The Dynamic Role Mining Problem: Role Mining in Dynamically Changing Business Environments

Simon Anderer[1], Tobias Kempter[1], Bernd Scheuermann[1] and Sanaz Mostaghim[2]
[1]*Faculty of Management Science and Engineering, Hochschule Karlsruhe, Moltkestrasse 30, Karlsruhe, Germany*
[2]*Institute for Intelligent Cooperating Systems, Otto-von-Guericke Universität, Magdeburg, Germany*

Keywords: Role Based Access Control, Dynamic Role Mining Problem, Dynamic Evolutionary Algorithm.

Abstract: Role Based Access Control is one of the most frequently used concepts for authorization management in today's business landscapes. The corresponding optimization problem, the so-called *Role Mining Problem (RMP)*, which was shown to be NP-complete, relies in finding a minimal set of roles and a corresponding assignment of those roles to users based on a static user permission assignment. However, as job duties, positions and responsibilities of users in companies constantly change, the corresponding user permission assignment is also subject to changes. Thus, the RMP in its present form has to be extended by dynamically occurring events, representing the changes in business environments. This paper defines the *Dynamic Role Mining Problem (DynRMP)* and presents the most relevant events from business perspective as well as their algorithmic implications for the RMP. Furthermore, several methods to include those events into the framework of an evolutionary algorithm, which is a suitable solution strategy for the RMP, are presented and evaluated in a range of experiments.

1 INTRODUCTION

Role Based Access Control (RBAC) is nowadays one of the widest spread means to avoid erroneous and fraudulent behavior of users in collaborative systems. This is of utmost relevance, since errors and frauds constitute one of the main sources of preventable financial or other damage in enterprises and other organizations (Verizon, 2019). At this, the access rights of users are restricted so that they can only access content and perform actions, which are required to perform the tasks of their work. In contrast to other access control models, RBAC groups permissions to so-called roles which are then assigned to the corresponding users (Sandhu et al., 1996). Especially in large companies or organizations, where direct user-permission assignments can turn out to be quite incomprehensible due to the large number of users and permissions, this approach leads to a substantial reduction of complexity and thus administrative costs.

In the past, role-engineering, which comprises the definition of roles and their assignment to users, was often realized in consultant projects (top-down) lasting months or even years, since they require thorough analysis of organization structures and business processes (Gallaher et al., 2002). In recent years, however, the bottom-up approach, where roles are derived automatically from the prevailing user-permission assignment of the considered organization or enterprise, has become increasingly popular (Mitra et al., 2016). The corresponding optimization problem is called the *Role Mining Problem* (RMP) and is NP-complete (Vaidya et al., 2007). At this, the underlying user-permission assignments are assumed to remain the same during the whole optimization process. However, companies are not static, but they change constantly. Employees change positions and departments, join or leave the company. These occurrences have an impact on a company's IT systems, because users are assigned different permissions depending on their job position. Thus, if role mining software is used in a company, it must be ensured that it reacts to relevant changes as quickly as possible, since an outdated role concept harbors security risks and leads to costs due to additional maintenance effort.

The aim of this paper is to present events, that reflect the dynamically changing business environments of companies and their integration into a formerly static evolutionary algorithm. Especially, in case a new user is joining the company, different role assignment strategies are presented and evaluated. On the one hand, one new role could be created for each

Anderer, S., Kempter, T., Scheuermann, B. and Mostaghim, S.
The Dynamic Role Mining Problem: Role Mining in Dynamically Changing Business Environments.
DOI: 10.5220/0010654000003063
In *Proceedings of the 13th International Joint Conference on Computational Intelligence (IJCCI 2021)*, pages 37-48
ISBN: 978-989-758-534-0; ISSN: 2184-3236

new user, containing all of the new user's permissions and then included it into the optimization process. On the other hand, it might be reasonable to first provide the new user with all roles, whose permissions are a subset of the new user's permission set or with only a selection of those roles, based on a greedy approach, or other key figures of the available roles, before creating a new role. In addition, dynamic role mining is experimentally compared with a static approach in order to show its strength in relation to the presented dynamically occurring events in business environments.

The remainder of this paper is organized as follows. Section 2 introduces to the Role Mining Problem and dynamic optimization problems in general and, building on that, derives a definition of the *Dynamic Role Mining Problem*. Section 3 presents an overview of previously published approaches considering the Role Mining Problem. In Section 4 and 5, the algorithmic consequences of the presented events are described and different role assignment strategies are presented. Section 6 presents the evaluation results of the experiments performed. Section 7 concludes the paper and presents paths for future research.

2 PROBLEM DESCRIPTION

This section introduces to the *Basic Role Mining Problem* and dynamic optimization problems. Based on that the *Dynamic Role Mining Problem* is defined.

2.1 The Role Mining Problem

The Role Mining Problem was firstly defined by Vaidya, Atluri and Guo in 2007 as minimum biclique cover problem (Vaidya et al., 2007). In contrast to that, this paper introduces the RMP as binary matrix decomposition problem (cf. (Anderer et al., 2020)), using the following definitions:

- $U = \{u_1, u_2, ..., u_m\}$ a set of $m = |U|$ users
- $P = \{p_1, p_2, ..., p_n\}$ a set of $n = |P|$ permissions
- $R = \{r_1, r_2, ..., r_k\}$ a set of $k = |R|$ roles
- $UPA \in \{0,1\}^{m \times n}$ the user-permission assignment matrix, where $UPA_{ij} = 1$ implies, that permission p_j is assigned to user u_i
- $UA \in \{0,1\}^{m \times k}$ the user-role assignment matrix, where $UA_{ij} = 1$ implies, that role r_j is assigned to user u_i
- $PA \in \{0,1\}^{k \times n}$ the role-permission assignment matrix, where $PA_{ij} = 1$ implies, that role r_i contains permission p_j.

