
domino	79	231	730
emea	35	3046	7220
apj	2044	1146	6841
firewall-1	365	709	31951
firewall-2	325	590	36428
americas-small	3477	1587	105205

Table 6.1: Real Datasets used in Role Mining research. The table lists the amount of users, permissions and user-permission assignments in each dataset.

6.1.2 Synthetic Datasets

The real dataset commonly used in Role Mining research (see Table 6.1) do not provide user attribute information. Only Xu & Stolter[44] are generating synthetic attribute information on the given datasets. Also the data generators for synthetic datasets for

role mining are only providing user-permission assignment information, but no user attribute information, e.g. in Vaidya et al.[43].

In order to test smaller datasets with user attribute information a new synthetic data generator has been created for this thesis. The data generator allows to adjust, amongst other configurations, the dimensions of users and permissions, the density of user-permission assignments and user attribute information. The data generator is based on a reversed role-engineering process, which constructs an access control policy out of a role model. This allows to compare mined role models with the role model, the access control policy has been constructed on.

The data generator generates users, roles, user attribute rules for the roles, user-role matrix, role-permission matrix, user-permission matrix and a user-permission matrix with noise. A more detailed description of the data generator can be read in the Appendix B.4.

Two datasets have been generated with the data generator in order to execute experiments with fitness functions $F^{min}_{basic\_INT}$ (5.7.7) and $F^{min}_{edge\_INT}$ (5.7.8). The synthetic datasets are listed in table 6.2.

Dataset	$\|U\|$	$\|P\|$	$\|UPA\|$	$\|R\|$
1	10	10	42	4
2	50	50	620	15

Table 6.2: Synthetic Datasets

A visualization of Dataset1 and the Healthcare dataset can be seen in Figure 6.1 and 6.2. Visualizations of the other real datasets can be seen in the Appendix C.2. Detailed information about the generated synthetic datasets can be found in Apendix C.1.

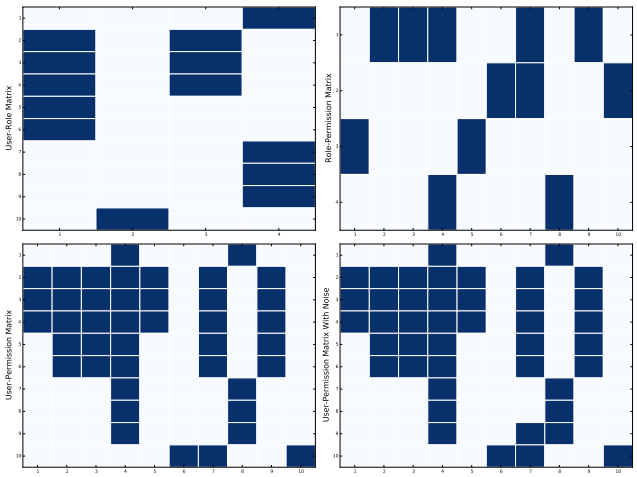


Figure 6.1: DATASET1: Role model of Dataset1 including User-Permission Matrix with 10 users and 10 permissions. From upper left to lower right: User-Role Matrix (Rows: Users, Columns: Roles), Role-Permission Matrix (Rows: Roles, Columns: Permissions), Resulting User-Permission Matrix used as input for the algorithm (Rows: Users, Columns: Permissions), User-Permission Matrix with Noise (Rows: Users, Columns: Permissions)

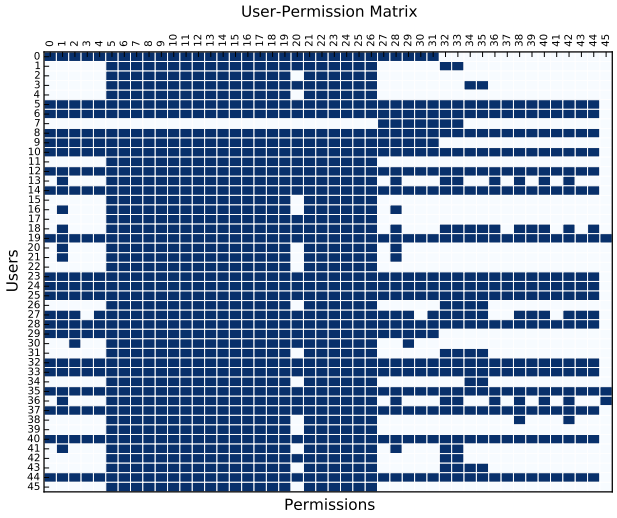


Figure 6.2: HEALTHCARE DATASET: User-Permission Matrix with 46 users and 46 permissions.

6.2 Experiment 1 - Single Objectives

For the first experiment single objective fitness functions are tested on the Evo-RoleMiner. The objectives tested are:

- Confidentiality Violations (Experiment 1a)
- Availability Violations (Experiment 1b)
- Role Count (Experiment 1c)
- User-Role-Assignment Count (Experiment 1d)
- Role-Permission-Assignment (Experiment 1e)
- Average Role Interpretability (Experiment 1f)

These are the objective measures introduces in section 5.6. All single objective fitness functions are formulated as minimization function except the fitness function with objective ”Average Role Interpretability”, which is a maximization fitness function. The experiments are executed in order to validate the objective measures and to analyse the single objectives in relation to each other.

A basic setup is chosen seen in Table 6.3. The experiments are based on the Dataset1.

Parameter	**Value**
Generations	100
Population	100
CXPB	0.25
MUTPB	0.25
MUTPB-Type1: Add role	0.25
MUTPB-Type2: Add User	0.25
MUTPB-Type3: Add Permission	0.25
MUTPB-Type4: Remove Role	0.25
MUTPB-Type5: Remove User	0.25
MUTPB-Type6: Remove Permission	0.25
Tournament size	2
Local optimization	True

Table 6.3: EXPERIMENT 1 setup

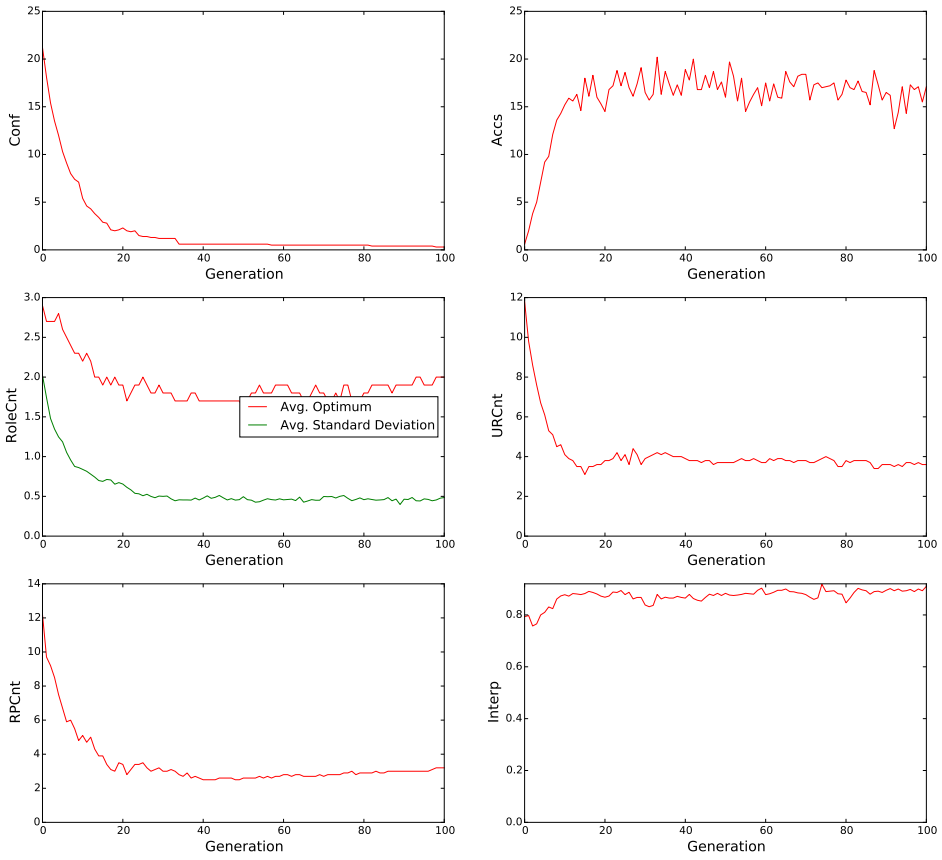


Figure 6.3: EXPERIMENT 1a: Results of Evo-RoleMiner with Fitness function $F = G_{conf}$ on Dataset1 with setup in table 6.3. From upper left to lower right: Confidentiality Violations, Availability Violations, Role Count, User-Role Assignments, Role-Permission Assignments, Interpretability.

For the single objective "Confidentiality Violations" it is expected that the individuals (role models) violate confidentiality compared to the original access configuration UPA less over time. The results in Figure 6.3 are confirming this and also show that individuals have less roles over time. This can be explained by that a lower count of roles probably result in less user-role- and role-permission assignments, which on the other hand lowers the probability of violating confidentiality. At the same time less user-role- and role-permission assignments let the probability for availability violations rise, which can be also seen in Figure 6.3.

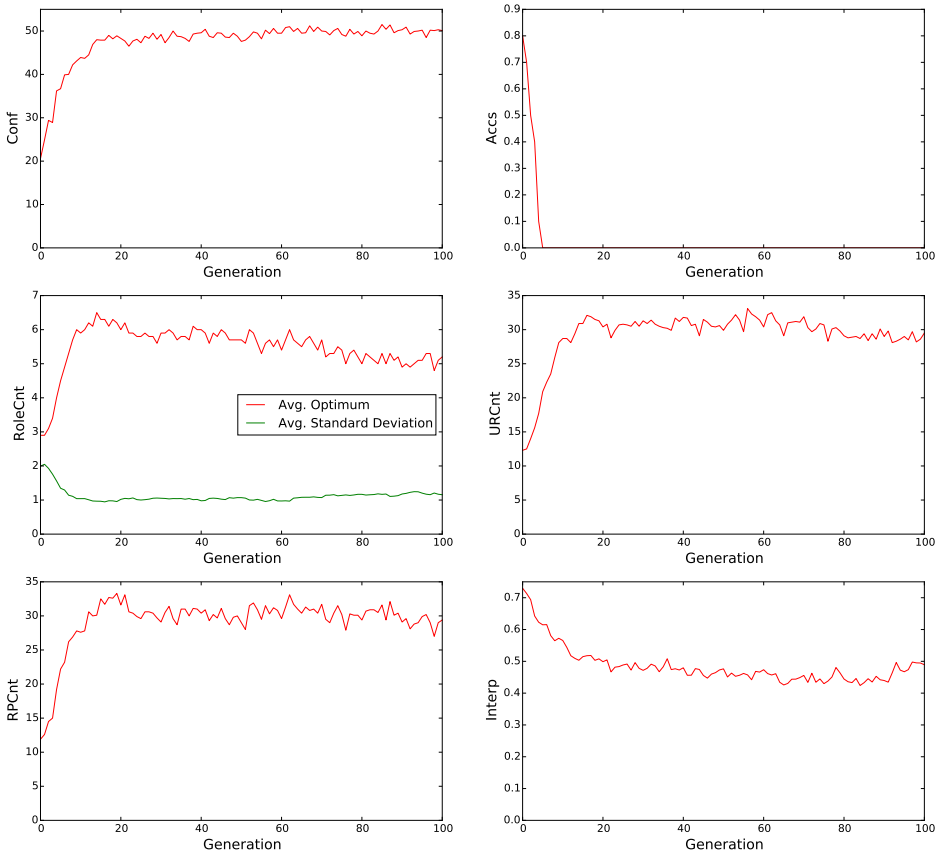


Figure 6.4: EXPERIMENT 1b: Results of Evo-RoleMiner with Fitness function $F = G_{accs}$ on Dataset1 with setup in table 6.3. From upper left to lower right: Confidentiality Violations, Availability Violations, Role Count, User-Role Assignments, Role-Permission Assignments, Interpretability.

An opposite impact can be seen when single objective "Availability Violations" is used as fitness function (see Figure 6.4). In the first generations the availability violations quickly achieve the goal of zero. During the same generations the role count, user-role- and role-permission assignments are rising. It is more likely that every user gets his permission (given in the original UPA-Matrix), if the amount of roles, user-role- and role-permission assignments are higher.

Hence, the objectives of minimizing confidentiality violations and minimizing avail-

ability violations can be conflicting. Other results of the first phase can be seen in the Appendix D.1.

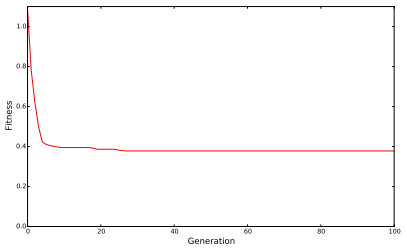
6.3 Experiment 2 - Fitness Functions

In the second set of experiments the fitness functions F^{min}_{basic} (5.7.2) and F^{min}_{edge} (5.7.6) are tested on the Dataset1 (Experiment 2a and 2b) and the Healthcare dataset (Experiment 2c and 2d). The setup for the experiments can be seen in table 6.4. The weights for F^{min}_{basic} and F^{min}_{edge} are set all to 1.0.

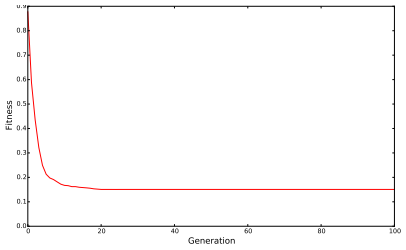
Parameter	**Value**
Generations	1000
Population	1000
CXPB	0.25
MUTPB	0.25
MUTPB-Type1: Add role	0.25
MUTPB-Type2: Add User	0.25
MUTPB-Type3: Add Permission	0.25
MUTPB-Type4: Remove Role	0.25
MUTPB-Type5: Remove User	0.25
MUTPB-Type6: Remove Permission	0.25
Tournament size	2
Local optimization	True
Weights for Fitness Function	1.0, 1.0, 1.0

Table 6.4: EXPERIMENT 2 setup

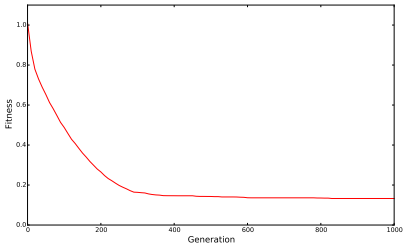
In all experiments the fitness is improving over time. The fitness functions look like a hyperbola, where the improvement is strong in the first generations and less strong in the last generations (see Figure 6.5 on the next page).



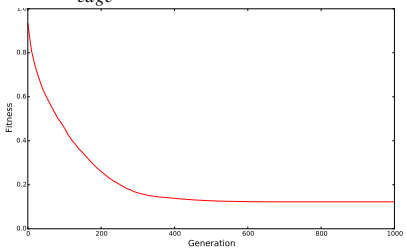
(a) F^{min}_{basic}, Dataset1, 1000 generations



(b) F^{min}_{edge}, Dataset1, 1000 generations

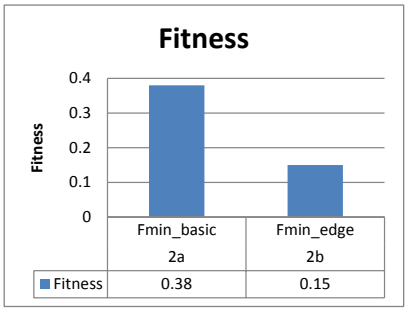


(c) F^{min}_{basic}, Healthcare, 1000 generations

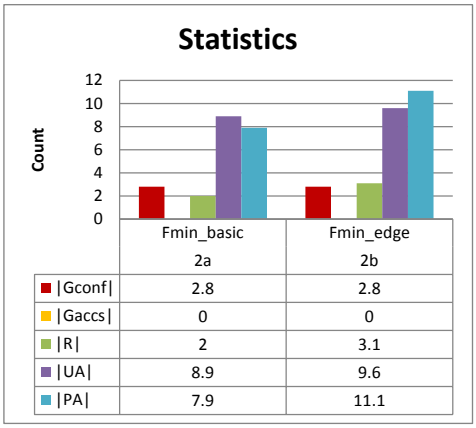


(d) F^{min}_{edge}, Healthcare, 1000 generations

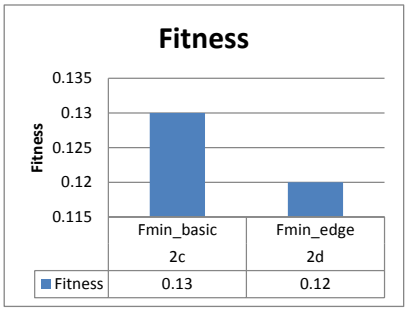
Figure 6.5: EXPERIMENT2: Fitness graphs of ten experiments with the Evo-RoleMiner with setup in Table 6.4 for Dataset1 and Healthcare dataset. The values for Fitness (y-axes) are the average minimum of all experiments over generations (x-axes)



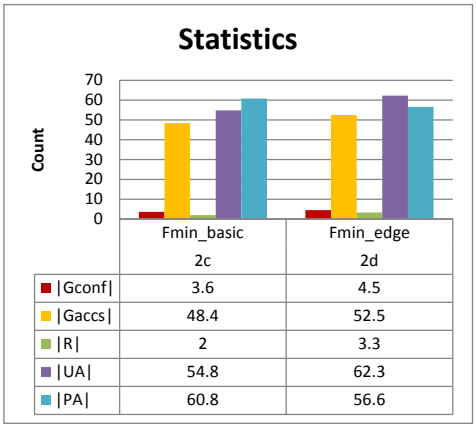
(a) Dataset1



(b) Dataset1



(c) Healthcare



(d) Healthcare

Figure 6.6: EXPERIMENT2: Statistics of ten experiments with the Evo-RoleMiner with setup in Table 6.4 for Dataset1 and Healthcare dataset. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last generation of all experiments.

The average fitness value in the last generation in the experiments with fitness function F^{min}_{edge} are better than in the experiments with fitness function F^{min}_{basic} (see Figure 6.6). A T-Test[1] on the results of the experiments[2] confirms that this difference is statistically

[1] A T-Test is used to test the null hypothesis that the means of two populations are equal

[2] See Appendix D.2 for data

different (P-Value is less than 0.0001).

For the experiments with fitness function F^{min}_{basic} on Dataset1 (see experiment 2a in Figure 6.6b), the average minimum number of roles in the results are two, which is according to the objective of minimizing roles in F^{min}_{basic} a good result. But it is known that a better result with a role count of four can be achieved (see Figure 6.1). The discovered small amount of roles has the drawback that the other objectives are harder to reach. The number of confidentiality violations has not reached zero in any of the ten experiments, where it is known that a solution exists with no violations when the role count is four.

Also the average minimum number of roles for the Healthcare dataset (see experiment 2c in Figure 6.6d) seem to be very low with two roles compared to results in other papers where the lowest role count is 14 [10][30]. The number of violations for the Healthcare dataset are not optimal, in the sense that when the according role model is applied some users will gain access rights and others will loose some. If this is acceptable is hard to judge since the details of the Healthcare dataset are unknown, e.g. how critical certain permissions are and how much noise is given.

The experiments with fitness function F^{min}_{edge} show a slightly higher amount of the average minimum role count (see experiment 2b and 2d in Figure 6.6). Comparing the minimum amount of roles of each experiment in experiment 2a and 2b with an unpaired T-Tests confirm that the difference is statistically significant (P-Value is less than 0.0001). The same is confirmed for the experiments in experiment 2c and 2d. This seem reasonable, since the fitness function F^{min}_{edge} tries to minimize $|UA|$ and $|PA|$ and only has an indirect focus on minimizing the role count $|R|$.

It can be noted that the average minimum count of $|UA|$ and $|PA|$ in the experiments with fitness function F^{min}_{basic} is lower than in the experiments with fitness function F^{min}_{edge}, although the latter functions' objective is to minimize $|UA|$ and $|PA|$. An explanation for this would be that the first fitness function encourages the reduction of the number of roles, which than lowers the maximum amount of possible $|UA|$ and $|PA|$.

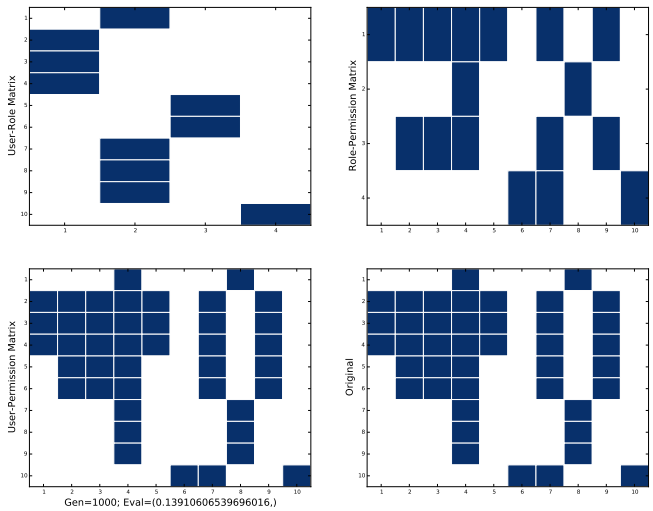


Figure 6.7: EXPERIMENT 2b: Example of fitness-optimal solution role model resulting of Evo-RoleMiner with Fitness function F_{edge}^{min} on Dataset1 with setup in table 6.4. From upper left to lower right: User-Role Matrix (Rows: Users, Columns: Roles), Role-Permission Matrix (Rows: Roles, Columns: Permissions), Resulting User-Permission Matrix (Rows: Users, Columns: Permissions), Original User-Permission Matrix from Input (Rows: Users, Columns: Permissions). A blue box stands for an assignment.

An optimal solution can not be reached if the role count or the assignment counts are zero. An optimal solution in regards of the violations of Dataset1 is not reached in the ten experiments with fitness function F_{basic}^{min} and is reached once in ten experiments with fitness function F_{edge}^{min} as seen in Figure 6.7. For the Healthcare dataset no optimal solution in regards of the violations is reached in any experiment.

There are several elements in the EA, which can influence the number of roles. One element is the mutation and the probability for role removal. The fitness function also has a big impact. While in F_{basic}^{min} it is obvious that the weight for the Role count $|R|$ has an impact, Experiment 1 (see section 6.2) showed that also the number of $|UA|$ and $|PA|$ have an indirect influence on the role count. Therefore the weight for $|UA| + |PA|$ in F_{edge}^{min} can influence the role count. Furthermore also the violation count (G_{conf} and G_{accs}) have an indirect impact on the role count, which was also found in the experiments of Experiment 1. A third element, which can lead to reduction of roles is the local optimization after a mutation and a crossover (see section 5.3.3).

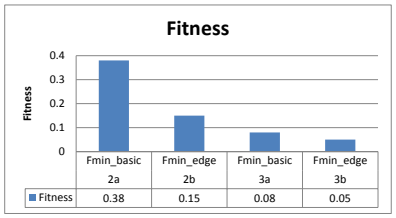
To relax this strong bias towards a small role count, the probability for role removal is set lower and the probability for adding new roles is set higher for Experiment 3. Furthermore the weights w_1 for the role count and the assignments counts in the fitness functions F^{min}_{basic} and F^{min}_{edge} respectively are adjusted.

6.4 Experiment 3 - Adjusting Parameters

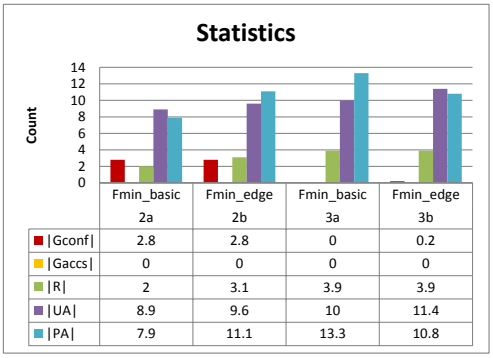
In this set of experiments the fitness functions F^{min}_{basic} (5.7.2) and F^{min}_{edge} (5.7.6) are tested again on the Dataset1 (Experiment 3a and 3b) and the Healthcare dataset (Experiment 3c and 3d), but with an adjusted setup as shown in table 6.5. The setup is relaxing the objective for minimizing the role model complexity (role count, assignment counts).

Parameter	**Value**
Generations	1000
Population	1000
CXPB	0.25
MUTPB	0.25
MUTPB-Type1: Add role	0.5
MUTPB-Type2: Add User	0.25
MUTPB-Type3: Add Permission	0.25
MUTPB-Type4: Remove Role	0.1
MUTPB-Type5: Remove User	0.25
MUTPB-Type6: Remove Permission	0.25
Tournament size	2
Local optimization	True
Weights for Fitness Function	0.1, 1.0, 1.0

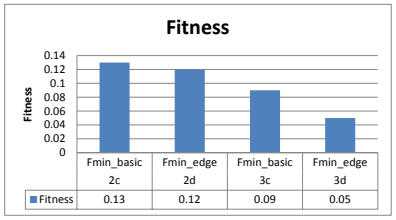
Table 6.5: EXPERIMENT 3 setup



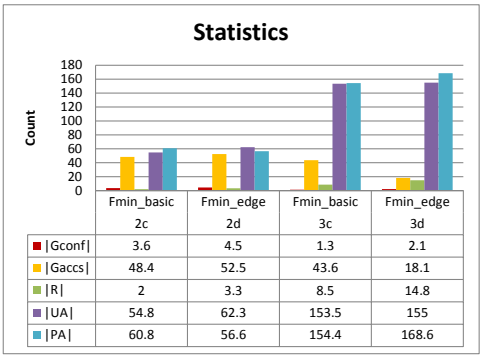
(a) Dataset1



(b) Dataset1



(c) Healthcare



(d) Healthcare

Figure 6.8: EXPERIMENT3: Statistics of ten experiments with the Evo-RoleMiner with setup in Table 6.5 for Dataset1 and Healthcare dataset. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last Generation of all experiments.

The results can be seen in experiment 3a-d in Figure 6.8. The results show that solutions with better fitness can be found compared to solutions in Experiment 2 by adjusting the parameters. There are many parameters, which can be adjusted and other settings might lead to even better results. One of the individuals with the best fitness solution of the experiments with the Healthcare dataset can be seen in Figure 6.9.

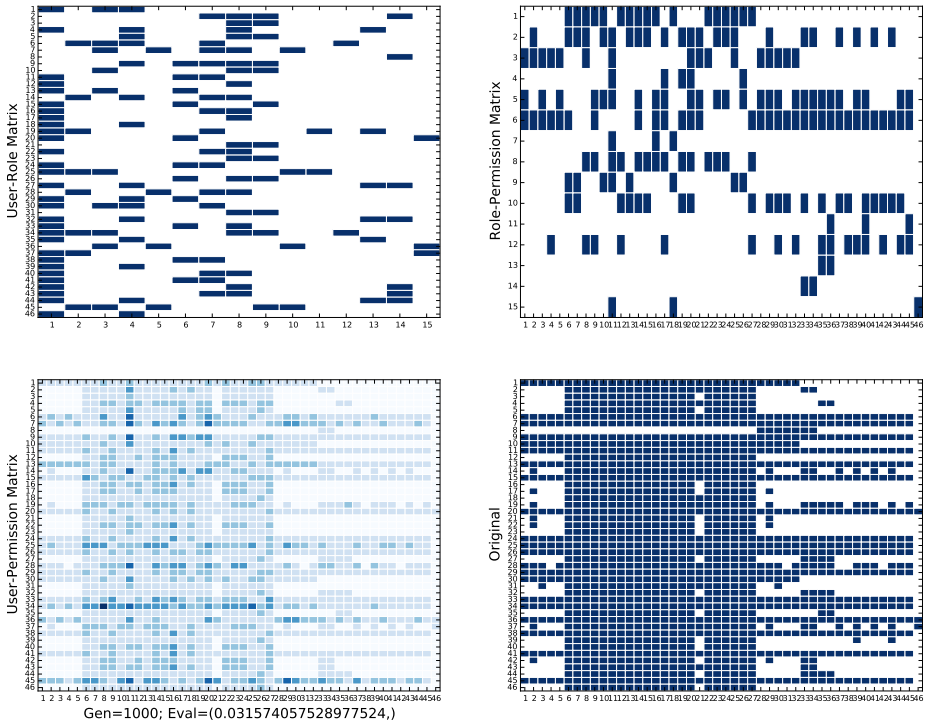


Figure 6.9: EXPERIMENT 3d: Example of role model resulting of Evo-RoleMiner with Fitness function F_{edge}^{min} on the Healthcare dataset with setup in Table 6.5. The role model has a role count of 14, one confidentiality violation, 23 availability violations, 152 user-role- and 192 role-permission-assignments. The fitness measure of the individual is 0.032. From upper left to lower right: User-Role Matrix, Role-Permission Matrix, Resulting User-Permission Matrix, Original User-Permission Matrix from Input. A blue box stands for an assignment. The darker the blue the more user-role- and role-permission assignments causing the user-permission assignment.

A critical point is the diversity of the individuals in the populations in order to be able to find better solutions. When looking on the role count diversity it can be noted, that the individuals of a population quickly get the same role count size. The boxplot in Figure 6.10 visualizes the role count diversity of the individuals in a population in different generations in one of the experiments (The boxplots in the other experiment look similar). It is therefore not surprising that it is difficult for the Evo-RoleMiner to find an individual with a minimum amount of violations. Therefore the Evo-RoleMiner*M* is tested in the next experiments with the expectation that more diversity can be achieved.

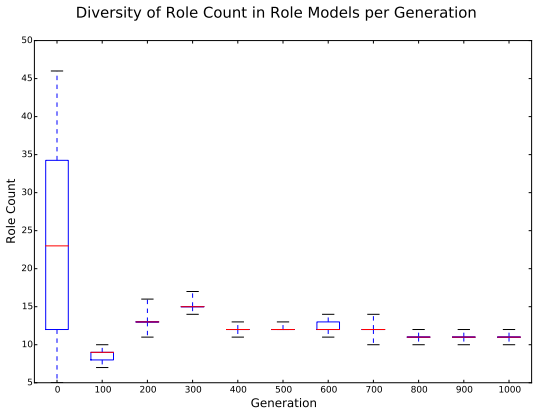


Figure 6.10: EXPERIMENT 3c: Example boxplot of role count diversity of individuals of a population in different generations. Experiment was executed on the Healthcare dataset with the Evo-RoleMiner and fitness function F^{min}_{basic}.

6.5 Experiment 4 - Multiobjective EA

In this experiment set the three versions of the Evo-RoleMiner*M* are tested (see section 5.1). To counteract on the diversity problem and the strong bias of the complexity measure (role count, assignments counts) in the previous experiments, the Evo-RoleMiner*M* is tested on the Dataset1 (Experiment 4a-c) and the Healthcare dataset (Experiment 4d-f). For the objectives ”Confidentiality Violations” and ”Availability Violations” are chosen, since as shown in Experiment 1 the objectives are conflicting (see section 6.2). The setup can be seen in Table 6.6.

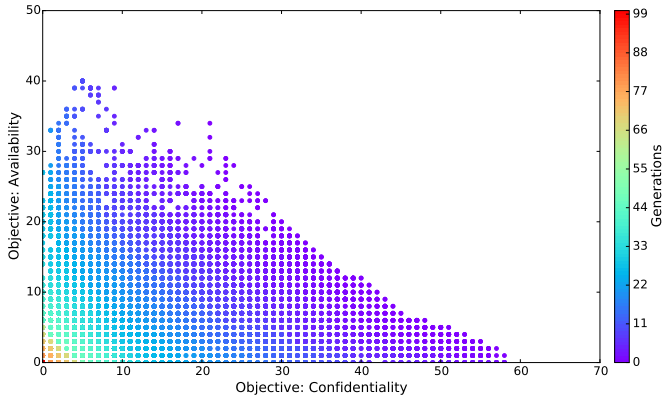
Parameter	Value
Generations	100 / 1000 / 2000
Population	1000
CXPB	0.25
MUTPB-Type1: Add role	0.25
MUTPB-Type2: Add User	0.25
MUTPB-Type3: Add Permission	0.25
MUTPB-Type4: Remove Role	0.25
MUTPB-Type5: Remove User	0.25
MUTPB-Type6: Remove Permission	0.25
Local optimization	True
Objective 1	Confidentiality / Violation Count
Objective 2	Availability / Assignment Count

Table 6.6: EXPERIMENT 4 setup. For experiments 4a-c a generation limit of 100 is chosen, while for experiments 4d-e the generation limit is 2000, experiment f has a limit of 1000. Objective 1 and 2 are confidentiality and availability violations respectively for experiments 4a,b,d,e. For Experiment 4c and f Objective 1 is the sum of confidentiality and availability violations and Objective 2 is the sum of user-role- and role-permission assignments.

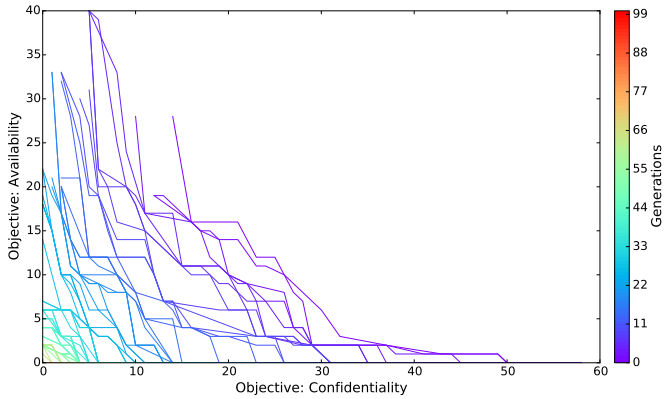
All results can be seen in the Appendix D.4. The most interesting results are presented in the following subsections.

6.5.1 Evo-RoleMiner*M* with NSGA-II

Figure 6.11 on the next page shows the development of the population in one of the experiments with Dataset1 (for the Healthcare dataset see Appendix D.4.4). It can be seen that the individuals in each generation have a wide spread and converging near a pareto front. Since the Dataset1 yields a solution where both objectives can be met, the pareto front consist of one point at (0,0) if the solution is found. The objectives of confidentiality and availability violations are only conflicting until the solution is found.



(a)



(b)

Figure 6.11: EXPERIMENT 4a: Example solution of the experiments with Evo-RoleMinerM (based on NSGA-II) on Dataset1. (a) Fitness of individuals over several generations (b) Pareto fronts of each generation.

An example boxplots on the diversity of the role count of individuals of a population is shown in Figure 6.12 for the Healthcare dataset. Compared to the role count diversity in the experiments with the Evo-RoleMiner (see Figure 6.10), the Evo-RoleMiner*M* allows more diverse role models with different role count. This is owed to the fact that the confidentiality violation objective tends to prefer less roles, while the availability violation objective can be probably better achieved if more roles are existing (see Experiment 1 in section 6.2).

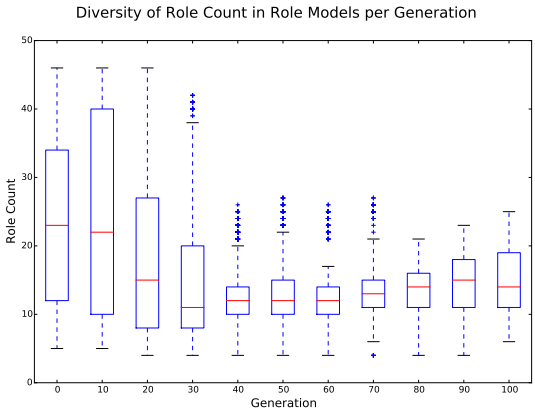
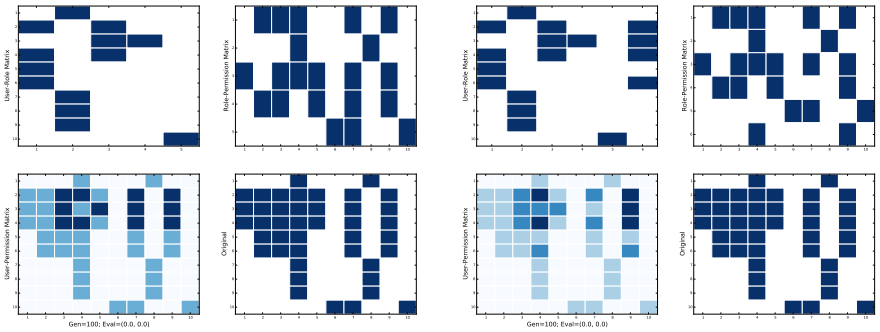


Figure 6.12: EXPERIMENT 4d: Example boxplot of role count diversity of individuals of a population in different generations with Evo-RoleMinerM (based on NSGA-II) on the healthcare dataset.

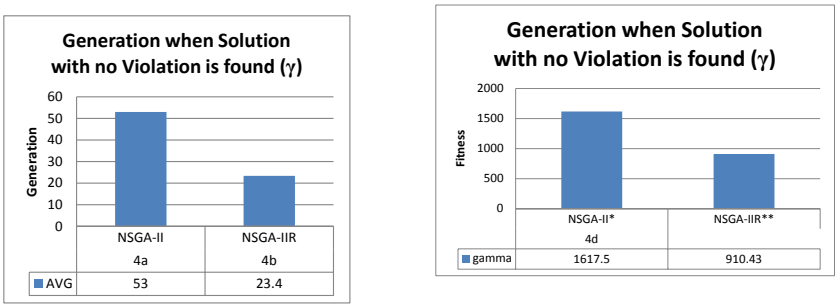
6.5.2 Evo-RoleMiner*M* with NSGA-II*R*

Role models can have the same fitness values for both objectives, although their genotype is different (compare for example the role models in Figure 6.13 on the next page). If several individuals have the same fitness and are in the same non-domination level, they get a crowding distance value of zero (see section 3.2.3). This would imply that role models, who share the same fitness, are favoured in the selection mechanism. Due to this instability in the NSGA-II algorithm, experiment 4b and e are running on a version of the Evo-RoleMiner*M* based on the improved NSGA-II*R* algorithm from Fortin et al.[13]. There is a chance that the convergence and diversity of the individuals can be improved, such that the Evo-RoleMiner*M* can find a solution even faster.



(a) *(b)*

Figure 6.13: EXPERIMENT 4a: Example role models resulting from an experiment on the Evo-RoleMinerM (based on NSGA-II) on Dataset1. For each role model from upper left to lower right: User-Role Matrix, Role-Permission Matrix, Resulting User-Permission Matrix, Original User-Permission Matrix from Input. A blue box stands for an assignment. The darker the blue the more user-role- and role-permission assignments causing the user-permission assignment.