Based on these definitions, the *Basic Role Mining Problem* can now be described: Given a set of users U, a set of permissions P and a user-permission assignment UPA, find a minimal set of Roles R, a corresponding user-role assignment UA and a role-permission assignment PA, such that each user has exactly the set of permissions granted by the UPA matrix.

$$\boldsymbol{RMP} = \begin{Bmatrix} \min & |R| \\ \text{s.t.,} & \|UPA - UA \otimes PA\|_1 = 0. \end{Bmatrix} \tag{1}$$

where $\|.\|_1$ denotes the L_1-norm for matrices and $\otimes$ the Boolean Matrix Multiplication:

$$(UA \otimes PA)_{ij} = \bigvee_{l=1}^{k} (UA_{il} \wedge PA_{lj}).$$

Figure 1 shows an example of the schematic representation of the UPA, UA and PA matrix, as used throughout the remainder of the paper to illustrate the developed methods and results. The matrices in the figure are based on 3 users, 6 permissions and 4 roles. Black cells indicate 1's, white cells represent 0's.

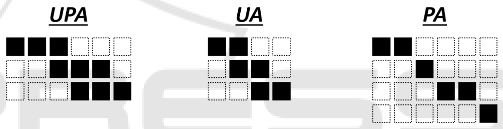

Figure 1: Schematic representation of the UPA, UA and PA matrix.

A tuple $\pi := \langle R, UA, PA \rangle$ can be considered one possible solution to the underlying RMP. If the constraint in (1) is satisfied, the corresponding RBAC solution candidate is denoted *0-consistent*.

There are several further variants of the RMP. In some of them, the 0-consistency-condition is relaxed, which means that the definition of roles and their assignment to users may result in deviations compared to the original user-permission assignment, possibly leading to less roles. In addition to the number of roles and possible deviations, other, more business-driven factors are often considered. Colantonio et al., for instance, consider administrative costs of RBAC schemes (Colantonio et al., 2008), while Molloy et al. as well as Xu and Stoller consider their interpretability (Molloy et al., 2008), (Xu and Stoller, 2012). A broad survey of specifications and definitions of different RMP variants is given in (Mitra et al., 2016).

2.2 Dynamic Optimization Problems

Many real-life optimization problems involve aspects of uncertainty and are subject to constraints or objective functions that change over time. In tour and route planning and the underlying *Traveling Salesperson Problem* and *Vehicle Routing Problem*, for example, the occurrence of new destinations and orders must be taken into account to dynamically adapt the calculated tours and routes to the actual conditions (Guntsch, 2004). In addition, travel time between different destinations can vary based on changing traffic conditions (Anderer et al., 2017). In machine scheduling problems, it is crucial to react to unforeseen events like unexpected machine failures, staff shortages, delayed material deliveries or urgent changes in customer orders (Ouelhadj and Petrovic, 2009), (Anderer et al., 2018).

In its most general form, a dynamic optimization problem can be defined as follows (Cruz et al., 2011):

$$\boldsymbol{DOP} = \begin{Bmatrix} \text{optimize} & f(x,t) \\ \text{s.t.,} & x \in F(t) \subseteq S, t \in T. \end{Bmatrix}$$

where:

- $S \in R^n$, S is the search space.
- $t \in T$ is the time.
- $f : S \times T \to \mathbb{R}$ is the objective function, that aasigns a numerical value $(f(x,t))$ to each possible solution $(x \in S)$ at time t.
- $F(t)$, is the set of feasible solutions $x \in F(t) \subseteq S$ at time t.

3 RELATED WORK

The Role Mining Problem and its different variants are well-studied problems. Many solution techniques have been applied in the last years. One of the first role mining tools is ORCA, which obtains roles by clustering permissions (Schlegelmilch and Steffens, 2005). Likewise, many other approaches are based on the permission grouping concept e.g. (Molloy et al., 2009), (Vaidya et al., 2010). Other authors map the RMP to different other problems known in data mining e.g. the *Matrix Decomposition Problem* (Lu et al., 2008) or the *Set Cover Problem* (Huang et al., 2015). Saenko and Kotenko are the first to apply evolutionary algorithms for role mining (Saenko and Kotenko, 2011). Subsequently, this approach of using evolutionary algorithms was also adopted by (Du and Chang, 2014) and (Anderer et al., 2020). However, none of the above approaches have considered the inclusion of dynamic elements in role mining. One work that comes close to dynamic role optimization is the consideration of the *RBAC Scheme Reconfiguration Problem* (Saenko and Kotenko, 2017). In addition to the initial user-permission assignment UPA_0 and a corresponding RBAC scheme $\pi_0 = \langle R_0, UA_0, PA_0 \rangle$, a further user-permission assignment UPA_1 is considered which reflects the authorization status at a later point in time. The challenge of the *RBAC Scheme Reconfiguration Problem* consists in finding matrices ΔUA and ΔPA, reflecting changes in user-role and role permission assignment, such that:

$$\begin{Bmatrix} \min & \|\Delta UA\|_1 + \|\Delta PA\|_1 \\ \text{s.t.} & (UA_0 \oplus \Delta UA) \otimes (PA_0 \oplus \Delta PA) = UPA_1. \end{Bmatrix}$$

However, in this approach, the changes in the user-permission assignment are aggregated over a certain period of time and then processed all at once, whereas dynamic role mining aims at handling these business-driven events in real time. An overview on dynamic data mining in general is given by (Du et al., 2016).

4 DYNAMIC ROLE MINING

In order to be able to react to dynamically occurring changes and events in business environments, these must be examined in more detail. Therefore, this section firstly defines the Dynamic Role Mining Problem to give a formal framework for dynamic role mining. Secondly, the dynamically occurring events and corresponding event handling methods are described.

4.1 The Dynamic Role Mining Problem

The definition of the Dynamic Role Mining Problem basically coincides with that of the Basic RMP. However, in order to represent the dynamic changes in the business environment, variables must be modeled as time-dependent. Based on the notations introduced in Section 2.1 and Section 2.2, the dynamic Role Mining Problem can be defined as follows:

$$\boldsymbol{DynRMP} = \begin{Bmatrix} \min & |R(t)| \\ \text{s.t.} & \|UPA(t) - UA(t) \otimes PA(t)\|_1 = 0, t \in T. \end{Bmatrix}$$

As in the static case, the optimization objective is to minimize the total number of roles, which coincides with the number of rows of $PA(t)$ respectively the number of columns of $UA(t)$. At this, it should be noted that each user must be assigned exactly the permissions that he or she requires as given by $UPA(t)$.

4.2 Events and Event Handling

One goal of dynamic optimization is to be able to react to changes as quickly as possible. Therefore, these should be forwarded to the optimization algorithm, if possible in real time. One advantage of the dynamically occurring events in role mining consists in the fact that they provide lead time for the optimization algorithm. At this, a distinction is made between the time of the occurrence of the information on the future event $t_{e_i,1}$ and the time of the actual occurrence $t_{e_i,2}$ of event e_i (see Figure 2).

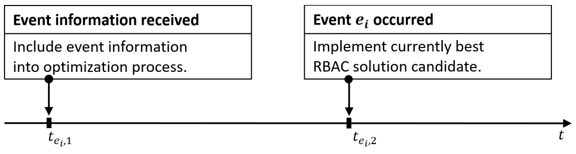
Event information received
Include event information into optimization process.
Event e_i occurred
Implement currently best RBAC solution candidate.
$t_{e_i,1}$
$t_{e_i,2}$
t

Figure 2: Event occurrence.

This means that the optimization algorithm is usually informed about the upcoming event before it occurs, such that event information can already be included in the optimization process at $t_{e_i,1}$. At $t_{e_i,2}$, simply the currently best RBAC solution candidate has to be selected and implemented as new authorization concept of the company.

In the further course of this chapter, several events, corresponding event-handling methods and their integration into an evolutionary algorithm are presented. To determine if a new event is currently pending, this is checked at the beginning of each new iteration. If this is the case, the corresponding event-handling method is called to adapt the individuals of the current population of the evolutionary algorithm to the new conditions of the business environment. Since, the event-handling methods are independent of the other methods of the evolutionary algorithm, they can be included in each evolutionary role mining algorithm. A schematic representation of the integration of the event-handling methods into the sequential process of the evolutionary algorithm is given in Figure 3.

Most companies fill a majority of their positions at least twice. This is necessary so that employees can substitute for each other in the event of vacation or illness. But also in case of employees leaving the company, the dual staffing of positions prevents hindrances in the operational process. From an algorithmic perspective, users that have exactly the same set of permissions can be considered as one row in the UPA matrix (see Figure 4).

The rows of the UPA matrix therefore no longer correspond to individual users but to whole classes of users with the same permission set. In order to be

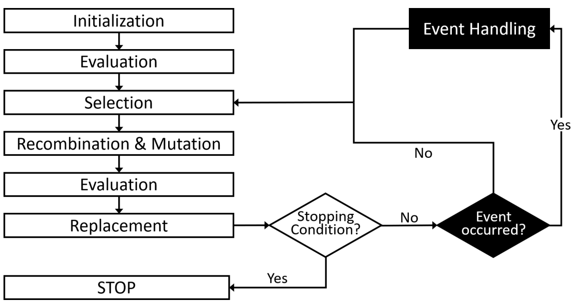

Figure 3: Integration of the event-handling methods into an evolutionary algorithm.

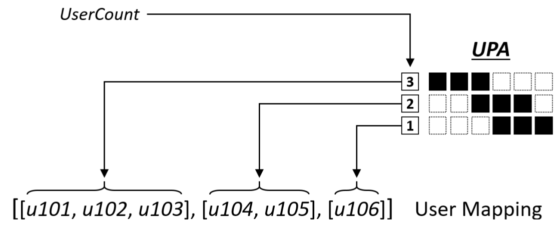

Figure 4: Exemplary user mapping.

able to identify all users even after aggregation, each user is assigned a unique UserID. Subsequently, the UserIDs corresponding to each user class are stored in a separate user mapping. For further modeling of the events, however, only the number of users per class, plays a role and is therefore added as *UserCount* to each row of the UPA (and UA) matrix.

4.2.1 The *UserJoinsCompany*-Event

Whenever a new employee joins a company, it must be decided which permissions he or she should receive. In order for the evolutionary algorithm to process this, the *UserJoinsCompany*-event is triggered, which includes information on the new user's permission set as well as the event times. Based on that, it can be checked whether the new user has the identical permission set as any of the already existing users. At this, two different cases are distinguished:

Case 1: Permission Set of New User Equals Permission Set of Existing User. If the new user corresponds to one of the existing users, there is no need to add a new row to the UPA and UA matrix, as this user is already assigned the same permissions as the new user. Thus, simply the corresponding row in the UPA and UA matrix needs to be identified and the associated user count has to be increased by one. In addition, the new user has to be added to corresponding class of the user mapping. For an example see Figure 5.

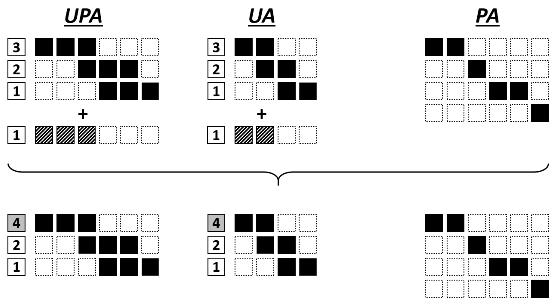

Figure 5: Exemplary handling of *UserJoinsCompany*-event Case 1.

Case 2: Permission Set of New User Does Not Equal Permission Set of Existing User. If the new user does not correspond to an existing user, at first, a new row is added to the *UPA* matrix containing all of the user's permissions. Subsequently, it is checked, whether the new user can be assigned already existing roles from the *PA* matrix. Again, two cases can be distinguished:

Case 2.1: Permission Set of New User Can Be Covered Completely by Existing Roles. In this case, existing roles are assigned to the new user in order to provide him or her with the required permissions. This can be achieved in several ways. For example, the user can simply be assigned all roles whose permissions form a subset of his or her permission set (see Figure 6). In contrast to this, a greedy approach could be applied, such that the roles are evaluated and ranked in terms of their contribution in covering user's permission set. Both of these as well as other role-assignment methods are described in more detail in Section 5.