(a)

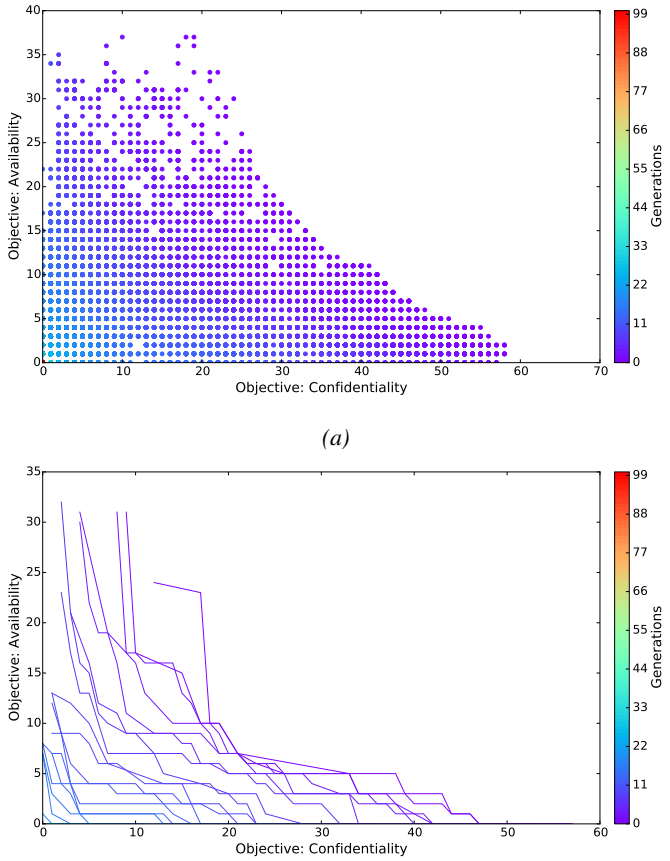
*(b) *NSGA-II: Solution found in 4 experiments; **NSGA-IIR: Solution found in 7 experiments)*

Figure 6.14: Comparison of Evo-RoleMinerM based on NSGA-II and on based on NSGA-IIR. It shows the average generation when a solution with no confidentiality or availability violations is found. (a) for Dataset1 (b) for Healthcare Dataset

In case of Dataset1 (Experiment 4b) it can be seen that the Evo-RoleMiner*M* based on NSGA-II*R* gives an improvement, since in average a solution with no violations is found after 23.4 generations, while in case of the Evo-RoleMiner*M* based on the

traditional NSGA-II, the average generation when a solution is found is 53 (see Figure 6.14a). Also a T-Test confirmed a statistical significant difference (P-Value in T-Test is less than 0.0001).

In case of the Healthcare dataset (Experiment 4e) after 1000 Generations no solution with no violations is found with Evo-RoleMiner*M* based on NSGA-II. Applying Evo-RoleMiner*M* based on NSGA-II*R* a solution with no violations could be found in five of the 10 experiments, where the average generation is 757.8.



(a)

(b)

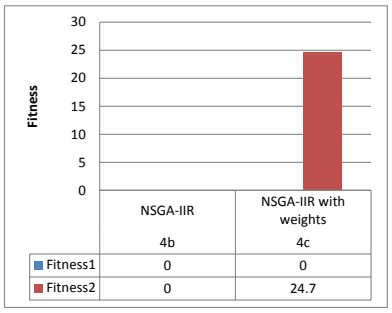
Figure 6.15: EXPERIMENT 4b: Example solution of the experiments with Evo-RoleMinerM based on NSGA-IIR on Dataset1. (a) Fitness of individuals over several generations (b) Pareto fronts of each generation.

6.5.3 Evo-RoleMiner*M* with NSGA-II*R* with weights

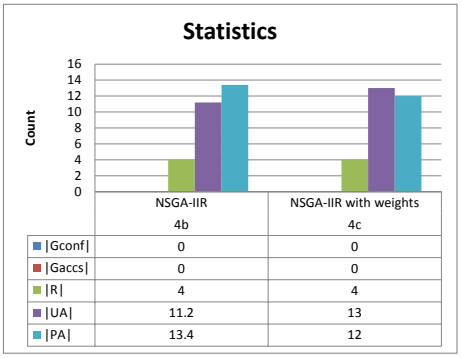
The objectives chosen for experiment 4a,b,d,e are only confidentiality and availability violations. Although the local optimization (see section 5.3.3) is preventing roles with the same user- or permission-assignments, the Evo-RoleMiner*M* cannot distinguish between the complexity of role models, which can reconstruct the original access control policy UPA. The Basic-RMP optimizes the complexity regards the role count, while the Min-Edge-RMP optimizes towards a minimum user- and permission assignment count.

In experiment 4c and f a third version of the Evo-RoleMiner*M* is tested, where one objective is the minimization of the combined violations count $|G_{conf} + G_{accs}|$ and the second objective is the minimization of user- and permission assignment count $|UA + PA|$. Since it is known from previous experiments, that the complexity measure has a strong influence the third version of the Evo-RoleMiner*M* is chosen, which allows to lower the selection pressure on the second objective by setting a probability (see section 3.2.4). The probability for the second objective is set to 0.8.

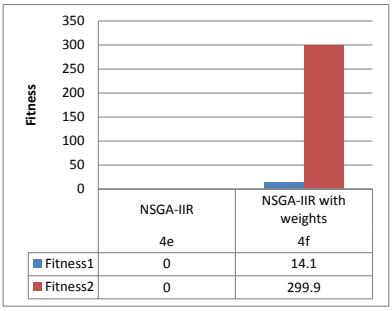
The results can be seen in Figure 6.16 in comparison with the Evo-RoleMiner*M* based on NSGA-II*R* without weights and objectives "Confidentiality" and "Availability". In Figure 6.17 the development of one of the ten experiments with Evo-RoleMiner*M* based on NSGA-II*R* with weights is illustrated.



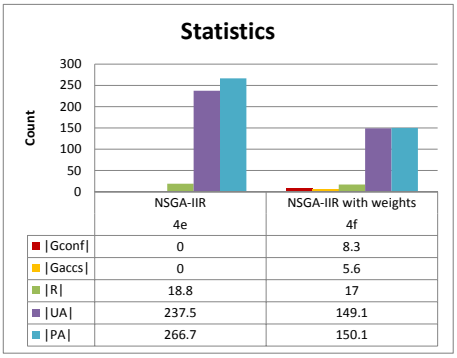
(a) Dataset1



(b) Dataset1



(c) Healthcare

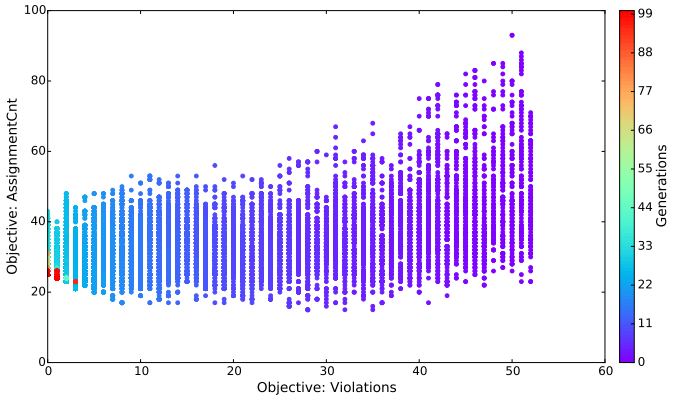


(d) Healthcare

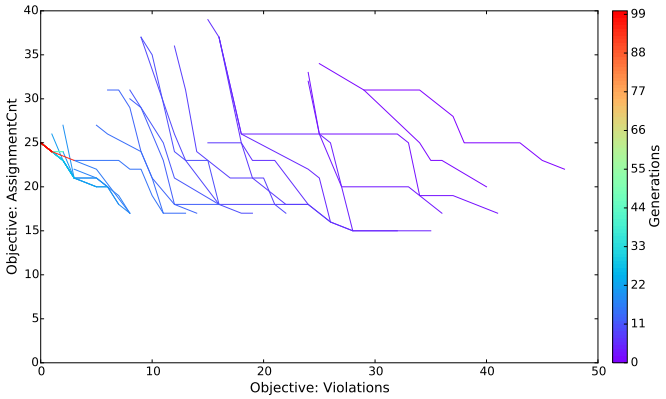
Figure 6.16: EXPERIMENT4b,c,e,f: Statistics of ten experiments with the Evo-RoleMinerM with setup in Table 6.5 for Dataset1 and Healthcare dataset. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last generation of all experiments.

The results show that the third version of Evo-RoleMiner*M* less likely finds a violation-free role model than the second version of Evo-RoleMiner*M*. On the other hand on the experiments on the Healthcare dataset, it can be seen that the count of user-role- and role-permission assignments is significantly lower when using the third version of Evo-RoleMiner*M* (see Figure 6.16).

T-Test missing



(a)



(b)

Figure 6.17: EXPERIMENT 4c: Example of result of the experiments with Evo-RoleMinerM version 3 on Dataset1. (a) Fitness of individuals over several generations (b) Pareto fronts of each generation.

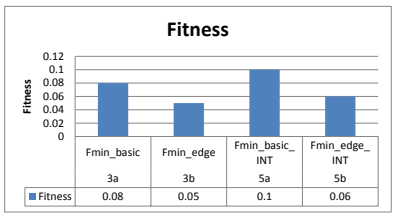
The development of the pareto front over the generations (see Figure 6.17) with the third version of the Evo-RoleMiner*M* looks different than development of the pareto fronts in the other versions of the Evo-RoleMiner*M* (see Figure 6.11 and 6.15). This is due to the stochastic version of pareto dominance (see section 3.2.4).

6.6 Experiment 5 - Interpretability Objective

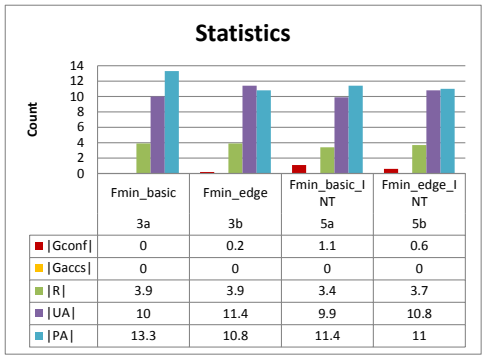
When looking back on the solution with an optimal fitness found in one of the experiments in Experiment 2, it can be noted that the matrix decomposition of the resulting UPA (see Figure 6.7) looks different than the matrices Dataset1 has been constructed on (see Figure 6.1). It shows that several solutions for a matrix decomposition can exist. To distinguish them another measure is needed.

One reason why bottom-up role mining is criticized, is that a solution might be found based on given technical data (user-permission-assignments), but it is not necessary feasible for the business. Therefore the Interpretability objective and the fitness functions $F^{min}_{basic\_INT}$ (5.7.8) and $F^{min}_{edge\_INT}$ (5.7.8) are observed in this experiment. The Experiment is executed with the Evo-RoleMiner.

For the experiments the setup in table 6.5 is taken and executed on the Dataset1. The weight for the interpretability measure is set 1.0. The results can be seen in Figure 6.18.



(a) Dataset1



(b) Dataset1

Figure 6.18: EXPERIMENT5: Statistics of ten experiments with the Evo-RoleMiner with setup in Table 6.4 for Dataset1. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last Generation of all experiments.

The same experiments have been executed on the healthcare dataset with the user attribute information provided by Xu & Stoller[44]. The experiments have been aborted

after the fitness calculation of the first generation could not be finished in a decent time. There are 20 attributes for the users in the healthcare dataset. For the interpretability measure rules are generated, where $2^X - 1$ combinations of X attributes are created. While for Dataset1 with three user attributes seven combinations are created and tested, the healthcare dataset requires 1,048,575 combinations for 20 attributes. The rule induction gets too expensive to finish in reasonable time.

6.7 Experiment 6 - Scalability

So far the experiments have been only executed on Dataset1 and the Healthcare dataset. In this experiment the Evo-RoleMiner*M* is also applied on the Domino dataset (see Table 6.1) and the average time needed for a generation is compared to experiments with the same settings with Dataset1 and the Healthcare dataset. The setup used can be seen in Table 6.6, where the generation size is set to 100 and the objectives "Confidentiality" and "Availability" are chosen for all experiments. An experiment on one dataset is executed 10 times. The specification of the machine, the experiments have been executed on, can be seen in Appendix B.1.

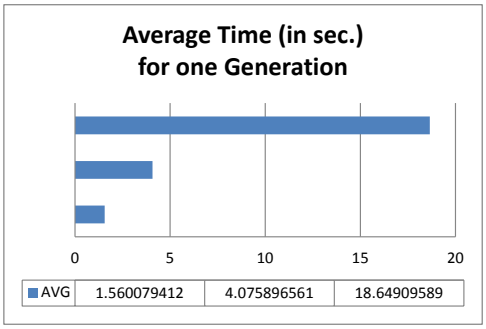


Figure 6.19: EXPERIMENT 6: Comparison of average time (in seconds) needed for a generation on Dataset1, Healthcare dataset and Domino dataset.

When applying the Evo-RoleMiner*M* on the Domino dataset, the time needed for a generation quickly rises comapared to the smaller Dataset1 and the Healthcare dataset.

6.8 Summary of Results

In Experiment 2 it is shown that the Evo-RoleMiner for the Min-Edge-RMP (fitness function F_{edge}^{min} (5.7.6)) performs better than for the Basic-RMP (F_{basic}^{min} (5.7.2)).

Experiment 3 demonstrates that parameter adjustments can reveal even better results. But the diversity of the role models in the population according role count gets very low after few generations and it is difficult for the Evo-RoleMiner to find a better solution since the search space gets limited to role models with a specific role count.

Experiment 4 demonstrates that a MOEA can help in the search since a wider solution space with role models of different role count is discovered. The completeness measure (confidentiality and availability violations) is well suited for the objectives in a MOEA, due to its conflicting complexity measure supporters: Higher vs. lower role count and assignment count (see Experiment 1). But the complexity measure has to be also an objective in order to remove unnecessary duplicate user-permission-assignments in the resulting role model (see section 6.5.3).

The individuals in the MOEA are more diverse regarding the role count compared to the individuals in the EA. The MOEA based on NSGA-II*R*[13] performs better than the MOEA based on NSGA-II[7]. The MOEA based on NSGA-II*R* is extended with a stochastic version of pareto dominance, which allows to set lower the selection pressure for the second objective[2].

In Experiment 5 it can not be demonstrated that the added interpretability measure improves the Evo-RoleMiner. The performance of the implemented rule induction is not good enough to be tested on bigger datasets with several user attribute information.

In Experiment 6 the Evo-RoleMiner*M* is tested on the bigger Domino dataset and shows that the average time for one generation rises with the dimension of the dataset.

Chapter 7

Discussion

The thesis question of how a role model can be mined from an access control policy with an evolutionary computation approach has been investigated by analyzing the domain and providing an EA and three implementations of a MOEA. Several different fitness functions are exchangeable and several parameters can be set in the implementation and have been tested.

The results show that an EA and an MOEA can find solutions for the role mining problems for small access control policies. The improvement by Fortin et al.[13] for the NSGA-II algorithm could be applied, resulting with a better performance than the MOEA based on classic NSGA-II[7]. The usage of a stochastic version of pareto dominance[2] relaxes the second objective in the improved MOEA. This allows to include the complexity measure as objective with a lower selection pressure on. The resulting role models can achieve less duplicate user-permission assignments than in the resulting role models of the previous versions of the MOEA.

The question if business information data can improve the evolutionary computation approach can not be confirmed. Approaches for measuring interpretability (”meaningfulness”) of roles are given (see section 5.5), but either not implemented or the complexity is too costly to be tested.

7.1 Contribution to current research

This thesis contributes with an analysis of the resulting role models of using an EA for Role Mining. Furthermore different MOEAs are applied for Role Mining and show more promising results than using an EA with a scalarization of multiple objectives in one fitness function. The EA and the MOEAs are tested on one of the real datasets commonly used by role mining research, which makes the result more comparable to other work. Furthermore two interpretability measures for roles based on clustering and on classification are suggested.

7.2 Reflection

The following reflections on the approach are splitted into topics.

Parameters

After re-adjusting parameters for the EA and MOEAs several times a role model of the given access control policy with few violations can be found with the Evo-RoleMiner. However, there exists little knowledge on how to choose the weight given only the access control policy as input data. The density of assignments in the given access control policy might help in the decision, but there is no concrete rule existing. For example, the weight for the number of roles in the fitness function F_{basic}^{Min} had to be set very low or even to a negative value (which encourages a maximization of role numbers) in the Evo-RoleMiner for the synthetic dataset to be able to find a role model, which does not violate the original access control policy (see section 6.4).

The evolutionary algorithms have many parameters and it is difficult to find a good configuration without pre-knowledge. The parameters have to be tuned by a trial-and-error approach, which is very time-consuming. A solution for this would be self-adaptation, where the parameters of the EA are evolving with the individuals in a population[9]. For example if the average role count of the role models (indivduals) get close to the minimum role count of the role models, the probability of the mutation of adding roles can be set higher.

Variation operators

The mutation in the EA and MOEAs chosen is highly random and chances are low that the mutation help to break out of a premature convergence to a local optimum. A mutation of adding or removing a user or permission in a role model might not be strong enough to lead to a better fitness of the offspring. This can be especially the case if user-permission matrix of the given access control policy is very sparse. Also the mutation of adding roles is generating completely random roles, where the algorithm might have discovered before, that certain combinations of users and permissions in a role are negatively influencing the fitness. A more guided mutation might improve the implemented EA. To elaborate on the thought, the probabilities for the different mutation types can be exchanged by an intelligent mutation, which chooses the mutation according to the current state.

The local optimization after a mutation or crossover (see section 5.3.3) is helping to avoid duplicates. On the other hand it misses out the chance that different variation operators on the role models with same user- or permission list can be explored, which might be close to a role, which is improving the role model heavily. Also the local optimization is currently first combining roles with the same user list. When a role with identical user and permission list occurs, it is not checked if a permission combining of the roles lead to a better role model. On the other hand the chances that this conflict occurs gets more rare the higher the dimension of users and permissions are.

Scalability

The experiments in this thesis are executed on very small datasets, such that visualizations of resulting role models can be used to support argumentations. But they might not reflect cases, which enterprises often have to face. Enterprises can have thousands of users and millions of permissions. The access control policies might be separated into smaller chunks, e.g. to only consider users of one department and find their according role model or consider only one application and its permissions for constructing a role model for that application. Finding a good role model which spans every user and permission in the enterprise on the other hand, might not be manageable in a decent amount of time.

Experiment 6 showed how quick a bigger dataset rises in computation complexity. The results in Saenko & Kotenko[35] measure how many generations are needed to find a role model, which describes the given access control policy. The computation time achieved for different datasets of different dimensions stated in the paper could not be achieved by the approach of this thesis. This might be due to the data, available computation resources or the implementation.

Since the dimension of real datasets are often much bigger than the datasets tested in this thesis, it can be argued that the implementations are unlikely to be used in practice when other role mining techniques perform faster and achieve similar results. A parallel EA could help in this regard. The basic idea of a parallel EA is to have multiple populations in parallel. Each population is evolved separately, but after a certain amount of generations, selected individuals from each population get exchanged with the individuals of an other population.[9]

Multi-Objectives

The role mining problems have several objectives and which objectives are considered depends on the particular case. Since the EA tend to converge to objectives, which optimal state can be more easily reached (e.g. minimizing role count), MOEAs seem to be the right approach, since they are searching for the trade-off solutions between several objectives and the individuals in a population are more diverse. However the problem of lacking diversity is not solved by the MOEA. It still can be difficult for the algorithm to break out of a niche, especially when the optimum of the objectives can not be equally easily achieved, which is the case of objective ”Role Count” and ”Violations”. While for the first objective only the gene (role) size has to be reduced, the second objective requires the right setting of the genes (user and permission list). This could speak for a disadvantageous representation.

The experiments only show the MOEA with maximum two objectives. This is due to the fact that the NSGA-II with stochastic pareto dominance has not been successfully tested with three objectives. The objectives for complexity require a relaxation, since they have a strong influence.

Interpretability

To measure the comprehension of a role model is a difficult task, since in the end only system administrators can judge if a role model is in their eyes comprehensive or not. By measuring the interpretability of roles by how predictive they are by using business information like user attributes is a first step. In theory it could help the evolutionary algorithm of choosing a role model with better predictive roles than a role model with less predictive roles, if the completeness and complexity measure is the same for both role models.