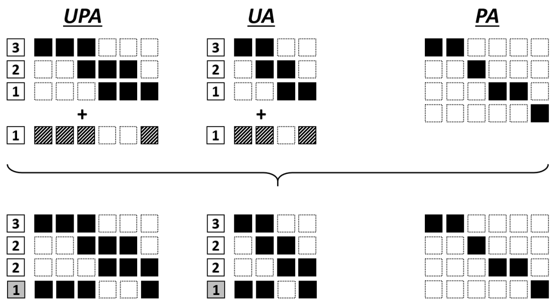

Figure 6: Exemplary handling of *UserJoinsCompany*-event Case 2.1.

Case 2.2: Permission Set of New User Cannot Be Covered Completely by Existing Roles. If not all of the new user's permissions can be covered by existing roles, a new role must be created for this user. This role can either directly include all of the user's permissions and be the only role assigned to the new user, or an attempt is first made to cover the user's permissions set as best as possible using existing roles and only the remaining permissions are used for the new role. An example for the second approach is given in Figure 7. In both cases, this means that the new role must be added to the *PA* matrix as additional row as well as to the *UA* matrix as additional column.

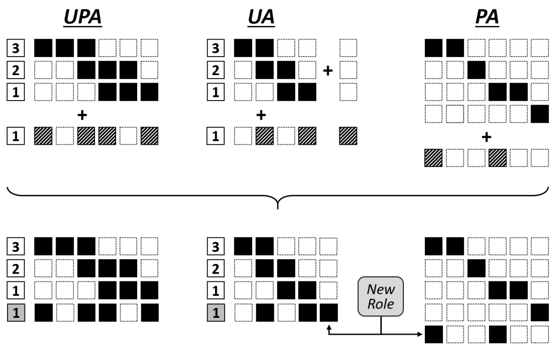

Figure 7: Exemplary handling of *UserJoinsCompany*-event Case 2.2.

4.2.2 The *UserLeavesCompany*-Event

The *UserLeavesCompany*-event is in some ways the opposite of the *UserJoinsCompany*-event. While a new user leads to an increase in the number of users in the user mapping, the departure of a user leads to a reduction in the number of users. If a new user can cause the need to create a new role, the departure of a user may lead to the removal of one or more roles.

As soon as the information about the imminent departure of a user is transmitted, this information can be included into the optimization process. This means that the user can be directly deleted from the UPA and *UA* matrix but is then written to a so-called *TemporaryUserList*. This list can be considered a technical auxiliary tool to ensure users who will leave the company to access their permissions directly and independently of the current role concept. In this way, they can continue to do their work until the day of their departure, without affecting the further role optimization process. Again, two different cases must be considered:

Case 1: Permission Set of Leaving User Equals Permission Set of Existing User. This corresponds to the simplest case of this event. If there are several users with the same permission profile, and one of them is leaving the company, this has no influence on the UPA, *UA* and *PA* matrix. Only the $UserCount$ and user mapping have to be adjusted in such a way that the UserID of the affected user is removed. This does not have any relevant effects on the optimization process. An example can be found in Figure 8.

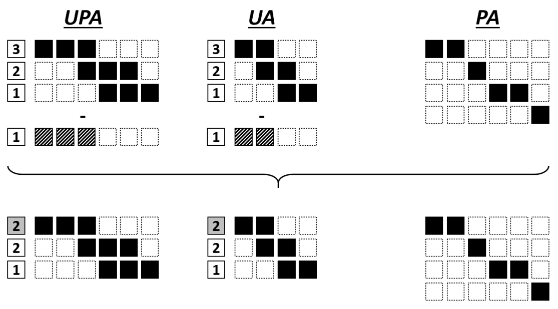

Figure 8: Exemplary handling of *UserLeavesCompany*-event Case 1.

Case 2: Permission Set of Leaving User Does Not Equal Permission Set of Existing User. In case the leaving user has a unique permission set, there is a corresponding row in the *UPA* and *UA* matrix (with user count 1), that can now be removed. In addition, the roles that were assigned to the leaving user must be reviewed. In case that one of these roles was assigned to only the leaving user, it becomes obsolete and can be deleted from the *PA* and *UA* matrix, removing the corresponding matrix row of the *PA* matrix respectively the corresponding column of the *UA* matrix. An example is given in Figure 9.

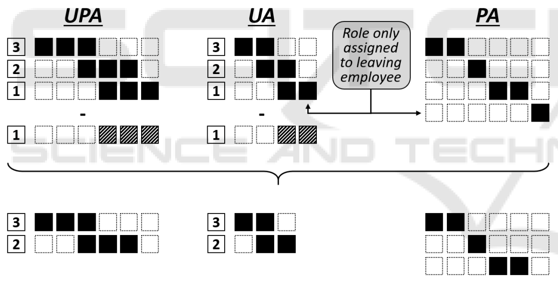

Figure 9: Exemplary handling of *UserLeavesCompany*-event Case 2.

4.2.3 The *PositionChange*-Event

The change of position of a user takes place within the context of relocations and promotions. In this context, it is common for users to continue to complete work from their previous job position, such that the user is assigned the permissions of the old and the new position at the same time during a certain transition period. This can lead to compliance conflicts, if the user is allowed to execute transactions, that may not normally be controlled by the same person. It is therefore of highest importance to ensure that the permissions of the old job position are revoked after the transition period has expired.

The handling of the *PositionChange*-Event can be achieved by applying the event-handling methods of the two previous events in parallel. As soon as the information about the impending change of position becomes known, the user concerned will be added to the optimization process using the methods of the *UserJoinsCompany*-event, but with unchanged UserID. At the same time, the permissions of the old job position are moved to the *TemporaryUserList*, using the methods of the *UserLeavesCompany*-event (again with unchanged UserID). This ensures that the permissions, that the user will hold in his future position, can already be part of the role optimization process, while the permissions of the user's old position lose their influence on the optimization process immediately. At the beginning of the transition period, the user is granted the permissions of the new job position by the roles obtained from the *UA* matrix of the currently best RBAC solution candidate and the permissions of the old position from the *TemporaryUserList*. At the end of the transition period, the user is removed from the *TemporaryUserList* such that only the permissions required for the new job position remain in the user's permission set.

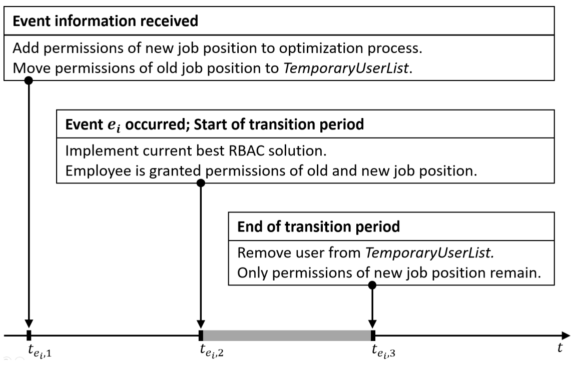
Event information received
Add permissions of new job position to optimization process.
Move permissions of old job position to TemporaryUserList.
Event e_i occurred; Start of transition period
Implement current best RBAC solution.
Employee is granted permissions of old and new job position.
End of transition period
Remove user from TemporaryUserList.
Only permissions of new job position remain.
$t_{e_i,1}$
$t_{e_i,2}$
$t_{e_i,3}$
t

Figure 10: Schematic handling of *PositionChange*-event.

4.2.4 The *PermissionRequest*-Event

The permission request is an event that occurs when a user realizes that he or she lacks authorizations to perform the tasks given. In such cases, it must be possible for users to request missing permissions autonomously. For this purpose, many companies use the concept of permission requests on which the user concerned can report which permissions are required additionally to complete the work. Subsequently, this is reviewed by a supervisor and either approved or rejected. In case of approval, the additional permissions assigned to the requesting user have immediate impact on the underlying role concept.

Even though a user usually requests only a few additional permissions, this event can also be processed using the mechanisms of the *UserJoinsCompany*-event. The difference between *PermissionRe-*

quest- and the *PositionChange*-event consists in the fact that the user should in any case continue to have the authorizations he already has. In addition, there is no transition period during which two job positions are held simultaneously. Therefore, entries on the *TemporaryUserList* are not necessary in this case.

5 ROLE ASSIGNMENT

As shown in Section 4, all events can be processed using the methods of the *UserJoinsCompany*- and the *UserLeavesCompany*-event. However, since the *UserLeavesCompany*-event can be implemented in a straight forward fashion, the main focus of this section lies on the *UserJoinsCompany*-event. Whenever a new user has a unique permission set, i.e. no user with the same permissions exists in the UPA matrix yet, the new user's permissions must be assigned either via the already existing roles of the *PA* matrix or via a new role, or sometimes via a combination of the two (see Section 4.2.1). For this purpose, different role assignment methods are described in this section. An evaluation of these methods in the framework of an evolutionary role mining algorithm is presented in Section 6.

***ORFA* - One Role for All.** This represents the simplest method to provide the user with permissions needed. At this, one role is created containing all of the new user's permissions and then added as additional row to the *PA* matrix and as additional column to the *UA* matrix. This is based on the assumption that the applied role mining algorithm will continue to optimize the created additional *ORFA*-roles automatically in the course of the further optimization process.

***AAR* - Assign All Roles.** Since it is possible that the permission set can completely be covered by the existing roles of the *PA* matrix (Case 2.1 of the *UserJoinsCompany*-event), it can be reasonable to provide the user with these roles and thereby prevent the unnecessary creation of a new role. For this, *AAR* is a straight forward method, as it assigns all roles $(PA)_j$ to the new user which are a subset of the new user's permission set $(UPA)_{m+1}$:

$$\sum_{k=1}^{n} UPA_{m+1,k} \cdot PA_{k,j} = \sum_{k=1}^{n} PA_{k,j} \tag{2}$$

In the case, that not all permission can be covered by existing roles (Case 2.2 of the *UserJoinsCompany*-event), a new role, containing the uncovered permissions of the new user, is created and added to the *PA* and *UA* matrix.

***ARR* - Assign Random Roles.** It may be useful not to assign all suitable roles to the new user, as it is possible that the required permissions can already be covered with fewer roles. Therefore, in addition to the subset condition (Equation 2), the contribution of the role to the covering of the remaining permissions of the new user is also considered. At this, iteratively a random role is selected. However, this role is only assigned to the user if it covers at least one of the user's remaining uncovered permissions. In case there is no role left that has positive contribution, the role assignment procedure is stopped and, if necessary, a new role with the remaining uncovered permissions of the new user is created.

***GREEDY* - Greedy Selection.** Sometimes it is desirable to assign as few roles as possible to the new user. Therefore a GREEDY approach is applied. It follows basically the same procedure as the method above. However, the roles are not selected randomly, but instead in each step the role is assigned to the new user that provides the greatest contribution to covering remaining uncovered permissions (and fulfills the subset condition in Equation 2). Again, if there is no role left that has positive contribution, the role assignment procedure is stopped and, if necessary, a new role with the remaining uncovered permissions of the new user is created.

***ABP* - Assign by Popularity.** The *ABP* method is based on the assumption that *popular* roles are good roles in terms of minimizing the total number of roles. *Popular* in this context means that they are assigned to many users. The less often a role is assigned to a user, the more likely it is to be dropped during the further optimization process. Therefore, in each step, of all the roles that satisfy the subset condition (2), the role is assigned to the new user, which has the highest *popularity* i.e. is assigned to the largest number of other users. As for the other methods, if there is no role left that has positive contribution, the role assignment procedure is stopped and, if necessary, a new role with the remaining uncovered permissions of the new user is created.

***ABS* - Assign by Similarity.** It can be assumed that the permission set of new users has great similarity to the permissions of the users from the same area or division of the company. Thus, *ABS* is a role assignment method based on the similarity between the new user and existing users. To determine the similarity of the permission sets *M* and *N* of two users, the Jaccard coefficient, which is a common measure for the similarity of sets, is used:

$$\text{Jaccard}(M,N) = \frac{|M \cap N|}{|M \cup N|}.$$

Based on this, the new user is iteratively assigned all roles of the next most similar user, which satisfy the subset condition (2), until there is no role left with positive contribution.