The measure for the role model is just an average of the interpretability of the roles it contains. That means that a highly interpretable role can get lost with the role model, since other roles in the role model are less interpretable. Therefore guiding the variation operators towards more interpretable roles in a role model might be a better approach than using an average role interpretability measure within the fitness function.

Compared to the other objectives in the fitness function, the interpretability measure requires the calculation of rules, which then get measured. This quickly gets computationally expensive when the dimensions of the dataset or the EA parameters, e.g. population and generation size, rises. In particular if the rule induction algorithm is deterministic. An approximation algorithm with lower time complexity, that produces almost as good rules for the measure of role model interpretability, could improve the performance. But still the objective requires some greater computation than for example just getting the length of a data object, which is the case of the role count objective. An approach could be to store the interpretability measure with the role model and variation operators are calculating the interpretability of a role model offspring. By this, the fitness function only needs to read the measure from the role model.

In the experiments it is looked on how well the original intended role model can be obtained by the Evo-RoleMiner and Evo-RoleMiner*M* when using the synthetic dataset. Since evolutionary algorithms are a stochastic search, they do not guarantee to find an optimal solution in a finite amount of time. Different role models can be a solution. For real datasets an evaluation of the resulting role models can be only executed by domain experts and the administrators, who have to work with the role model.

Also when the business information can be included into the role mining algorithm, in practice the resulting roles or role models have to be analysed and optimized in a post-process. The inclusion of business measure can only help to produce a better starting point for the post-process.

After all an evolutionary computation approach can be still attractive for role mining. But there is still research necessary to get to a state, which is promising for the usage in practice. Future work in this field is suggested in the next chapter.

Chapter 8

Future Work

The experiments in this thesis have only been executed on two small datasets. To get a better understanding on how to set parameters in the suggested EA and MOEA several datasets of different dimensions (user and permission size) and density (number of assignments) should be tested. Also applying evolutionary algorithms on noisy access control policies has to be investigated. It is very likely that given access control policies have a lot of over-entitlements and some under-entitlements. It would be interesting to see how an EA can reveal these false entitlements.

There are several parts in the EA and MOEA, which can be further improved, for example to develop more intelligent variation operators. A different representation of individuals might reveal new opportunities. The approach in this thesis builds strongly on the related boolean matrix decomposition problem (see section 2.5.2). A different approach and representation might be more suitable. For example a representation, which is based on user attributes to include the interpretability of roles into the role mining.

A novelty search could counteract on limited diversity in a population. The basic idea is to explicitly reward individuals, which are diverging from previous individuals. By this a greater search space can be reached more efficiently. The measure for novelty could be an additional objective in the fitness function.

A motivation for the approach was also to incorporate human feedback in the EA. An

interactive evolutionary computation (IEC)[38] could be an interesting topic for further research. Domain experts could navigate the algorithm to an optimal result, accepted by the business, by selecting the individuals for reproduction. Such an approach might consider a different representation of individuals, since it is difficult for a human to evaluate a whole role model. The evaluation of single roles on the other hand can be accomplished by experts.

Another idea is the usage of Co-Evolution. In Co-Evolution several separate populations are influencing the fitness of the individuals of each other. This idea is derived by co-evolution in biology, where the evolution of a species can influence the evolution of another species and vice versa. Instead of that individuals represent RBAC models, the individuals represent roles. The motivation behind this approach is to evolve candidate roles instead of a complete role model. For example each population consists of roles as individuals. Each population might only consider a specific segment of the access control policy, e.g. only users of a department or only permissions of an application. Roles from each population are then getting into a trial, where they should form a role model, which then gets evaluated. This technique is inspired by the Symbiotic Adaptive Neuro-Evolution (SANE) and Enforced Sub-Populations Method (ESP) approach in Gomez & Miikkulainen[17].

To find a measure for the comprehension of a role model or the interpretability of roles is an interesting topic. The initial ideas suggested in section 5.5 could be investigated further. A more detailed research on how such an measure could be integrated into a role mining algorithm could be done.

Chapter 9

Conclusion

In this thesis two types of role-mining algorithms based on evolutionary computation have been developed, which generate role models from scratch by given access control policies. The first evolutionary algorithm developed is partially based on the work from previous research[35] and aims for the objectives of role mining problems. The different components of the algorithm are explained and evaluated. A second approach is a multi-objective evolutionary algorithm for role mining based on NSGA-II[7]. The improvement by Fortin et al. for the NSGA-II[13] could be successfully applied. The usage of a stochastic version of pareto dominance[2] has been applied on the improved algorithm to relax the objective of minimizing role model complexity. In experiments the different algorithms and objectives are tested on a synthetic dataset and the commonly used Healthcare dataset in role-mining research[10]. The results show that the algorithms can find solutions regarding to the chosen objectives. The resulting role models require further optimization by domain experts before they can be applied in practice. Furthermore suggestions for measuring ”role interpretability” are made, which are based on data mining techniques.

For future work different directions of evolutionary computation are suggested for a role mining algorithm. For example Co-Evolution and Interactive Evolutionary Algorithms can contribute to involve expert knowledge into the algorithm to lead to more optimal role models.

Bibliography

[1] Ajith Abraham and Lakhmi Jain. *Evolutionary multiobjective optimization*. Springer, 2005.

[2] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society of London B: Biological Sciences*, 280(1755):20122863, 2013.

[3] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[4] Ed Coyne and Timothy R. Weil. Abac and rbac: Scalable, flexible, and auditable access management. *IT Professional*, 15(3):14–16, 2013.

[5] Edward J. Coyne, Timothy R. Weil, and Rick Kuhn. Role engineering: Methods and standards. *IT Professional*, 13(6):54–57, November 2011.

[6] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. Deap: A python framework for evolutionary algorithms. In *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '12, pages 85–92, New York, NY, USA, 2012. ACM.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comp*, 6(2):182–197, April 2002.

[8] Xuanni Du and Xiaolin Chang. Performance of AI algorithms for mining meaningful roles. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*, pages 2070–2076, 2014.

[9] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[10] Alina Ene, William Horne, Nikola Milosavljevic, Prasad Rao, Robert Schreiber, and Robert E. Tarjan. Fast exact and heuristic methods for role minimization problems. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 1–10, New York, NY, USA, 2008. ACM.

[11] David B. Fogel. The advantages of evolutionary computation. In *Biocomputing and Emergent Computation: Proceedings of BCEC97*, pages 1–11. World Scientific Press, 1997.

[12] Félix-Antoine Fortin, Simon Grenier, and Marc Parizeau. Generalizing the improved run-time complexity algorithm for non-dominated sorting. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 615–622, New York, NY, USA, 2013. ACM.

[13] Félix-Antoine Fortin and Marc Parizeau. Revisiting the nsga-ii crowding-distance computation. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 623–630, New York, NY, USA, 2013. ACM.

[14] Mario Frank. Role Mining. `http://www.mariofrank.net/rolemining.html`. [Online; accessed 30-October-2015].

[15] Mario Frank, Joachim M. Buhman, and David Basin. Role mining with probabilistic models. *ACM Trans. Inf. Syst. Secur.*, 15(4):15:1–15:28, April 2013.

[16] Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. A probabilistic approach to hybrid role mining. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 101–111, New York, NY, USA, 2009. ACM.

[17] Faustino J. Gomez and Risto Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1356–1361, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[18] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[19] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Arthur R. Friedman, Alan J. Lang, Margaret M. Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, and Scarfone Cybersecurity. Guide to attribute based access control (abac) definition and considerations (draft), 2013.

[20] Axel Kern, Martin Kuhlmann, Andreas Schaad, and Jonathan Moffett. Observations on the role life-cycle in the context of enterprise security management. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 43–51. ACM, 2002.

[21] Igor Kotenko and Igor Saenko. Improved genetic algorithms for solving the optimisation tasks for design of access control schemes in computer networks. *Int. J. Bio-Inspired Comput.*, 7(2):98–110, May 2015.

[22] Martin Kuhlmann, Dalia Shohat, and Gerhard Schimpf. Role mining - revealing business roles for security administration using data mining technology. In *Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies*, SACMAT '03, pages 179–186, New York, NY, USA, 2003. ACM.

[23] Michael Kunz, Ludwig Fuchs, Michael Netter, and Günther Pernul. Analyzing quality criteria in role-based identity and access management. In *Proceedings of the 1st International Conference on Information Systems Security and Privacy*, pages 64–72, 2015.

[24] Haibing Lu, J. Vaidya, and V. Atluri. Optimal boolean matrix decomposition: Application to role engineering. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 297–306, April 2008.

[25] Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Trans. Dependable Sec. Comput.*, 9(5):655–669, 2012.

[26] S. Mandala, M. Vukovic, J. Laredo, Yaoping Ruan, and M. Hernandez. Hybrid role mining for security service solution. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 210–217, June 2012.

[27] Jonathan D. Moffett. Control principles and role hierarchies. In *Proceedings of the Third ACM Workshop on Role-based Access Control*, RBAC '98, pages 63–69, New York, NY, USA, 1998. ACM.

[28] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with semantic meanings. In *Proceedings of the 13th ACM Symposium on Access Control Models and Technologies*, SACMAT '08, pages 21–30, New York, NY, USA, 2008. ACM.

[29] Ian Molloy, Hong Chen, Tiancheng Li, Qihua Wang, Ninghui Li, Elisa Bertino, Seraphin Calo, and Jorge Lobo. Mining roles with multiple objectives. *ACM Trans. Inf. Syst. Secur.*, 13(4):36:1–36:35, December 2010.

[30] Ian Molloy, Ninghui Li, Tiancheng Li, Ziqing Mao, Qihua Wang, and Jorge Lobo. Evaluating role mining algorithms. In *Proceedings of the 14th ACM Symposium on Access Control Models and Technologies*, SACMAT '09, pages 95–104, New York, NY, USA, 2009. ACM.

[31] Lionel Montrieux, Michel Wermelinger, and Yijun Yu. Challenges in model-based evolution and merging of access control policies. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution*, IWPSE-EVOL '11, pages 116–120, New York, NY, USA, 2011. ACM.

[32] Alan C O'Connor and Ross J Loomis. 2010 economic analysis of role-based access control. *NIST, Gaithersburg, MD*, 20899, 2010.

[33] Mohammad Rawashdeh and Anca Ralescu. Fuzzy cluster validity with generalized silhouettes. In *Proceedings of the 23rd Annual Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, Ohio, April*, pages 11–18. Citeseer, 2012.

[34] Igor Saenko and Igor Kotenko. Genetic algorithms for role mining problem. In *Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, PDP '11, pages 646–650, Washington, DC, USA, 2011. IEEE Computer Society.

[35] Igor Saenko and Igor Kotenko. Design and performance evaluation of improved genetic algorithm for role mining problem. In *Parallel, Distributed and Network-Based Processing (PDP), 2012 20th Euromicro International Conference on*, pages 269–274. IEEE, 2012.

[36] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The nist model for role-based access control: towards a unified standard. In *ACM workshop on Role-based access control*, volume 2000, 2000.

[37] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996.

[38] H. Takagi. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, Sep 2001.

[39] Gregory Tassey, Michael P Gallaher, Alan C O'Connor, and Brian Kropp. The economic impact of role-based access control. *Economic Analysis*, 2002.

[40] Jaideep Vaidya. Boolean matrix decomposition problem: Theory, variations and applications to data engineering. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 1222–1224. IEEE, 2012.

[41] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: Finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, SACMAT '07, pages 175–184, New York, NY, USA, 2007. ACM.

[42] Jaideep Vaidya, Vijayalakshmi Atluri, Qi Guo, and Haibing Lu. Edge-rmp: Minimizing administrative assignments for role-based access control. *J. Comput. Secur.*, 17(2):211–235, April 2009.

[43] Jaideep Vaidya, Vijayalakshmi Atluri, and Janice Warner. Roleminer: Mining roles using subset enumeration. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 144–153, New York, NY, USA, 2006. ACM.

[44] Zhongyuan Xu and Scott D. Stoller. Algorithms for mining meaningful roles. In *Proceedings of the 17th ACM Symposium on Access Control Models and Technologies*, SACMAT '12, pages 57–66, New York, NY, USA, 2012. ACM.

Appendices

Appendix A

Alternatives

A.1 Representation of a Role Model as Individual

A.1.1 Representation as Bit-string

The representation of individuals as bit-string is one of the simplest and is often mistakenly used in genetic algorithms[9]. A suggestion for a bit-string representation for a role model is derived from a combined user-role- (UA) and role-permission-matrix (PA). Figure A.1 shows an example of a combined UA- and PA-Matrix and an according bit-string representation.

The motivation of this initial idea is that the mutation operator can be easily executed by just flipping a bit within the bit-string. The mutation is therefore adding or removing a user or permission from a role. For the decoding two leading 8-bit integers can represent the number of users and permission. However, the problem with this representation is that the number of roles needs to be pre-defined, such that the bit-string can be decoded again. Alternatively another leading 8-bit integer can represent the number of roles in the role model. But varying the number of roles demands adding or removing bits in the bit-string for each user-role- and role-permission relation. Furthermore the information that a user or a permission is not part of a role seems to unnecessarily lengthens the bit-string, which can consume a lot of space for high-dimensional UA-

UA-Info	Role 1	Role 2	Role 3
User 1	0	0	1
User 2	0	1	0
User 3	1	0	0
User 4	0	1	0
User 5	0	1	0
User 6	0	1	1
User 7	0	1	0

PA-Info	Role 1	Role 2	Role 3
Permission 1	1	1	1
Permission 2	0	0	1
Permission 3	1	0	0
Permission 4	1	1	0
Permission 5	1	1	0

Bitstring:
00000111 | 00000101 | 001 | 010 | 100 | 010 | 010 | 011 | 010 | 111 | 001 | 100 | 110 | 110

Users # Permissions UA-Information PA-Information

Figure A.1: Example of a UA- and PA-Matrix with its bit-string representation

and PA-Matrices.

A.1.2 Representation as Multi-chromosomal Objects

In Saenko & Kotenko[35] two different representations are introduced, which are briefly described in section 4.2. With the first representation the authors suggest a multi-chromosomal representation where the number of roles becomes part of the search. An example can be seen in figure A.2. A drawback of this representation are unnecessary genes (roles) like the unnecessary information in the bit-string representation. Another disadvantage of the representation are complex crossover operations[35].

$$\text{Chr}(\mathbf{X}) = \left\langle \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\end{bmatrix}, \dots, \begin{bmatrix}0\\1\\0\\1\\1\\1\\1\end{bmatrix}, \dots, \begin{bmatrix}1\\0\\1\\0\\0\\1\\0\end{bmatrix} \right\rangle \quad \text{Chr}(\mathbf{Y}) = \left\langle \begin{bmatrix}1\\0\\1\\1\\1\end{bmatrix}, \dots, \begin{bmatrix}1\\0\\0\\1\\1\end{bmatrix}, \dots, \begin{bmatrix}1\\1\\0\\0\\0\end{bmatrix} \right\rangle$$

$$\text{Chr}(\mathbf{Z}) = \langle\ 1\ ,\ 0\ ,\ 1\ ,\ 0\ ,\ 1\ \rangle$$

Figure A.2: Example of a multi-chromosomal representation. Chromosome X represents the UA-Informationa and Chromosome Y the PA-Information. Chromosome Z determines if a role is active or passive. The columns in all chromosomes are consistent to roles. In this example roles 1, 3 and 5 are active.

Appendix B

Implementation

B.1 Implementation Information

For the implementation of the Evo-RoleMiner and the Evo-Roleminer*M* the programming language Python has been chosen. For rapid prototyping of the Evo-RoleMiner and the Evo-Roleminer*M* the Framework "Distributed Evolutionary Algorithms in Python (DEAP)"[6] has been used. The development and testing have been executed on a computer with the following specifications.

Table B.1: Computer specifications

Specification	
CPU	1.90GHz-2.49GHz
RAM	12 GB
Processor	Intel(R) Core(TM) i5-4300U
System type	x64-bit

B.2 Evo-RoleMiner Algorithm

In Algorithm 2 the pseudo-code of the Evo-RoleMiner is given. The algorithm starts with the initial generation and evaluation of the parent population P_p. Then, for each generation, individuals from P_p are selected and cloned as offspring population P_o. A loop over neighboring individuals in P_o is mating individuals $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ by crossover probability $CXPB$. The resulting children $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$ replace their respective parents in P_o. In a second loop over all individuals in P_o the individuals get mutated by mutation probability $MUTPB$. The resulting mutation is replacing the original individual. The resulting individuals in P_o can be generated from crossover only, mutation only, crossover and mutation, and reproduction according to the given probabilities $CXPB$ and $MUTPB$. The offspring population P_o is then getting the parent population P_p for the next generation, after each individual got evaluated.

Algorithm 2 Evolutionary algorithm for single objectives

```
1:  procedure EASINGLE(popSize, NGEN, CXPB, MUTPB)
2:      P_p = generatePopulation(popSize)
3:      evaluateIndividuals(P_p)
4:      solutionFound = False
5:      generation = 1
6:      while not solutionFound AND generation <= NGEN do
7:          P_p = select(population, k=length(P_p))
8:          P_o = clonePopulation(P_p)
9:          i = 0
10:         while i < length(P_o) − 1 do
11:             P_o[i],P_o[i+1] = crossover(P_o[i],P_o[i+1],CXPB)
12:             i += 2
13:         end while
14:         i = 0
15:         while i < length(P_o) do
16:             P_o[i] = mutation(P_o[i],MUTPB)
17:             i += 1
18:         end while
19:         P_p = P_o
20:         evaluateIndividuals(P_p)
21:         generation += 1
22:     end while
23:     return result
24: end procedure
```

B.3 Rule Induction

Algorithm 3 Rule induction algorithm for classifier-rules for roles

D = List of role members, which are not matched by current rule

RM = List of users, which are member of the role

$notRM$ = List of users, which are not members of the role

$rule$ = Current rule

max = Maximum allowed rule size (number of OR-connected rules)

$Sensitivity$ = Probability that rule matches a user in a user set

```
 1: procedure RULEINDUCTION(D,RM,notRM,rule,max)
 2:     allAttrSets = Create all combinations of attributes
 3:     sort(allAttrSets)
 4:     ruleSet = []
 5:     while len(D) > 0 do
 6:         member = D.pop()
 7:         while len(allAttrSets) > 0 do
 8:             attrSet = allAttrSets.pop()
 9:             newRule = CREATERULE(attrSet,member)
10:             rule.add(newRule)
11:             Sensitivity_notRM = CALCULATESENSITIVITY(rule,notRM)
12:             if Sensitivity_notRM <= (1/3) then
13:                 Sensitivity_RM = CALCULATESENSITIVITY(rule,RM)
14:                 Remove users in D, which are matching rule
15:                 if Sensitivity_RM >= (2/3) then
16:                     ruleSet.append(rule,PB_RM,PB_notRM)
17:                     if Sensitivity_RM == 1 AND PB_notRM == 0 then
18:                         allAttrSets = REDUCEALLATTRSET(allAttrSets, rule)
19:                     end if
20:                 else
21:                     if size(rule) <= max then
22:                         ... recursion ...
23:                     end if
24:                 end if
25:             end if
26:         end while
27:     end while
28:     return ruleSet
29: end procedure
```

B.4 Data Generator

Some papers are providing data generators for synthetic datasets for the performance evaluation of the various role mining algorithms. The data generator in Vaidya et al.[43] takes as input the number of users, permissions and roles to generate a pair of User-Role-Assignment and Role-Permission-Assignment matrices. Combining these two matrices, the corresponding User-Permission-Assignment matrix is obtained.
These data generators only consider users, permissions and their assignment to each other. They do not consider user attribute information. Due to this a new synthetic data generator has been created for this thesis, where the dimensions of users and permissions can be defined. The data generator is based on a reversed role-engineering process and generates users, roles, rules, user-role matrix, role-permission matrix, user-permission matrix and a user-permission matrix with noise.