6 EVALUATION

In this section, the presented methods are evaluated in two different test setups. At first, the role-assignment methods are evaluated. Subsequently, the advantages of dynamic role mining compared to the implementation of a static role concept are demonstrated in the context of a second experiment. As in Section 5, the main focus here is again on the *UserJoinsCompany* event. All experiments were conducted on different benchmark instances of *RMPlib* (Anderer et al., 2021). Since no other approaches for dynamic role optimization are known so far, the focus of this section is on the evaluation of the role-assignment methods and the advantages of dynamic optimization.

6.1 The addRole-EA

All evaluated methods were integrated into an existing evolutionary algorithm for the Role Mining Problem, the *addRole-EA*. At this, each individual consists of two binary matrices $UA(t)$ and $PA(t)$. In each iteration, new roles are added to the individuals (Mutation) and existing roles are exchanged between individuals (Crossover). These are then assigned to all users, whose permission set is a superset of the permissions contained in the roles at the given time (0-consistency). Subsequently all roles, that became obsolete by the addition of the new roles, are deleted from the individuals, resulting in a reduction of the total number of roles, which serves as objective function of the *addRole-EA*. The different event-handling methods are integrated into the optimization algorithm as described in Section 4 without changing its general framework. A detailed description of the *addRole-EA* is given in (Anderer et al., 2020).

6.2 Evaluation of Role-assignment Methods

As a basis for the evaluation of the role-assignment methods, three different benchmark instances with different numbers of users were selected (*PLAIN_small_02*, *PLAIN_small_05* and *PLAIN_medium_01*). In all three cases, different numbers of new users were added at the beginning (Position 1), around the middle (Position 2), and towards the end (Position 3) of the role optimization process (see Figure 11). For this purpose, the benchmark instances were reduced in size by randomly selecting a number of users *usersAtStart*, that represent the existing employees at the beginning of the optimizing process. The remaining users then serve as the basis for dynamically adding new users during the optimization process.

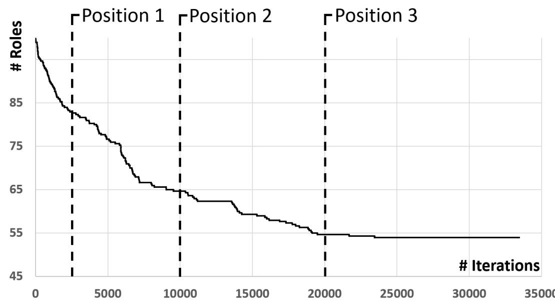

Figure 11: Positions of the adding of users exemplaryly at *PLAIN_small_05*.

The parameters of the different test setups are shown in Table 1. At this, the tests were repeated 20 times with different random seeds for each parameter combination. Results were averaged.

Table 1: Parameter values for the evaluation of role-assignment methods.

	PLAIN_ small_02	PLAIN_ small_05	PLAIN_ medium_01
Original Number of Users	50	100	500
usersAtStart	40	75	350
Users added at Iteration	2,500 12,500 25,000	2,500 10,000 20,000	5,000 10,000 20,000
Number of Users added	5, 10	5, 10, 20	10, 20, 50

Due to the initial reduction of the benchmark instances, all new users that are added during the optimization process correspond to either Case 2.1 or Case 2.2 of the *UserJoinsCompany*-event (see Section 4). Since Case 1 has no influence on the optimization process from an algorithmic point of view, the selected evaluation scenario still covers the entire event. Table 2 shows the percentage of new roles compared to the number of new users. If this equals 0%, no new role had to be created (Case 2.1). If it equals 100%, one new role had to be created for every new user (Case 2.2). The position specifications refer to the various times at which new roles are added (see Figure 11 & Table 1). It becomes apparent, that hardly any new roles need to be created in the case of *PLAIN_medium_01* at all three positions. In the other benchmark instances, the existing set of roles can also be used in a considerable number of cases to completely cover the permissions of a new user.

Table 2: Percentual average of new roles in each test scenarios.

Benchmark	add. Users	add. Roles (%)	add. Roles (%)	add. Roles (%)
PLAIN_small_02	5	28.00%	68.00%	77.65%
PLAIN_small_02	10	33.33%	72.50%	77.00%
PLAIN_small_05	5	27,00%	46.00%	45.00%
PLAIN_small_05	10	13.00%	38.00%	42.00%
PLAIN_small_05	20	10,50%	26.00%	32.75%
PLAIN_medium_01	10	0.50%	1.00%	0.50%
PLAIN_medium_01	20	0.50%	1.75%	1.75%
PLAIN_medium_01	50	0.60%	0.70%	1.80%
		Position 1	Position 2	Position 3

Across all test scenarios, it turned out that the *ORFA*-method performs the worst. In almost all cases, the added *ORFA*-roles were reduced at a significantly slower rate. In some cases, the added *ORFA*-roles could in fact not be further optimized at all. Figure 12 shows the representative course of role optimization using *ORFA*-role -assignment at *PLAIN_small_02*, adding 10 roles at iteration $12,500$.

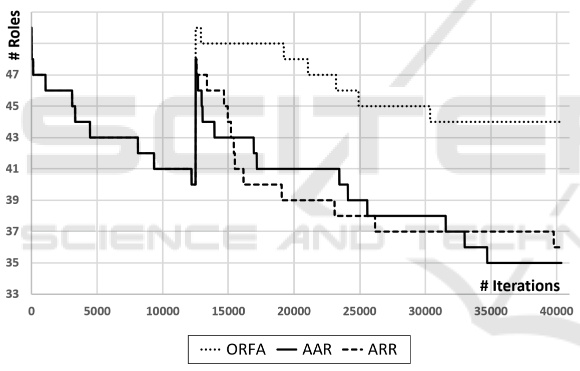

Figure 12: Representative course of role optimization with *ORFA* compared to *ARR* and *AAR*.

For a more detailed comparison of the different role-assignment methods, the integral over four fixed iteration intervals (500, 1,000, 5,000 and 10,000 iterations), each starting at the iteration of the arrival of the new users (Positions 1-3), was calculated, in order to evaluate the speed of the optimization process subsequent to the addition of the new users depending on the choice of the role-assignment method:

$$\int_{\text{Position}_i}^{\text{Position}_i+k} r(x^*(\tau),\tau)d\tau,$$

where $k \in \{500, 1000, 5000, 10000\}$ and $r(x^*(\tau),\tau)$ denotes the number of roles of the best individual $x^*(\tau)$ at iteration τ. Subsequently, the different role-assignment methods were ranked based on the average integral values over all runs of each test case (Figure 13).

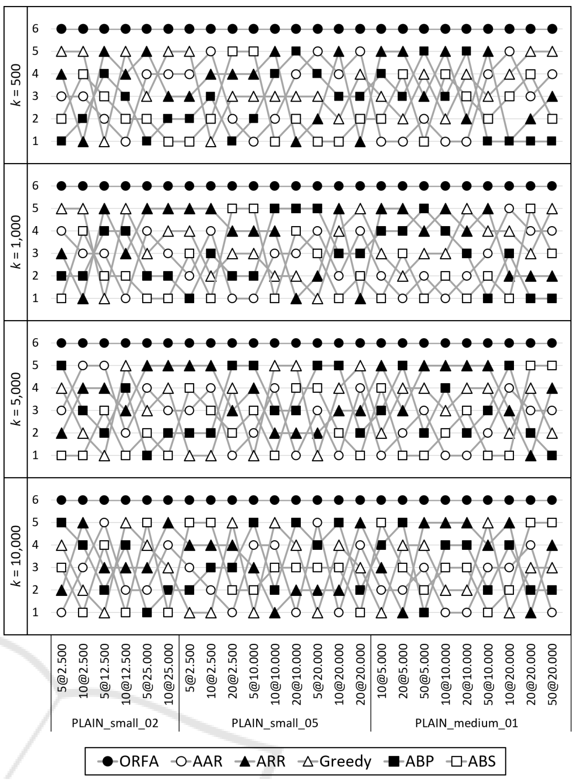

Figure 13: Ranking of role-assignment methods.

For a more detailed statistic consideration, Table 3 shows the mean values and the standard deviations of the ranks across all test cases of *PLAIN_small_02*, *PLAIN_small_05* and *PLAIN_medium_01*.

Table 3: Mean values and standard deviations of ranks across all test cases.

	ORFA	AAR	ARR	GREEDY	ABP	ABS
Rank (Mean)	6.0	2.75	3.521	3.208	3.0	2.51
SD	0.0	1.323	1.399	1.376	1.369	1.392

This again underlines the poor performance of the ORFA method (rank 6 of 6 in all test cases). For the more sophisticated role-assignment methods (*GREEDY*, *ABP* and *ABS*), the similar mean values combined with the high standard deviation in each case suggest, that none of them performs significantly better than the two straight-forward methods (*AAR* and *ARR*). This shows that, while it is important to consider the existing roles in handling the *UserJoins Company*-event, the choice of method is not a critical factor.

6.3 Dynamic vs. Static Role Mining

In practice, once a good role concept has been found and implemented, the contained roles usually remain unchanged over time. However, this static approach is inflexible and can lead to the unnecessary creation of roles in case of the *UserJoinsCompany*-Event. It may therefore be advisable to consider role mining as an ongoing dynamic process to be able to consider the dynamically occurring events in the role mining process.

In order to examine this more closely, again new users are added to the reduced benchmark instances (*PLAIN_small_02*, *PLAIN_small_05* and *PLAIN_medium_01*). In the static case, at a certain point in time, the current best PA matrix (in terms of the number of roles) and the contained roles are fixed as PA_{static}. From this point on, a certain number of new users is added at regular iteration intervals based on a given frequency. Whenever a new user joins the company, it is checked, whether his or her permission set can be covered by the roles contained in PA_{static} (Case 2.1 of the *UserJoinsCompany*-event). If this is not the case, a new role containing the uncovered permissions is created as in Case 2.2 and added to PA_{static}. In the dynamic case, the new user is integrated into the optimization process via the event *UserJoinsCompany* (see Section 4). Since no role assignment method has proven to be particularly performant, the *ARR*-method is used exemplarily. The parameters underlying the different test cases are shown in Table 4. Each test setup was repeated 20 times with different random seeds. Results were averaged.

Table 4: Parameter values for the evaluation of role-assignment methods.

	PLAIN_small_02	**PLAIN_small_05**	**PLAIN_medium_01**
Original Number of Users	50	100	500
$usersAtStart$	40	75	350
PA_{static} fixed at Iteration	15,000	15,000	50,000
Number of Users added	1, 2	2, 4	10, 20
Every ... Iterations	2,500	5,000	10,000

The final average results of all test cases are shown in Figure 14. In all test cases, it is evident that dynamic role optimization produces significantly fewer roles. While the new roles are simply added in the static case (staircase-shaped curve), the curve associated with dynamic optimization also increases at the corresponding positions, but then drops again significantly. At this, the average reduction ranges from 4.63% (*PLAIN_medium_01*, 20 users every $10,000$ iterations) to 11.60% (*PLAIN_small_02*, 2 users every $2,500$ iterations) compared to the static approach. This is due to the fact that during dynamic role optimization, already existing roles, which are adapted to the current user structure, can be modified in such a way, that they can also be assigned to the joining users.

7 CONCLUSION AND FUTURE WORKS

This paper presented the *Dynamic RMP* as extension to the well-known Role Mining Problem including the consideration of dynamically changing business environments. Four different event types, comprising new employees joining the company, employees leaving the company, position changes and permission requests, are presented and corresponding event-handling methods are described. Especially for the *UserJoinsCompany*-event different role-assignment methods are presented and evaluated. At this, it became apparent, that it is more effective to build on the existing roles at the time of the new employee's company entry, rather than creating a separate role for each new employee (*ORFA*). One possible reason could be the role structure underlying the benchmark instances of *RMPlib*. In real-world scenarios, where it can be assumed that new and existing users are more similar, especially the *ABS*-method could achieve better results. Furthermore, it is possible that the chosen evolutionary algorithm, in which the methods were embedded, had impact on the evaluation results. Hence, all methods should be examined in the context of other role mining algorithms.