- **Generation of users**
  Given a user count u and a list of user attributes $attr$ containing a list of user attribute values each, u users are generated. A user is described by a list of user attributes, which have a specific user attribute value assigned. For example:

  Given $attr$ = [$OrganizationalUnit$,$Location$,$EmployeeType$]

  where $OrganizationalUnit$ = [$HumanResources$,$Sales$,$Motor$],

  $Location$ = [$Denmark$, $Germany$, US] and

  $EmployeeType$ = [$Internal$,$External$]

  An user can be generated as follows: $user_1$ = [$Sales$,$Denmark$,$Internal$]. An additional parameter for a usertype count t determines how many different users can exist. This ensures that users with the same user attributes can occur several times.

- **Generation of roles**
  Given a permission count p and a role count r, r roles are generated, which are described by a set of permissions from the range [1;p]. For example given p=7 a role can be: $role_1 = \{1,5,7\}$. A parameter max_p limits how often a permission can occur in all r roles. Additionally a configuration list $conf_p$ can regulate the density of roles. A configuration list entry contains 1) a percentage of all

roles, which are affected by this configuration entry 2) the minimum count of permissions the role must have and 3) the maximum count of permissions the role must have. For example the configuration list [(0.2,10,20),(0.8,1,5)] would describe the configuration that 20% of the r roles must have between 10 and 20 permissions and the other 80% of the r roles must have between 1 and 5 permissions.

- **Generation of rules**
  Given a role count r and a list of user attributes $attr$ containing a list of user attribute values each, r rules are generated (a rule for each role). A rule is a list of conditions which are OR-connected. A condition is a list of user attributes, which have no, one or several user attribute values assigned. For example given the user attributes $attr$ from above, $rule_1$ = [[{$HumanResources$},{$Denmark$}, {}], [{$HumanResources$},{$Germany$},{}]] would describe the rule $HumanResources \wedge (Denmark \vee Germany)$. A parameter for the maximal condition count $cond_{max}$ configures how many OR-connected conditions can be in a rule.
  This rule is assigned to a role. Rules are generated until:
  - each role has a rule assigned
  - each generated user is meeting one rule
  - each rule is met by at least one user

Since users and roles have been generated before, the rules can be applied to create the following outputs:

- **User-role matrix**
  The User-role matrix is a boolean matrix, where the rows are users and the columns are roles. A ”1” in the matrix would mean that the user (row) is assigned to a the role (column). The users get those roles assigned, which have a rule assigned the user is fullfilling. For example $user_1$ would not fulfill $rule_1$, since the first condition for OrganizationalUnit is not met. The user $user_2$ = [$HumanResources$,$Denmark$,$Internal$] on the other hand would met the conditions in $rule_1$ and would therefore get get according role assigned.

- **Role-permission matrix**
  The Role-permission matrix is a boolean matrix, where the rows are roles and

the columns are permissions. A ”1” in the matrix would stand for that the role (row) is containing the permission (column). The matrix is generated from the generated roles.

- **User-permission matrix**
  The User-permission matrix is a boolean matrix, where the rows are users and the columns are permissions. A ”1” in the matrix would imply that the user (row) is has the permission (column). The matrix is generated from the matrix multiplication of the User-role matrix and the Role-permission matrix.

- **User-permission matrix with noise**
  Given two parameters for noise, random bits in the User-permission matrix are flipped. The first parameter is the probability that a user-permission-relation is removed (”1” flipped to ”0”) and the second parameter is the probability that a user-permission-relation is added (”0” flipped to ”1”).

Appendix C

Datasets

C.1 Sythentic Datasets

C.1.1 Dataset1

Table C.1: Users with attributes

User	Attribute1	Attribute2	Attribute3
1	HumanResources	Denmark	Internal
2	Motor	Denmark	External
3	Motor	Denmark	External
4	Motor	Denmark	External
5	Motor	Germany	Internal
6	Motor	Germany	Internal
7	Sales	Denmark	Internal
8	Sales	Denmark	Internal
9	Sales	Denmark	Internal
10	Sales	US	External

Table C.2: Roles and their containing permissions

Role	Permissions
1	8,1,2,3,6
2	9,5,6
3	0,4
4	3,7

Table C.3: Rules for roles

Role	Attribute1	Attribute2	Attribute3
1	Motor		
2		US	
3		Denmark	External
4		Denmark	Internal

C.1.2 Dataset 2

Table C.4: Users with attributes

User	Attribute1	Attribute2	Attribute3
1-4	Development	Denmark	External
5	Development	Germany	External
6	Development	US	External
7	Development	US	Internal
8-9	FacilityManagement	Denmark	Internal
10-11	FacilityManagement	Germany	External
12-13	FacilityManagement	Germany	Internal
14-15	FacilityManagement	US	External
16	FacilityManagement	US	Internal
17	HumanResources	Denmark	External
18	HumanResources	Germany	External
19	HumanResources	Germany	Internal
20	HumanResources	US	External
21	HumanResources	US	Internal
22-26	Motor	Denmark	External
27-28	Motor	Denmark	Internal
29-32	Motor	Germany	Internal
33-34	Motor	US	External
35-36	Motor	US	Internal
37-38	Sales	Denmark	External
39-43	Sales	Denmark	Internal
44	Sales	Germany	External
45	Sales	Germany	Internal
46-49	Sales	US	External
50	Sales	US	Internal

Table C.5: Roles and their containing permissions

Role	Permissions
1	44,13,46,19,22,24,25,26,27
2	37,6,41,45,15,48,49,19
3	32,35,36,10,44,12,47,24
4	32,25,18,3,15
5	20,13
6	11,29,23
7	32,34,10,4,6
8	0,3,14,31
9	1,2,34,5,7,20
10	37,40,43,13,45,15,17
11	8,43,11,17,25,28
12	35,39,43,45,16,20
13	34,35,42,12,23,30
14	21,38
15	38,9,17,30,39

Table C.6: Rules for roles

Role	Attribute1	Attribute2	Attribute3
1	Sales		
2	Development	US	External
3	Development	Germany	External
4	FacilityManagement		
5		US	
6	HumanResources		External
7			Internal
8	HumanResources	Germany	External
9			Internal
10	Motor	Germany	Internal
11	HumanResources	US	Internal
12	HumanResources	Germany	
13		Denmark	
14	Motor	US	
15	HumanResources	US	External

C.2 Real Datasets

C.2.1 Domino

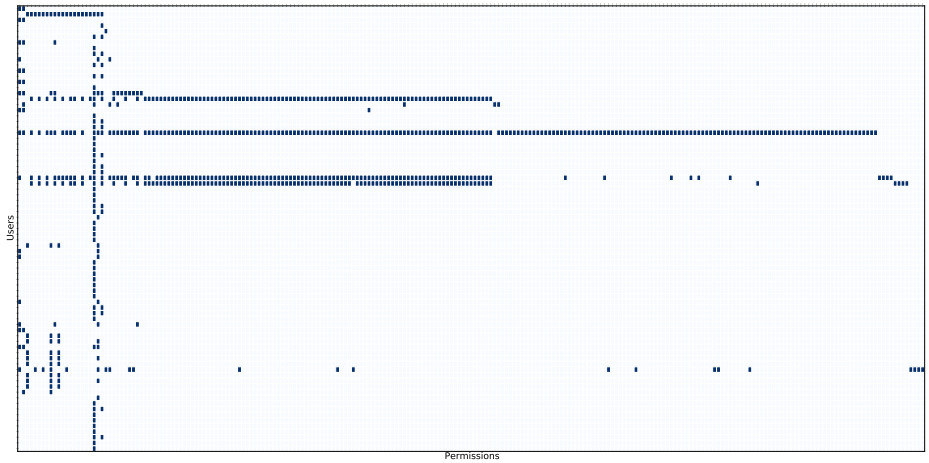


Figure C.1: DOMINO

Appendix D

Experiment Results

D.1 Experiment 1

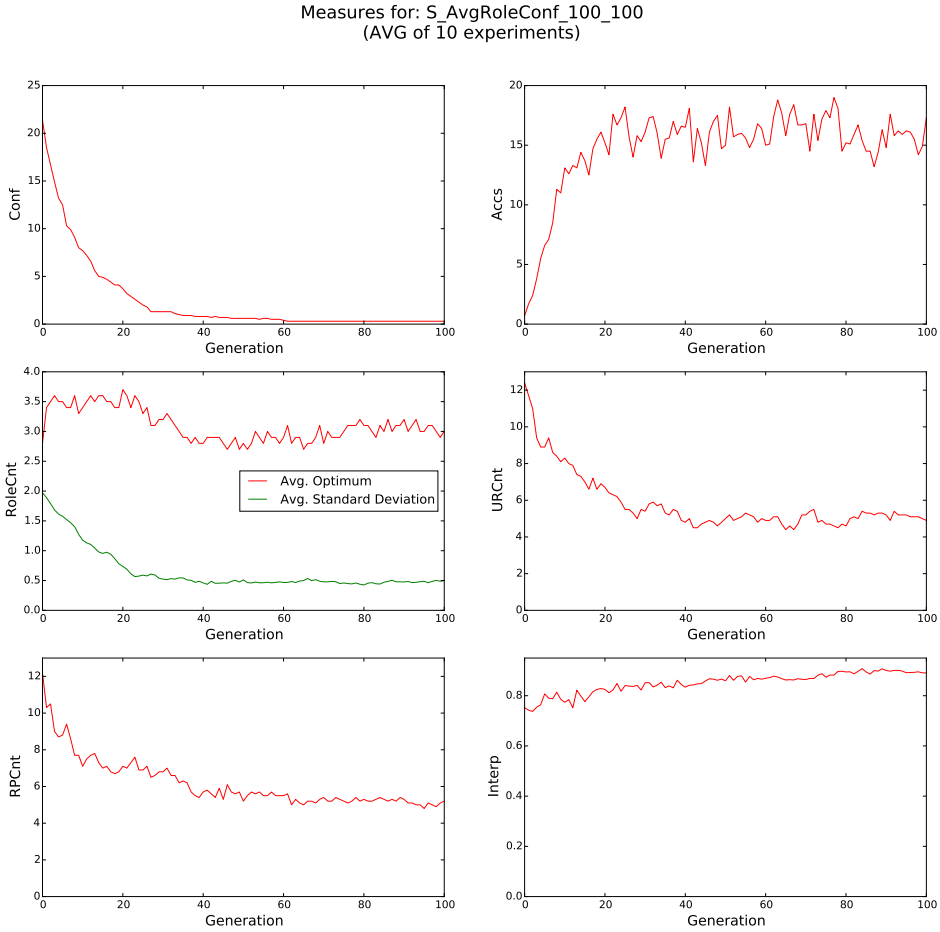


Figure D.1: EXPERIMENT 1c: Results of EvoRoleMiner with Fitness function $F = avg(G_{conf})$ on synthetic dataset 1 with setup in table **??**. *From u.l. to l.r.: Confidentiality Violations, Availability Violations, Role Count, User-Role Assignments, Role-Permission Assignments, Interpretability.*

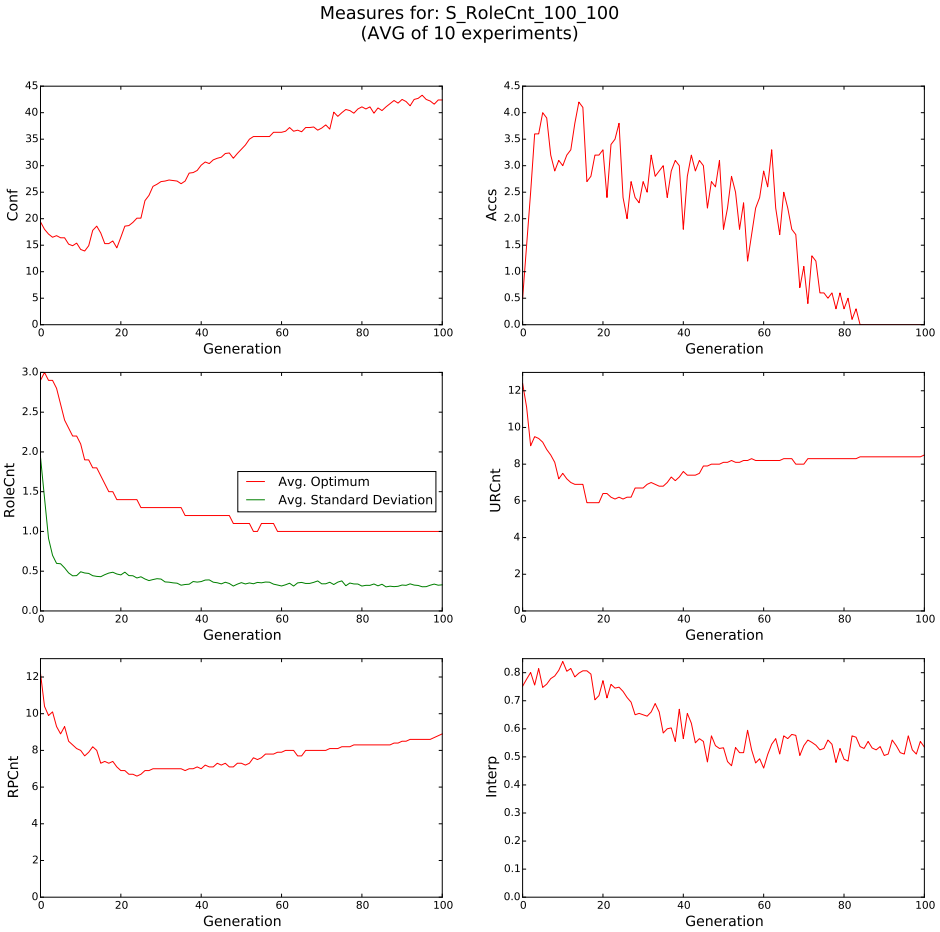


Figure D.2: EXPERIMENT 1d: Results of EvoRoleMiner with Fitness function $F = |R|$ on synthetic dataset 1 with setup in table **??**. *From u.l. to l.r.: Confidentiality Violations, Availability Violations, Role Count, User-Role Assignments, Role-Permission Assignments, Interpretability.*

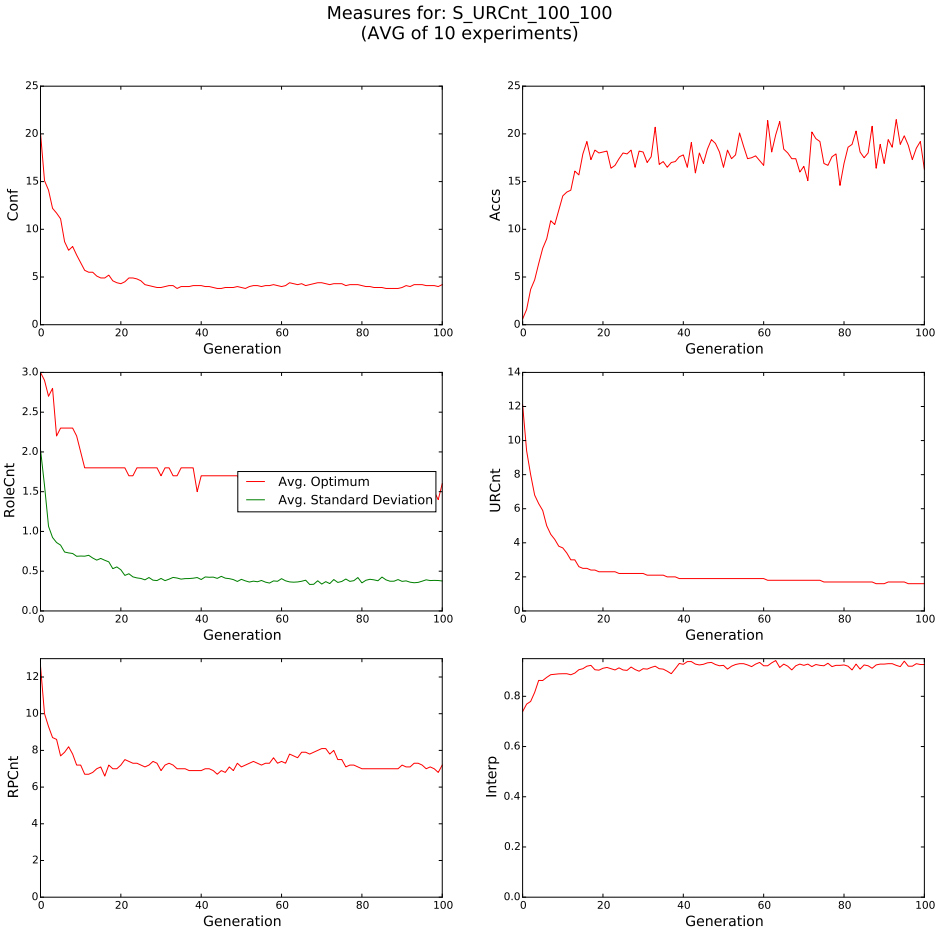


Figure D.3: EXPERIMENT 1e: Results of EvoRoleMiner with Fitness function $F = |UA|$ on synthetic dataset 1 with setup in table 6.3. From u.l. to l.r.: Confidentiality Violations, Availability Violations, Role Count, User-Role Assignments, Role-Permission Assignments, Interpretability.

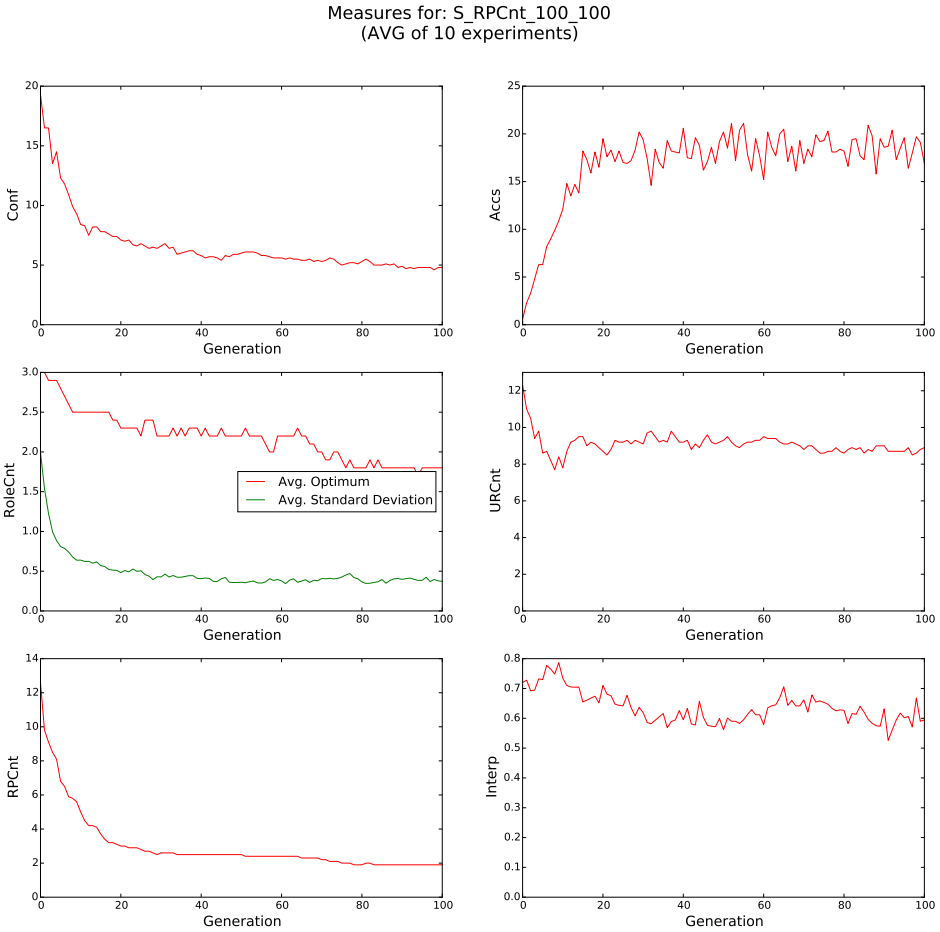


Figure D.4: EXPERIMENT 1f: Results of EvoRoleMiner with Fitness function $F = |PA|$ on synthetic dataset 1 with setup in table 6.3. From u.l. to l.r.: Confidentiality Violations, Availability Violations, Role Count, User-Role Assignments, Role-Permission Assignments, Interpretability.

D.2 Experiment 2

Table D.1: Evo-RoleMiner: Results of ten experiments for Dataset1 and Healthcare dataset. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last Generation of all experiments. The values for Interpretability (INT) are the average maximum in the last Generation of all experiments. The time is the average runtime in seconds of all experiments.

DATASET1

Experiment	**Fitness Function**	**Fitness**	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|PA\|$	**INT**	**Time (in sec)**
2a	F_{basic}^{min}	0.38	2.8	0	2	8.9	7.9	1	336
2b	F_{edge}^{min}	0.15	2.8	0	3.1	9.6	11.1	1	343

HEALTHCARE DATASET

Experiment	**Fitness Function**	**Fitness**	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|PA\|$	**INT**	**Time (in sec)**
2c	F_{basic}^{min}	0.13	3.6	48.4	2	54.8	60.8	-	664
2d	F_{edge}^{min}	0.12	4.5	52.5	3.3	62.3	56.6	-	775

D.2.1 Experiment 2a

Result Data

Table D.2: Results of Experiment 2a: Dataset1, F^{Min}_{Basic}, Setup 2

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min
1	424	0.377769019	1.149704375	0.426066102	0.112333904	4	40	5.842	4.785084743	0
2	443	0.377769019	1.221836815	0.433073354	0.120951333	3	42	6.068	5.067087526	0
3	449	0.377769019	1.204887663	0.436002483	0.129238799	3	44	6.226	5.437915409	0
4	423	0.377769019	0.962987781	0.428166732	0.110813532	2	34	5.836	4.625916558	0
5	404	0.377769019	1.18793851	0.432763618	0.123186393	2	40	6.109	5.244722967	0
6	458	0.377769019	1.141426882	0.428852621	0.115522095	3	40	5.926	4.787747278	0
7	434	0.377769019	1.135120221	0.427447024	0.109024379	3	41	5.758	4.503047413	0
8	482	0.377769019	1.194245171	0.428993023	0.121726637	3	45	5.962	5.30721735	0
9	423	0.377769019	1.152069373	0.422152266	0.107280322	3	42	5.686	4.499489304	0
10	435	0.377769019	1.117776902	0.428012929	0.114866488	2	35	5.898	4.766927312	0
AVG	437.5	0.377769019	1.146799369	0.429153015	0.116494388	2.8	40.3	5.9311	4.902515586	0

Experiment	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	10	3.214	1.101001362	2	4	2.121	0.363811765	9	24	9.6
2	11	3.256	1.185944349	2	4	2.143	0.382819801	9	22	9.656
3	12	3.241	1.189503678	2	4	2.149	0.410851555	9	23	9.741
4	12	3.313	1.296545796	2	4	2.12	0.345832329	9	18	9.599
5	10	3.23	1.170939794	2	4	2.139	0.379050129	9	21	9.665
6	12	3.221	1.11631492	2	4	2.133	0.388987146	9	21	9.663
7	13	3.326	1.341537923	2	4	2.123	0.365883861	9	24	9.61
8	14	3.218	1.153462613	2	4	2.129	0.366550133	8	21	9.655
9	13	3.211	1.086498504	2	4	2.109	0.348021551	9	22	9.544
10	13	3.201	1.163872416	2	4	2.134	0.382157036	9	20	9.649
AVG	12	3.2431	1.180562135	2	4	2.13	0.373396531	8.9	21.6	9.6382

Experiment	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime
1	1.722788437	8	22	9.571	1.872153573	0.45	1	0.953491667	0.123598094	406.153624
2	1.695188485	8	21	9.66	1.901157542	0.333333333	1	0.9512	0.123606472	413.456295
3	1.976339799	8	23	9.68	2.021781393	0.3	1	0.951525	0.124034468	386.242228
4	1.582466113	8	21	9.511	1.701728239	0.25	1	0.951675	0.127128161	0
5	1.791863555	8	22	9.624	1.833745893	0	1	0.951616667	0.127101874	254.715648
6	1.799841938	8	23	9.576	1.824342073	0.333333333	1	0.95085	0.126742739	309.848566
7	1.716362433	8	21	9.518	1.771911309	0.25	1	0.955308333	0.118857767	303.092263
8	1.827559849	8	23	9.565	1.809910219	0.3	1	0.951008333	0.130667187	315.908143
9	1.674533965	8	22	9.481	1.648526312	0.25	1	0.958041667	0.122806243	325.160113
10	1.748656341	7	23	9.583	1.830057649	0.333333333	1	0.949616667	0.12924242	312.639989
AVG	1.753560091	7.9	22.1	9.5769	1.821531598	0.28	1	0.952433333	0.125378543	336.3574299

D.2.2 Experiment 2b

Result Data

Table D.3: Results of Experiment 2b: Dataset1, F_{Edge}^{Min}, Setup 2

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min
1	448	0.151957621	0.950218091	0.197838357	0.112719345	3	40	5.653	4.484929319	0
2	468	0.151957621	0.910013124	0.196031551	0.106774032	3	39	5.597	4.268558422	0
3	414	0.151957621	0.954900441	0.193278273	0.108811647	3	41	5.53	4.379394935	0
4	446	0.151957621	0.904052983	0.195771496	0.106907863	4	38	5.561	4.295844387	0
5	418	0.151957621	0.828515985	0.193427922	0.109222984	2	34	5.479	4.28830491	0
6	430	0.151957621	0.931835212	0.198015858	0.114541813	4	39	5.692	4.632832395	0
7	444	0.151957621	0.859165667	0.193724664	0.107789762	3	36	5.495	4.325734042	0
8	450	0.151957621	0.957217792	0.192751294	0.102121551	2	35	5.418	3.887322472	0
9	434	0.151957621	0.893063972	0.196081224	0.10881912	4	38	5.564	4.240271689	0
10	434	0.139106065	0.863263264	0.177983839	0.10480901	0	33	1.44	4.305159695	0
AVG	438.6	0.150672465	0.905224653	0.193490448	0.108251713	2.8	37.3	5.1429	4.310835227	0

Experiment	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	8	0.246	0.971331046	3	5	3.123	0.334471224	9	22	10.583
2	9	0.28	1.051475154	3	4	3.098	0.297314648	10	19	10.531
3	10	0.218	0.976972876	3	4	3.094	0.291828717	10	19	10.508
4	8	0.296	1.090130267	3	4	3.102	0.302648311	9	20	10.488
5	9	0.272	1.046907828	3	5	3.095	0.296605799	9	19	10.506
6	9	0.246	0.956809281	3	5	3.111	0.317299543	10	19	10.545
7	9	0.292	1.090291704	3	4	3.09	0.28618176	9	19	10.513
8	9	0.242	0.953643539	3	4	3.113	0.316592798	10	18	10.538
9	10	0.275	1.111474246	3	5	3.111	0.320435641	10	23	10.57
10	10	0.231	0.838831926	4	5	4.087	0.281835058	10	19	10.492
AVG	9.1	0.2598	1.008786787	3.1	4.5	3.2024	0.30452135	9.6	19.7	10.5274

Experiment	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime_Sum
1	1.574519292	11	21	12.522	1.583513814	0.5	1	0.967731667	0.087395377	347.57399
2	1.484263791	11	21	12.424	1.465682094	0.5	1	0.965483333	0.090418716	380.48422
3	1.477137773	11	21	12.431	1.4564739	0.25	1	0.969391667	0.084816006	355.304555
4	1.404227902	10	21	12.465	1.555241139	0.5	1	0.968441667	0.085626624	328.616984
5	1.499988	10	20	12.411	1.467678098	0.5	1	0.970053333	0.085864961	336.439346
6	1.477827798	11	22	12.516	1.680994943	0.45	1	0.969331667	0.084649838	321.674719
7	1.527688123	11	21	12.364	1.359229193	0.333333333	1	0.968058333	0.090619439	318.312672
8	1.45827158	11	24	12.475	1.556076798	0.45	1	0.971433333	0.079903073	326.259763
9	1.644718821	11	20	12.451	1.457257355	0.36	1	0.964068333	0.096519867	331.524604
10	1.398547818	14	25	17.336	1.323292863	0.6	1	0.974785	0.068962517	387.358279
AVG	1.49471909	11.1	21.6	12.9395	1.490541369	0.444333333	1	0.968877833	0.085477642	343.3549132

D.2.3 Experiment 2c

Result Data

Table D.4: Results of Experiment 2c: Healthcare Dataset, F^{Min}_{Basic}, Setup 2

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min
1	438	0.130323897	0.615999032	0.151347383	0.062477382	0	301	16.692	37.71603288	63
2	458	0.130323897	0.73341388	0.157447219	0.077672584	5	384	20.391	47.67792067	53
3	430	0.130323897	0.674437884	0.15165764	0.061900488	0	354	16.864	37.7087192	36
4	432	0.130323897	0.67147639	0.15386933	0.066658348	0	365	18.377	40.80447121	36
5	419	0.130323897	0.635496057	0.154462113	0.068817039	5	315	18.572	41.44727755	52
6	429	0.130323897	0.606344306	0.155570018	0.071631728	1	300	19.096	43.8827846	58
7	394	0.130323897	0.679098455	0.151950346	0.067782347	0	351	17.278	41.49003153	41
8	446	0.130323897	0.705846916	0.15379811	0.068870807	5	370	18.383	42.26594742	45
9	429	0.154335484	0.695665704	0.177067085	0.066169436	15	364	33.61	39.63107241	44
10	427	0.130323897	0.642329969	0.154207422	0.06740591	5	336	18.355	41.13041423	56
AVG	430.2	0.132725056	0.666010859	0.156137667	0.067938607	3.6	344	19.7618	41.37546717	48.4

Experiment	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	166	113.959	9.677981143	2	4	2.116	0.358530333	45	104	55.021
2	166	113.472	10.66954619	2	5	2.142	0.392219326	61	126	65.536
3	166	114.112	10.06804132	2	4	2.113	0.35246183	54	110	61.993
4	166	113.609	10.3082549	2	4	2.12	0.36	60	111	64.862
5	166	113.74	10.23964843	2	4	2.129	0.392885479	60	122	65.298
6	194	113.894	11.18028461	2	4	2.137	0.382401621	51	115	57.558
7	142	113.41	9.154993173	2	4	2.118	0.352244233	50	101	58.511
8	166	113.295	9.809280045	2	4	2.126	0.363488652	61	121	65.013
9	164	112.174	9.37879118	2	4	2.121	0.361052628	44	104	50.491
10	163	113.808	9.832148087	2	4	2.131	0.37126675	62	119	65.25
AVG	165.9	113.5473	10.03189691	2	4.1	2.1253	0.368655321	54.8	113.3	60.9533

Experiment	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime
1	7.009319439	65	120	67.974	6.876577928	0	0	0	0	673.170894
2	7.75646208	48	105	58.294	8.528866513	0	0	0	0	738.401978
3	6.618833054	65	118	67.965	6.670215514	0	0	0	0	651.093222
4	6.791830092	64	122	67.961	7.148389959	0	0	0	0	671.645715
5	7.580844016	54	118	62.096	8.028498241	0	0	0	0	646.72621
6	7.605040171	64	121	68.416	7.439014989	0	0	0	0	659.20267
7	7.017825803	65	120	68.074	6.856422099	0	0	0	0	635.65854
8	7.236078427	63	125	67.672	6.86239142	0	0	0	0	651.406479
9	7.509721633	66	121	69.12	6.836929135	0	0	0	0	661.926049
10	7.231839323	54	115	64.092	7.749679735	0	0	0	0	651.49447
AVG	7.235779404	60.8	118.5	66.1664	7.299698553	0	0	0	0	664.0726227

D.2.4 Experiment 2d

Result Data

Table D.5: Results of Experiment 2d: Healthcare Dataset, F^{Min}_{Edge}, Setup 2

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min
1	440	0.122094308	0.652777475	0.148445876	0.073942818	5	313	14.563	36.09113508	56
2	441	0.123248421	0.610611923	0.149685959	0.06763063	16	266	23.673	28.61307518	67
3	459	0.127405752	0.613321023	0.148798701	0.061634077	0	282	12.531	29.8048157	58
4	452	0.093314057	0.637202857	0.116263647	0.065041304	5	297	13.28	32.23950372	40
5	449	0.120162874	0.525440678	0.140486656	0.057612971	5	271	13.19	31.11407881	46
6	444	0.110944496	0.627823762	0.126268738	0.050248204	10	319	18.644	28.01548258	35
7	455	0.140641126	0.657918404	0.162385924	0.067111664	4	327	16.918	31.17180258	37
8	433	0.135131489	0.602258371	0.151203818	0.052246174	0	305	10.006	23.94201253	60
9	433	0.12686091	0.497834593	0.14489562	0.053041654	0	239	8.977	29.78826062	64
10	453	0.124793266	0.536038771	0.141382439	0.051006272	0	247	6.705	27.88777465	62
AVG	445.9	0.12245967	0.596122786	0.142981738	0.059951577	4.5	286.6	13.8487	29.86679414	52.5

Experiment	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	542	130.46	64.14480805	3	6	4.016	0.334281319	63	127	93.667
2	473	122.756	76.4776599	3	6	4.018	0.357317786	63	117	80.422
3	417	127.939	49.77868298	4	6	4.991	0.344846343	69	126	94.775
4	361	101.391	53.52397705	2	6	3.019	0.344440125	47	107	66.252
5	337	124.957	39.13327422	3	5	3.992	0.343418113	48	103	71.756
6	270	74.845	27.51350532	4	6	5.007	0.314564779	69	124	91.205
7	628	129.756	67.40009246	4	6	5.004	0.322465502	75	145	114.852
8	520	126.496	54.14161047	4	6	4.99	0.303150128	55	111	79.069
9	294	121.788	28.74239127	3	6	5.003	0.324023147	67	144	113
10	402	122.901	33.35936449	3	6	4.989	0.32996818	67	143	110.832
AVG	424.4	118.3289	49.42153662	3.3	5.9	4.5029	0.331847542	62.3	124.7	91.583

Experiment	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime
1	7.462714721	50	118	69.596	6.207316973	0	0	0	0	750.174128
2	6.558804464	45	97	66.239	7.180381536	0	0	0	0	718.898073
3	6.503258798	63	113	79.399	5.554079492	0	0	0	0	820.277028
4	6.36007044	52	117	75.058	7.190732647	0	0	0	0	692.369822
5	6.842694206	70	120	87.194	5.871657688	0	0	0	0	732.606132
6	6.038292391	72	120	90.192	5.105402629	0	0	0	0	796.778843
7	6.599552712	37	104	70.532	5.899235205	0	0	0	0	826.102517
8	5.32524544	77	144	110.383	6.907409862	0	0	0	0	796.886848
9	6.6938778	51	105	73.322	5.59234441	0	0	0	0	819.119086
10	6.078632741	49	106	74.121	6.022819855	0	0	0	0	798.321932
AVG	6.446314371	56.6	114.4	79.6036	6.153138029	0	0	0	0	775.1534409

D.3 Experiment 3

Table D.6: Evo-RoleMiner: Results of ten experiments for Dataset1 and Healthcare dataset. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last Generation of all experiments. The values for Interpretability (INT) are the average maximum in the last Generation of all experiments. The time is the average runtime in seconds of all experiments.

DATASET1

Experiment	Fitness Function	Fitness	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|PA\|$	INT	Time (in sec)
3a	F^{min}_{basic}	0.08	0	0	3.9	10	13.3	1	372
3b	F^{min}_{edge}	0.05	0.2	0	3.9	11.4	10.8	0.998	371

HEALTHCARE DATASET

Experiment	Fitness Function	Fitness	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|PA\|$	INT	Time (in sec)
3c	F^{min}_{basic}	0.09	1.3	43.6	8.5	153.5	154.4	-	1329
3d	F^{min}_{edge}	0.05	2.1	18.1	14.8	155	168.6	-	1554

D.3.1 Experiment 3a

Result Data

Table D.7: Results of Experiment 3a: Dataset1, F^{Min}_{Basic}, Setup 3

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std
1	418	0.080204966	0.751221916	0.120818881	0.096566214	0	39	1.999	5.112240116
2	431	0.080204966	0.683031139	0.114806342	0.086074097	0	28	1.678	4.488910335
3	424	0.080204966	0.717323611	0.120406977	0.095696585	0	37	1.891	5.051051277
4	484	0.080204966	0.742550256	0.122039365	0.098460377	0	33	2.032	5.186036637
5	466	0.080204966	0.734272763	0.123115278	0.097921781	0	38	2.079	5.258398901
6	452	0.080204966	0.77447773	0.12583406	0.096859097	0	39	2.199	5.155133267
7	458	0.080204966	0.768171068	0.1200862	0.097547104	0	40	1.936	5.218228052
8	434	0.080204966	0.683425305	0.12142095	0.096204664	0	35	1.924	5.085294878
9	455	0.080204966	0.82793851	0.126556334	0.108014221	0	40	2.184	5.706324912
10	475	0.080204966	0.757528577	0.127114253	0.099805549	0	38	2.192	5.268314341
AVG	449.7	0.080204966	0.743994088	0.122219864	0.097314969	0	36.7	2.0114	5.152993272

Experiment	Aces_Min	Aces_Max	Aces_Avg	Aces_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max
1	0	9	0.225	0.899096769	4	6	4.15	0.365376518	10	21
2	0	11	0.206	0.804713614	4	6	4.137	0.352464183	10	22
3	0	7	0.286	0.946680516	4	6	4.15	0.362629287	9	23
4	0	8	0.253	0.898326778	4	6	4.151	0.366331817	10	20
5	0	7	0.262	0.856362073	4	6	4.158	0.370183738	9	21
6	0	11	0.285	0.979681071	4	6	4.173	0.386097138	9	22
7	0	8	0.245	0.920312447	3	6	4.137	0.360875325	9	21
8	0	16	0.312	1.13607042	4	7	4.135	0.356054771	11	28
9	0	12	0.33	1.031067408	4	6	4.166	0.38005789	11	23
10	0	9	0.343	1.11146345	4	6	4.178	0.390276825	12	22
AVG	0	9.8	0.2747	0.958377455	3.9	6.1	4.1535	0.369034749	10	22.3

Experiment	URCnt_Avg	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std
1	11.427	1.975517907	15	29	17.668	1.797157756	0.58	1	0.960866667	0.07235882
2	12.731	2.003157258	12	26	17.471	1.550857505	0.58	1	0.964871667	0.074832908
3	10.879	1.758510449	15	27	17.6	1.740689519	0.58	1	0.965778333	0.076289888
4	11.17	1.940386559	15	26	17.637	1.719660141	0.58	1	0.959558333	0.075461741
5	11.051	1.854831259	16	31	17.684	1.862295358	0.58	1	0.962136667	0.075361877
6	11.563	2.057190074	15	27	17.726	1.825629754	0.58	1	0.955131667	0.074497963
7	10.927	1.78764398	12	27	17.595	1.74555865	0.58	1	0.966021667	0.07165196
8	13.576	1.796725911	11	26	15.736	2.654864215	0.475	1	0.969466429	0.07717343
9	13.705	1.853098756	11	26	14.498	2.109027264	0.58	1	0.969443333	0.073332205
10	13.743	1.84416675	11	25	14.17	2.283221408	0.6	1	0.966083333	0.078820885
AVG	12.0772	1.88712289	13.3	27	16.7785	1.928896157	0.5715	1	0.96393581	0.074978168

D.3.2 Experiment 3b

Result Data

Table D.8: Results of Experiment 3b: Dataset1, F_{Edge}^{Min}, Setup 3

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std
1	458	0.083993381	0.732289587	0.122651478	0.094165354	2	38	3.894	4.933230585
2	467	0.047897274	0.736816435	0.099473542	0.107569635	0	32	2.339	5.567232616
3	421	0.047897274	0.785848562	0.088373656	0.098144757	0	41	1.742	5.113651924
4	410	0.047897274	0.703300602	0.089456206	0.104384665	0	36	1.879	5.397440041
5	451	0.047897274	0.737784092	0.089120937	0.095646145	0	33	1.756	4.819591684
6	435	0.047897274	0.639553076	0.093023199	0.101357985	0	34	2.067	5.291361923
7	453	0.047897274	0.667920924	0.088545634	0.094607305	0	32	1.785	4.836401038
8	450	0.047897274	0.778443	0.094319312	0.104969409	0	42	2.053	5.464813904
9	423	0.047897274	0.645859737	0.091738335	0.100962444	0	34	1.991	5.263166252
10	445	0.047897274	0.588705618	0.089753182	0.093913997	0	31	1.832	4.715694647
AVG	441.3	0.051506885	0.701652163	0.094645548	0.099572169	0.2	35.3	2.1338	5.140258461

Experiment	Acces_Min	Acces_Max	Acces_Avg	Acces_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max
1	0	9	0.217	0.855517972	4	6	4.15	0.359861084	10	20
2	0	15	0.438	1.424133421	4	6	4.178	0.392830752	12	25
3	0	9	0.42	1.258411697	3	6	4.121	0.338170076	11	22
4	0	9	0.359	1.275977664	4	6	4.137	0.360875325	11	23
5	0	9	0.439	1.30854079	4	6	4.131	0.343276856	12	23
6	0	9	0.367	1.198461931	4	6	4.162	0.386983204	11	26
7	0	15	0.391	1.352818909	4	7	4.138	0.361878433	12	21
8	0	11	0.437	1.376237988	4	6	4.152	0.364549036	12	26
9	0	10	0.373	1.269594817	4	6	4.14	0.349857114	11	26
10	0	10	0.403	1.248435421	4	6	4.154	0.382470914	12	22
AVG	0	10.6	0.3844	1.256813061	3.9	6.1	4.1463	0.364075279	11.4	23.4

Experiment	URCnt_Avg	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std
1	10.758	1.800398845	16	26	17.616	1.672884933	0.475	0.975	0.918085	0.077128191
2	13.816	1.99352552	10	25	12.773	1.996364446	0.5	1	0.963571667	0.083548299
3	13.511	1.646171012	10	21	12.566	1.691639441	0.4	1	0.973356667	0.071602254
4	13.591	1.766272629	10	26	12.649	1.847105574	0.5	1	0.976213333	0.064058654
5	13.561	1.642035018	10	21	12.578	1.684017815	0.483333333	1	0.97152	0.073107308
6	13.742	1.938926507	11	22	12.675	1.795932905	0.58	1	0.965418333	0.080052864
7	13.599	1.709444062	10	25	12.585	1.720109008	0.6	1	0.971635	0.073770979
8	13.647	1.730430871	10	25	12.684	1.825963855	0.58	1	0.969061667	0.074534258
9	13.626	1.765821055	10	23	12.667	1.812211632	0.58	1	0.972381667	0.071748267
10	13.645	1.770021186	11	22	12.659	1.796863656	0.475	1	0.970683333	0.076525702
AVG	13.3496	1.776304671	10.8	23.6	13.1452	1.784309327	0.517333333	0.9975	0.965192667	0.074607678

D.3.3 Experiment 3c

Result Data

Table D.9: Results of Experiment 3c: Healthcare, F^{Min}_{Basic}, Setup 3

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min
1	454	0.09631301	0.521953054	0.117408675	0.06024663	0	288	13.241	39.79184488	62
2	427	0.087570576	0.537670653	0.108458851	0.060542374	0	294	13.26	40.07303832	54
3	412	0.074913125	0.469259001	0.096010416	0.061664844	1	257	14.97	39.79873239	39
4	442	0.09972249	0.555489216	0.119168047	0.060485206	5	304	18.344	39.71890814	49
5	437	0.08634554	0.660621889	0.107149975	0.060066003	0	377	14.969	39.45589993	43
6	441	0.081205265	0.545458882	0.107452036	0.072578819	0	306	16.029	46.94326532	45
7	452	0.097345255	0.52778655	0.115143962	0.052697535	0	297	13.94	34.75325021	46
8	443	0.093993088	0.516849365	0.113037376	0.058918085	5	287	17.177	38.85680984	41
9	439	0.086225697	0.582029026	0.108630059	0.065806248	2	334	16.891	42.88481222	33
10	456	0.064069838	0.455089714	0.090540953	0.071123279	0	255	17.554	45.60327931	24
AVG	440.3	0.086770388	0.537220735	0.108300035	0.062412902	1.3	299.9	15.6375	40.78798406	43.6

Experiment	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	217	113.807	12.32541078	7	10	8.111	0.377728739	132	218	167.867
2	302	100.47	12.49236167	7	10	8.106	0.369816171	117	206	159.823
3	154	68.251	8.195852549	10	13	11.1	0.4	152	233	187.837
4	170	91.493	7.659500702	11	14	12.103	0.372009408	202	295	241.322
5	188	88.002	10.91540178	9	12	10.115	0.376530211	159	228	195.766
6	360	98.861	21.38035732	5	8	6.122	0.38094094	92	188	141.144
7	197	102.356	11.87178436	9	12	10.102	0.373625481	180	252	221.052
8	486	101.264	14.50711219	6	9	7.111	0.364251287	156	238	194.118
9	156	79.176	9.741818311	11	14	12.125	0.399217986	195	266	232.619
10	109	53.938	6.472415005	10	13	11.128	0.386802275	150	232	188.192
AVG	233.9	89.7618	11.55620147	8.5	11.5	9.6123	0.38009225	153.5	235.6	192.974

Experiment	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime
1	7.722649222	133	197	167.516	7.866113653	0	0	0	0	1222.111381
2	8.051314862	152	235	191.823	8.122171569	0	0	0	0	1160.901816
3	8.159193036	192	259	225.137	7.950863538	0	0	0	0	1377.85762
4	7.840555848	200	278	237.224	7.511446199	0	0	0	0	1563.08834
5	8.034876726	166	236	202.892	7.784878676	0	0	0	0	1377.497199
6	8.470021488	97	194	142.946	7.935432187	0	0	0	0	1308.503251
7	7.554819389	135	205	172.372	7.385094177	0	0	0	0	1318.397817
8	8.522445424	81	143	109.728	6.736469105	0	0	0	0	1071.383932
9	8.303965258	182	268	217.77	7.72263556	0	0	0	0	1450.218401
10	8.194945759	206	290	245.185	8.450016272	0	0	0	0	1436.321498
AVG	8.085478701	154.4	230.5	191.2593	7.746512094	0	0	0	0	1328.628126

D.3.4 Experiment 3d

Result Data

Table D.10: Results of Experiment 3d: Healthcare, F_{Edge}^{INT}, Setup 3

Experiment	evals	Fitness_Min	Fitness_Max	Fitness_Avg	Fitness_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min
1	391	0.04830975	0.5351468	0.070338926	0.06551064	2	317	18.813	40.38286804	15
2	440	0.046074091	0.496643925	0.065260755	0.057031959	1	292	12.784	35.58389164	8
3	480	0.031574058	0.531037603	0.058975111	0.072997781	1	320	17.252	45.36153101	9
4	436	0.061840319	0.508138653	0.084854268	0.06290241	3	292	18.129	39.68262036	29
5	420	0.046519185	0.453583914	0.071023532	0.06780553	2	262	16.614	42.8514761	18
6	441	0.051339123	0.545951394	0.075824402	0.068084464	2	292	16.424	43.19113594	20
7	459	0.039501013	0.546070896	0.060126611	0.058808818	0	324	13.07	37.03378323	19
8	417	0.046421462	0.59435116	0.072799077	0.06595082	5	355	20.58	41.3848958	15
9	452	0.050368266	0.536045493	0.074125999	0.068784288	3	307	17.937	43.11694598	17
10	445	0.060800464	0.430020205	0.079157181	0.053618115	2	240	13.088	34.08577791	31
AVG	438.1	0.048274773	0.517699004	0.071248586	0.064149483	2.1	300.1	16.4691	40.2674926	18.1

Experiment	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	219	32.123	16.14242457	17	20	18.102	0.376292439	201	290	236.384
2	327	47.373	15.89074797	11	15	13.095	0.382066748	137	210	167.539
3	305	24.828	14.59412265	14	17	15.15	0.399374511	124	197	156.707
4	282	60.07	17.03071105	13	16	14.145	0.431248188	146	226	177.648
5	244	42.444	13.93818008	16	19	17.12	0.384187454	156	218	187.01
6	270	45.611	10.0998851	16	19	17.147	0.411571379	163	231	191.932
7	239	36.302	13.38629135	14	17	15.118	0.363422619	136	211	160.871
8	227	35.976	12.35457098	14	17	15.122	0.375654096	164	251	195.884
9	176	41.962	14.06074522	15	18	16.126	0.390030768	155	237	186.578
10	248	58.63	10.66644739	18	21	19.129	0.369268195	168	245	203.88
AVG	253.7	42.5319	13.81641258	14.8	17.9	16.0254	0.38831164	155	231.6	186.4433

Experiment	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime
1	7.463681665	163	247	198.738	7.236667465	0	0	0	0	1701.798888
2	7.003176351	125	194	147.899	6.70543056	0	0	0	0	1344.548749
3	7.474031777	165	241	195.904	8.006546322	0	0	0	0	1507.567716
4	7.733440114	162	238	191.308	7.390914648	0	0	0	0	1514.599084
5	7.497993065	156	230	190.719	7.562806291	0	0	0	0	1590.020047
6	7.4290898 5	211	283	247.351	7.879961865	0	0	0	0	1605.112305
7	6.982002506	160	235	193.042	6.538672342	0	0	0	0	1448.200305
8	7.415560936	143	227	178.531	7.675482982	0	0	0	0	1449.578342
9	7.61484839	188	250	218.694	7.33909831	0	0	0	0	1583.98909
10	7.047382493	213	262	232.595	6.767050687	0	0	0	0	1794.548328
AVG	7.366120715	168.6	240.7	199.4781	7.305073147	0	0	0	0	1553.996285

D.4 Experiment 4

Table D.11: Evo-RoleMinerM: Results of ten experiments for Dataset1 and Healthcare dataset. The values for each objectives' Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|RP|$) are the average minimum in the last Generation of the experiments. The values for Interpretability (INT) are the average maximum in the last Generation of the experiments. The generation (γ), where an individual occurs the first time, which has no violations, is the average out of the ten experiments (the average out of 4, in the other 6 experiments the state has not been reached; ** the average out of 7, in the other 3 experiments the state has not been reached). The time is the average runtime in seconds.*

DATASET1

Exp.	Algorithm	Objectives	Fitness	γ	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|RP\|$	INT	Time (in sec)
4a	NSGA-II	Conf,Accs	[0,0]	53	0	0	4	11.1	13	1	156
4b	NSGA-II*R*	Conf,Accs	[0,0]	23.4	0	0	4	11.2	13.4	1	256
4c	NSGA-II*R* with weights	Viol.,$\|UA+PA\|$	[0,24.7]	26.2	0	0	4	13	12	1	139

HEALTHCARE

Exp.	Algorithm	Objectives	Fitness	γ	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|RP\|$	INT	Time (in sec)
4d	NSGA-II	Conf,Accs	[0,0]	1617.5*	0	0	19.2	238.8	275.7	-	7792
4e	NSGA-II*R*	Conf,Accs	[0,0]	910.43**	0	0	18.6	242.2	274	-	7408
4f	*R* with weights	Viol.,$\|UA+PA\|$	[14.1,299.9]	$>$1000	8.3	5.6	17	149.1	150.1	-	3083

D.4.1 Experiment 4a

Result Data

Table D.12: Results of Experiment 4a: Dataset1, NSGA-II, Setup 4

Experiments	evals	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min
1	1000	0	0	0	0	0	0	0	0	4
2	1000	0	0	0	0	0	0	0	0	4
3	1000	0	0	0	0	0	0	0	0	4
4	1000	0	0	0	0	0	0	0	0	4
5	1000	0	0	0	0	0	0	0	0	4
6	1000	0	0	0	0	0	0	0	0	4
7	1000	0	0	0	0	0	0	0	0	4
8	1000	0	0	0	0	0	0	0	0	4
9	1000	0	0	0	0	0	0	0	0	4
10	1000	0	0	0	0	0	0	0	0	4
AVG	1000	0	0	0	0	0	0	0	0	4

Experiments	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg
1	5	4.047	0.211638843	11	20	13.044	0.556833907	12	19	15.063
2	5	4.141	0.348021551	10	18	11.298	1.426602958	16	21	17.258
3	5	4.235	0.423998821	11	17	13.338	1.18395777	12	21	14.589
4	5	4.054	0.226017698	10	18	10.564	0.834208607	17	22	17.087
5	5	4.093	0.290432436	10	20	12.772	0.827052598	13	22	16.666
6	5	4.046	0.209485083	13	18	13.091	0.496708164	12	18	12.706
7	6	4.187	0.397531131	13	18	13.233	0.581988832	12	21	14.676
8	5	4.034	0.181229137	13	17	13.058	0.353038242	12	16	13.152
9	6	4.101	0.311125377	10	19	13.097	0.921732608	12	20	14.26
10	5	4.038	0.191196234	10	20	13.027	0.813800344	12	21	15.711
AVG	5.2	4.0976	0.279067631	11.1	18.5	12.6522	0.799592403	13	20.1	15.1168

Experiments	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime	Generation when Objectives are reached
1	1.308063836	0.8	1	0.997115	0.013649241	214.36717	48
2	0.695295621	0.78	1	0.972965	0.042219472	211.689995	49
3	1.443633956	0.8	1	0.97817	0.048073913	210.841971	54
4	0.44881065	0.78	1	0.986335	0.024496791	133.880021	57
5	0.979001532	0.78	1	0.98806	0.020969893	131.295422	46
6	0.731822383	0.8	1	0.9969	0.021363286	127.745241	48
7	1.968	0.8	1	0.991426667	0.025174682	132.985224	56
8	0.76347626	0.8	1	0.99728	0.020449978	130.854325	66
9	1.091970696	0.8	1	0.99249	0.029953017	134.178506	51
10	0.807142491	0.78	1	0.996065	0.022124438	132.241537	55
AVG	1.023721743	0.792	1	0.989680667	0.026847471	156.0079412	53

Result Visualizations

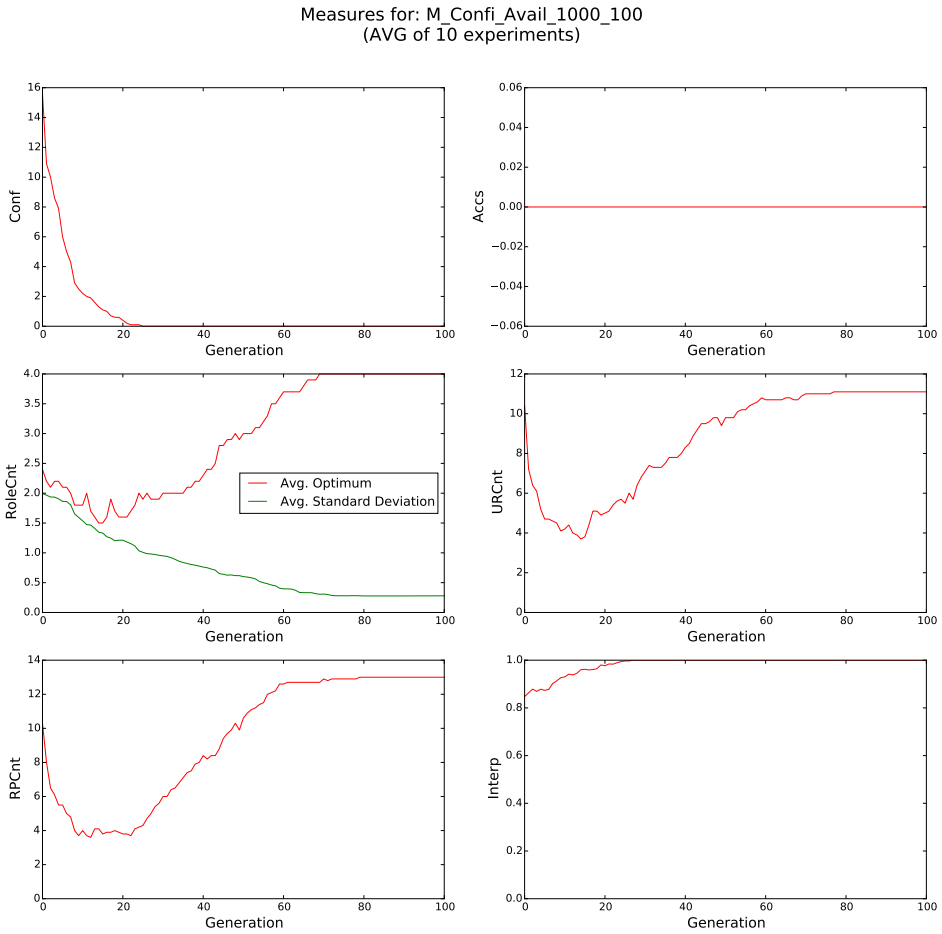


Figure D.5: EXPERIMENT 4a: Average optimum of the single objective measures of the ten experiments with Evo-RoleMinerM on Dataset1

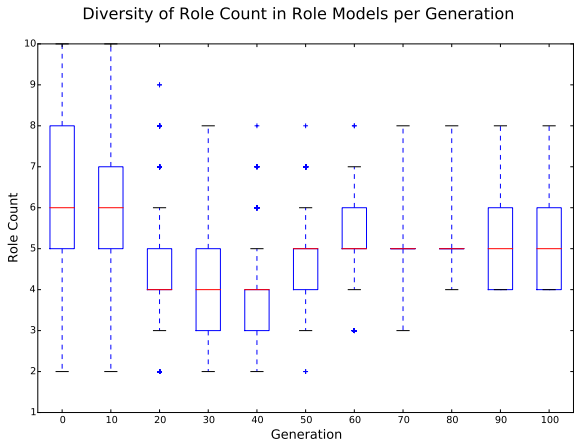


Figure D.6: EXPERIMENT 4a: Example boxplot of role count diversity of individuals of a population in different generations with Evo-RoleMinerM on Dataset1.

D.4.2 Experiment 4b

Result Data

Table D.13: Results of Experiment 4b: Dataset1, NSGA-IIR, Setup 4

Experiments	evals	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min
1	1000	0	0	0	0	0	0	0	0	4
2	1000	0	0	0	0	0	0	0	0	4
3	1000	0	0	0	0	0	0	0	0	4
4	1000	0	0	0	0	0	0	0	0	4
5	1000	0	0	0	0	0	0	0	0	4
6	1000	0	0	0	0	0	0	0	0	4
7	1000	0	0	0	0	0	0	0	0	4
8	1000	0	0	0	0	0	0	0	0	4
9	1000	0	0	0	0	0	0	0	0	4
10	1000	0	0	0	0	0	0	0	0	4
AVG	1000	0	0	0	0	0	0	0	0	4

Experiments	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg
1	6	4.175	0.382589859	13	23	13.611	1.480432032	12	17	13.585
2	6	5.011	0.157730783	10	22	18.866	1.061152204	15	23	16.914
3	6	4.51	0.577840809	10	19	13.347	1.368426469	14	26	17.932
4	5	4.047	0.211638843	12	19	13.087	0.470564555	12	21	14.829
5	5	4.199	0.399248043	10	19	11.584	0.982315632	15	22	17.331
6	6	4.113	0.337980769	13	24	13.517	1.873955976	11	18	12.365
7	6	4.861	0.392019132	11	23	17.992	2.507974482	14	20	16.734
8	6	4.085	0.296268459	10	20	13.06	0.922171351	13	21	16.058
9	6	4.267	0.449122478	13	21	13.363	0.826577885	12	20	14.402
10	6	4.087	0.288844249	10	17	10.85	0.805915628	16	22	17.111
AVG	5.8	4.3355	0.349328342	11.2	20.7	13.9277	1.229948621	13.4	21	15.7261

Experiments	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime	Generation when Objectives are reached
1	1.044401743	0.666666667	1	0.972746667	0.067174154	153.645712	23
2	0.636849	0.78	1	0.957731667	0.01625463	395.101408	24
3	2.182974118	0.78	1	0.936465	0.078063051	412.185388	21
4	0.907611701	0.8	1	0.997465	0.017591156	531.735146	23
5	0.787044471	0.78	1	0.972	0.020654297	401.769795	23
6	0.719565841	0.8	1	0.98738	0.045313011	125.19408	21
7	0.628684341	0.78	1	0.943533333	0.054914115	144.290197	19
8	0.954272498	0.78	1	0.989538333	0.036481589	126.567993	27
9	1.316205151	0.8	1	0.988283333	0.022780541	138.144138	20
10	0.467631265	0.78	1	0.98082	0.017019591	135.587696	33
AVG	0.964447603	0.774666667	1	0.972596333	0.037624614	256.4221553	23.4

D.4.3 Experiment 4c

Result Data

Table D.14: Results of Experiment 4c: Dataset1, NSGA-IIR with weights, Setup 4

Experiments	evals	Violations_Min	Violations_Max	Violations_Avg	Violations_Std	AssignmentCnt_Min	AssignmentCnt_Max	AssignmentCnt_Avg	AssignmentCnt_Std	Conf_Min	Conf_Max	Conf_Avg
1	1000	0	0	0	0	25	25	25	0	0	0	0
2	1000	0	1	0.019	0.136524723	24	26	24.988	0.16079801	0	0	0
3	1000	0	1	0.02	0.14	24	26	24.994	0.184293245	0	0	0
4	1000	0	0	0	0	27	27	27	0	0	0	0
5	1000	0	0	0	0	25	26	25.105	0.306553421	0	0	0
6	1000	0	1	0.014	0.117490425	24	26	25.24	0.458693798	0	0	0
7	1000	0	0	0	0	25	29	25.004	0.126427845	0	0	0
8	1000	0	1	0.041	0.198290191	24	28	24.971	0.245273317	0	0	0
9	1000	0	0	0	0	25	27	25.129	0.361052628	0	0	0
10	1000	0	1	0.058	0.233743449	24	25	24.942	0.233743449	0	0	0
AVG	1000	0	0.5	0.0152	0.082604879	24.7	26.5	25.2373	0.207683571	0	0	0

Experiments	Conf_Std	Accs_Min	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	0	0	0	0	0	4	4	4	0	13	13	13
2	0	0	1	0.019	0.136524723	4	4	4	0	13	13	13
3	0	0	1	0.02	0.14	4	4	4	0	13	13	13
4	0	0	0	0	0	4	4	4	0	10	10	10
5	0	0	0	0	0	4	4	4	0	13	13	13
6	0	0	1	0.014	0.117490425	4	4	4	0	13	13	13
7	0	0	0	0	0	4	5	4.001	0.031606961	13	15	13.002
8	0	0	1	0.041	0.198290191	4	5	4.002	0.044676616	13	14	13.002
9	0	0	0	0	0	4	4	4	0	13	13	13
10	0	0	1	0.058	0.233743449	4	4	4	0	13	13	13
AVG	0	0	0.5	0.0152	0.082604879	4	4.2	4.0003	0.007628358	12.7	13	12.7004

Experiments	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime	Generation when Objectives are reached
1	0	12	12	12	0	1	1	1	0	147.777375	23
2	0	11	13	11.988	0.16079801	1	1	1	0	150.971537	24
3	0	11	13	11.994	0.184293245	1	1	1	0	134.201258	29
4	0	17	17	17	0	1	1	1	0	128.918312	21
5	0	12	13	12.105	0.306553421	1	1	1	0	133.957595	28
6	0	11	13	12.24	0.458693798	1	1	1	0	126.298156	22
7	0.063213923	12	14	12.002	0.063213923	1	1	1	0	159.747057	32
8	0.044676616	11	14	11.969	0.228120582	0.96	1	0.99994	0.00141294	136.910545	30
9	0	12	14	12.129	0.361052628	1	1	1	0	133.527571	25
10	0	11	12	11.942	0.233743449	1	1	1	0	134.280632	28
AVG	0.010789054	12	13.5	12.5369	0.199646905	0.996	1	0.999994	0.000141294	138.6590038	26.2

D.4.4 Experiment 4d

Result Data

Table D.15: Results of Experiment 4d: Healthcare, NSGA-II, Setup 4

Experiments	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min
1	0	2	1	1	0	1	0.5	0.5	20
2	0	5	2.287	2.034362554	0	3	1.357	1.2318892	17
3	0	5	2.526	2.430498714	0	2	1.001	0.971081356	19
4	0	0	0	0	0	0	0	0	18
5	0	0	0	0	0	0	0	0	20
6	0	0	0	0	0	0	0	0	18
7	0	0	0	0	0	0	0	0	20
8	0	1	0.5	0.5	0	2	1	1	15
9	0	2	1	1	0	1	0.5	0.5	21
10	0	0	0	0	0	0	0	0	20
AVG	0	1.5	0.7313	0.696486127	0	0.9	0.4358	0.420297056	18.8

Experiments	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg	URCnt_Std	RPCnt_Min	RPCnt_Max
1	23	21.894	0.398452005	266	303	273.216	3.149181481	303	351
2	19	17.117	0.362368597	224	270	231.296	5.542236372	221	254
3	23	19.183	0.521067174	290	344	297.031	3.800005132	265	299
4	20	18.934	0.282212686	213	246	219.712	1.916	228	282
5	21	20.018	0.132951119	235	267	238.038	1.997136951	299	309
6	19	18.011	0.104302445	203	230	209.828	1.564102298	258	276
7	22	20.969	0.253059282	229	271	239.592	2.420234699	354	366
8	18	16.183	0.597922236	160	210	184.449	11.66350715	237	268
9	22	21.045	0.207304124	278	309	283.795	2.965295095	272	308
10	22	21.013	0.137226091	277	320	293.078	2.063471832	230	265
AVG	20.9	19.4367	0.299686576	237.5	277	247.0035	3.708117101	266.7	297.8

Experiments	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime	Gamma
1	323.73	4.981475685	0	0	0	0	3830.31913	None
2	227.359	3.415570084	0	0	0	0	3443.911226	None
3	276.515	5.038826748	0	0	0	0	3955.861228	None
4	251.3	6.192091731	0	0	0	0	3605.379102	696
5	302.986	1.164389969	0	0	0	0	3999.715123	578
6	262.8	1.299230542	0	0	0	0	3578.616864	774
7	360.512	1.621066316	0	0	0	0	4182.867743	793
8	241.838	2.318567661	0	0	0	0	3246.070478	None
9	278.75	4.371212646	0	0	0	0	3901.098761	None
10	236.304	2.127342004	0	0	0	0	3914.201339	948
AVG	276.2094	3.252977339	0	0	0	0	3765.804099	757.8

Result Visualizations

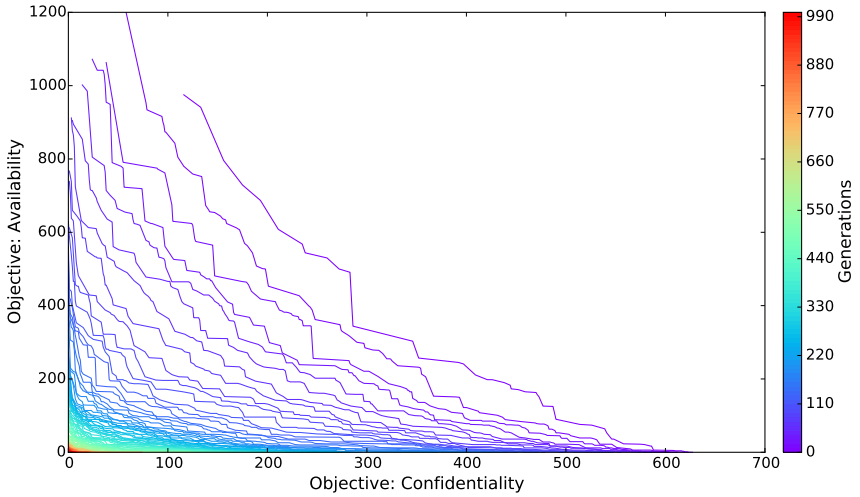


Figure D.7: EXPERIMENT 4d: Example of result of the experiments with Evo-RoleMinerM (based on NSGA-II) on the Healthcare dataset. Illustrates the pareto fronts of each generation.

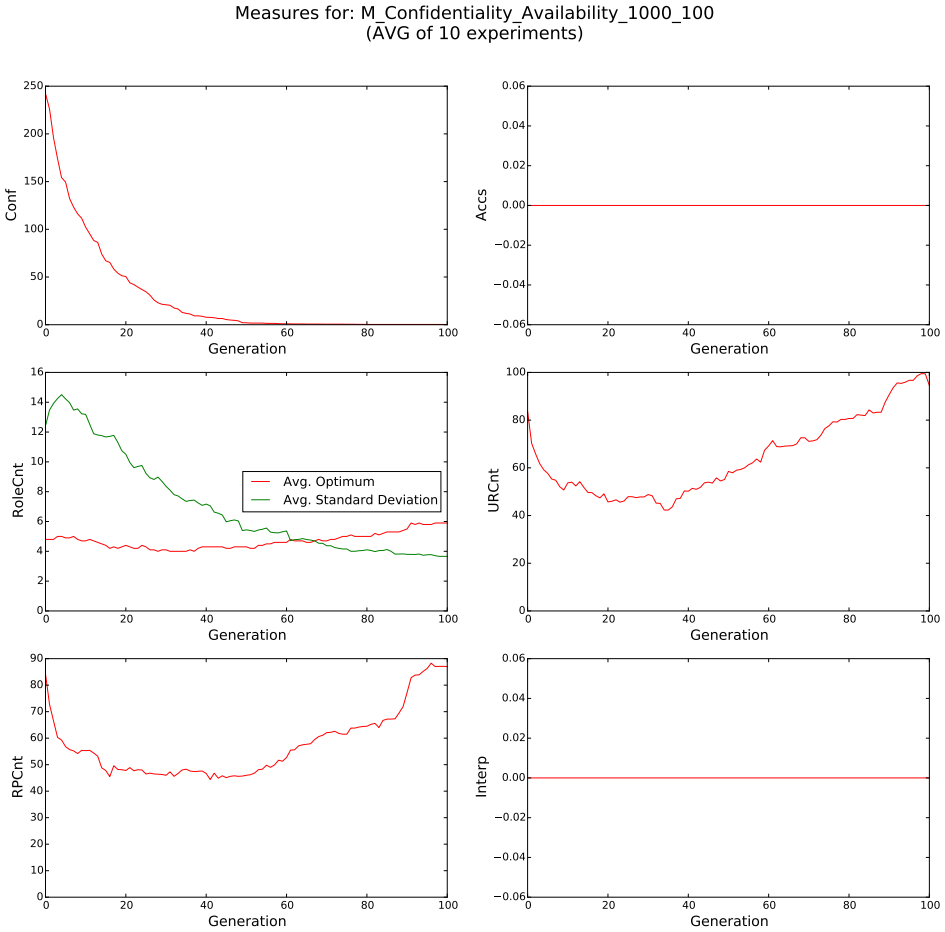


Figure D.8: EXPERIMENT 4d: Average optimum of the single objective measures of the ten experiments with Evo-RoleMinerM on the healthcare dataset. The Interpretability measure is not activated and therefore constantly zero.

D.4.5 Experiment 4e

Result Data

Table D.16: Results of Experiment 4e: Healthcare, NSGA-IIR, Setup 3

Experiments	evals	Conf_Min	Conf_Max	Conf_Avg	Conf_Std	Accs_Min	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min
1	1000	0	0	0	0	0	0	0	0	18
2	1000	0	0	0	0	0	0	0	0	19
3	1000	0	3	1.5	1.1180339887	0	4	1.75	1.4790199457	17
4	1000	0	0	0	0	0	0	0	0	18
5	1000	0	0	0	0	0	0	0	0	18
6	1000	0	0	0	0	0	0	0	0	18
7	1000	0	0	0	0	0	0	0	0	19
8	1000	0	1	0.5	0.5	0	1	0.5	0.5	19
9	1000	0	0	0	0	0	0	0	0	19
10	1000	0	1	0.5	0.5	0	1	0.5	0.5	21
AVG	1000	0	0.5	0.25	0.211803399	0	0.6	0.275	0.247901995	18.6

Experiments	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg
1	19	18.035	0.183779759	227	258	232.01	2.527033834	275	308	279.182
2	21	19.048	0.227367544	284	321	289.44	4.41320745	236	264	240.031
3	22	17.582	0.865607301	258	315	274.498	8.424725277	224	287	238.928
4	22	20.682	0.613902272	234	271	240.715	3.365676009	283	365	338.951
5	20	19.02	0.278567766	189	215	194.045	2.100708214	281	320	287.712
6	21	18.92	0.381575681	229	266	233.833	2.605208437	265	301	271.98
7	21	19.921	0.383091373	254	290	260.874	3.267739892	266	307	275.21
8	21	19.871	0.533253223	208	252	230.128	9.683677814	311	369	343.169
9	20	19.024	0.153049012	252	282	256.847	2.178437743	298	327	302.1
10	23	22.07	0.401372645	287	323	294.353	5.205995678	301	339	313.923
AVG	21	19.4173	0.402156658	242.2	279.3	250.6743	4.377241035	274	318.7	289.1186

Experiments	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime	solutionFound	solutionFound		
1	2.459446279	0	0	0	0	7004.041559	729	"[111	1	729]"
2	2.134019447	0	0	0	0	7270.995842	804	"[77	1	804]"
3	8.305709843	0	0	0	0	7167.064926		"[34	1	None]"
4	13.63020906	0	0	0	0	7805.748457	898	"[55	1	898]"
5	3.779822218	0	0	0	0	6869.585912	1522	"[70	1	1522]"
6	2.89786128	0	0	0	0	7261.956335	696	"[113	1	696]"
7	3.948658	0	0	0	0	7310.509125	1114	"[54	1	1114]"
8	4.443696547	0	0	0	0	7674.168937		"[69	1	None]"
9	1.962651268	0	0	0	0	7585.876887	610	"[48	1	610]"
10	4.916815128	0	0	0	0	8128.169905		"[92	1	None]"
AVG	4.847888907	0	0	0	0	7407.811789	910.4285714			

D.4.6 Experiment 4f

Result Data

Table D.17: Results of Experiment 4f: Healthcare, NSGA-IIR with weights, Setup 3

Experiments	evals	Violations_Min	Violations_Max	Violations_Avg	Violations_Std	AssignmentCnt_Min	AssignmentCnt_Max	AssignmentCnt_Avg	AssignmentCnt_Std	Conf_Min	Conf_Max	Conf_Avg	Conf_Std
1	1000	17	18	17.01	0.099498744	322	324	323.026	0.21289434	15	15	15	0
2	1000	31	31	31	0	279	279	279	0	14	14	14	0
3	1000	10	12	10.004	0.089353232	336	339	338.994	0.134029847	8	8	8	0
4	1000	5	5	5	0	254	255	254.015	0.121552458	1	1	1	0
5	1000	20	20	20	0	277	277	277	0	17	17	17	0
6	1000	11	11	11	0	310	310	310	0	3	3	3	0
7	1000	15	16	15.305	0.460407428	288	289	288.695	0.460407428	10	10	10	0
8	1000	3	4	3.023	0.149903302	304	307	304.987	0.206956517	3	3	3	0
9	1000	15	15	15	0	366	366	366	0	12	12	12	0
10	1000	14	14	14	0	263	264	263.377	0.484634914	0	2	0.002	0.063213923
AVG	1000	14.1	14.6	14.1342	0.079916271	299.9	301	300.5094	0.16204755	8.3	8.5	8.3002	0.006321392

Experiments	Conf_Std	Accs_Min	Accs_Max	Accs_Avg	Accs_Std	RoleCnt_Min	RoleCnt_Max	RoleCnt_Avg	RoleCnt_Std	URCnt_Min	URCnt_Max	URCnt_Avg
1	0	2	3	2.01	0.099498744	20	20	20	0	142	144	143.014
2	0	17	17	17	0	14	14	14	0	147	148	147.004
3	0	2	4	2.004	0.089353232	18	19	18.998	0.044676616	175	176	175.998
4	0	4	4	4	0	16	16	16	0	134	135	134.008
5	0	3	3	3	0	16	16	16	0	133	134	133.001
6	0	8	8	8	0	17	17	17	0	155	157	155.002
7	0	5	6	5.305	0.460407428	15	15	15	0	154	156	154.972
8	0	0	1	0.023	0.149903302	19	19	19	0	150	152	151
9	0	3	3	3	0	21	21	21	0	180	180	180
10	0.063213923	12	14	13.998	0.063213923	14	14	14	0	121	122	121.082
AVG	0.006321392	5.6	6.3	5.834	0.086237663	17	17.1	17.0998	0.004467662	149.1	150.4	149.5081

Experiments	URCnt_Std	RPCnt_Min	RPCnt_Max	RPCnt_Avg	RPCnt_Std	Interp_Min	Interp_Max	Interp_Avg	Interp_Std	Runtime	Generation when Objectives are reached
1	0.160636235	179	181	180.012	0.14091132	0	0	0	0	3149.02411	None
2	0.063118935	131	132	131.996	0.063118935	0	0	0	0	3040.65811	None
3	0.044676616	161	163	162.996	0.089353232	0	0	0	0	3282.632739	None
4	0.08908423	120	121	120.007	0.083372657	0	0	0	0	2768.589949	None
5	0.031606961	143	144	143.999	0.031606961	0	0	0	0	2900.297081	None
6	0.063213923	153	155	154.998	0.063213923	0	0	0	0	3147.467418	None
7	0.170926885	133	134	133.723	0.44751648	0	0	0	0	2815.456229	None
8	0.126491106	153	155	153.987	0.151099305	0	0	0	0	3097.387152	None
9	0	186	186	186	0	0	0	0	0	3531.217362	None
10	0.274364721	142	143	142.295	0.45604761	0	0	0	0	3099.341863	None
AVG	0.102411961	150.1	151.4	151.0013	0.152623557	0	0	0	0	3083.207201	None

D.5 Experiment 5

Table D.18: Evo-RoleMiner: Results of ten experiments for Dataset1. The values for Fitness, Confidentiality violations ($|G_{conf}|$), Availability violations ($|G_{accs}|$), Roles ($|R|$), User-Role-Assignments ($|UA|$) and Role-Permission assignments ($|PA|$) are the average minimum in the last Generation of all experiments. The values for Interpretability (INT) are the average maximum in the last Generation of all experiments. The time is the average runtime in seconds of all experiments.

DATASET1

Experiment	**Fitness Function**	**Fitness**	$\|G_{conf}\|$	$\|G_{accs}\|$	$\|R\|$	$\|UA\|$	$\|PA\|$	**INT**	**Time (in sec)**
5a	$F^{min}_{basic\_INT}$	0.1	1.1	0	3.4	9.9	11.4	1	584
5b	$F^{min}_{edge\_INT}$	0.06	0.6	0	3.7	10.8	11	1	585

Todo list