In addition, initial experiments were conducted to distinguish between static and dynamic role mining, showing that the inclusion of the dynamically occurring events into the optimization process results in a substantial reduction of roles. This needs to be investigated in more detail in the future. In particular, the frequency and number of the added users as well as the correlation to the selected role-assignment method are of great interest in this context.

Moreover, since the role mining problem is of great practical relevance, it is desirable to apply and examine the developed methods, in particular including the event-handling methods for all of the described events, in real-life business use-cases.

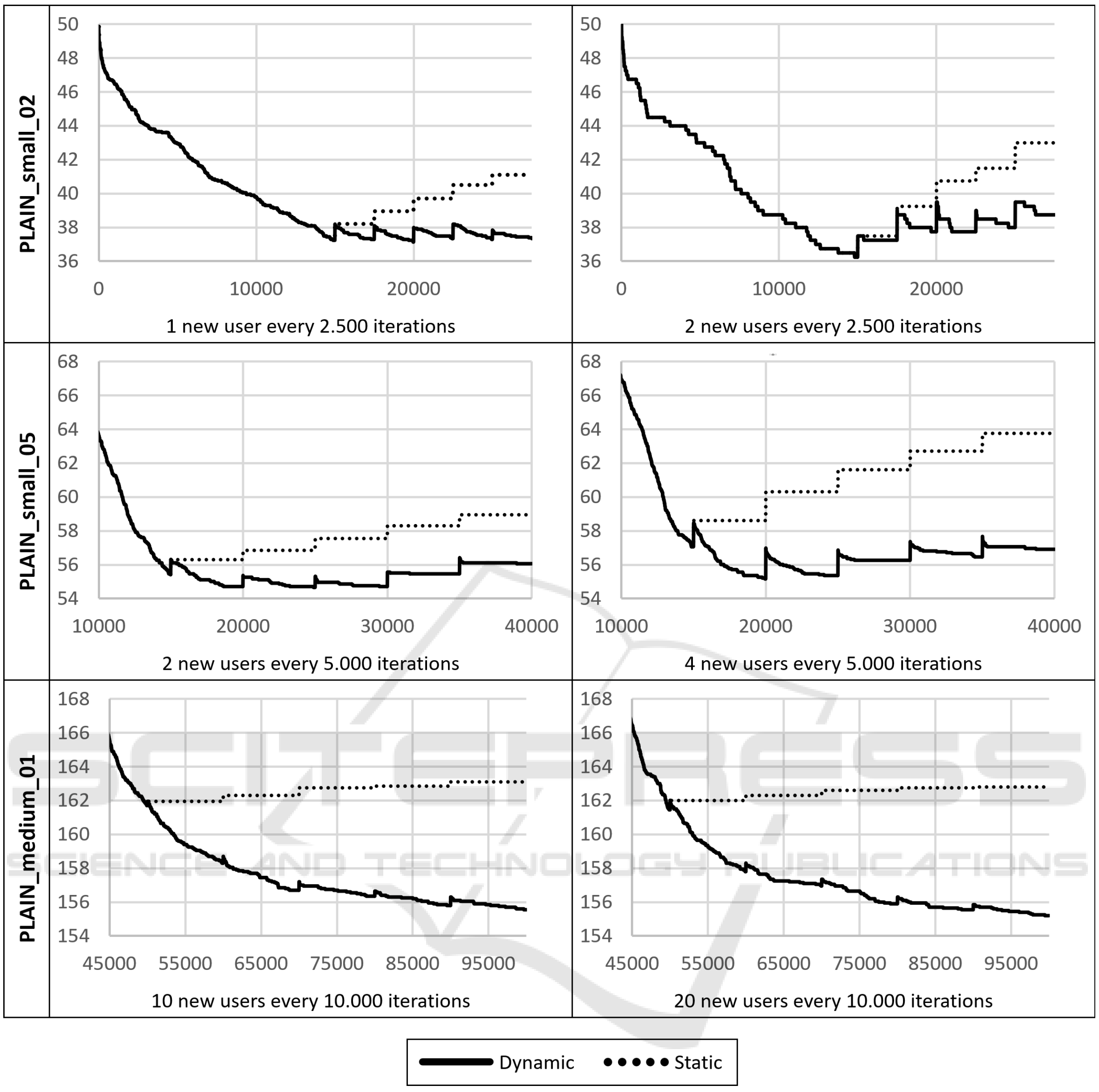

Figure 14: Average results of all test cases.

REFERENCES

Anderer, S., Halbich, M., Scheuermann, B., and Mostaghim, S. (2017). Towards real-time fleet-event-handling for the dynamic vehicle routing problem. In *Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI,*, pages 35–44. INSTICC, SciTePress.

Anderer, S., Kreppein, D., Scheuermann, B., and Mostaghim, S. (2020). The addrole-EA: A new evolutionary algorithm for the role mining problem. In *Proceedings of the 12th International Joint Conference on Computational Intelligence, IJCCI 2020, Budapest, Hungary, November 2-4, 2020*, pages 155–166. SCITEPRESS.

Anderer, S., Scheuermann, B., Mostaghim, S., Bauerle, P., and Beil, M. (2021). Rmplib: A library of benchmarks for the role mining problem. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, SACMAT '21, page 3–13, New York, NY, USA. Association for Computing Machinery.

Anderer, S., Vu, T.-H., Scheuermann, B., and Mostaghim, S. (2018). Meta heuristics for dynamic machine scheduling: A review of research efforts and industrial requirements. In *IJCCI*, pages 192–203.

Colantonio, A., Di Pietro, R., and Ocello, A. (2008). A cost-driven approach to role engineering. In Wainwright, R. L. and Haddad, H. M., editors, *Proceedings of the*

2008 ACM symposium on Applied computing - SAC '08, pages 2129–2136, New York, USA. ACM Press.

Cruz, C., González, J. R., and Pelta, D. A. (2011). Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, 15(7):1427–1448.

Du, J., Zhou, J., Li, C., and Yang, L. (2016). An overview of dynamic data mining. In *2016 3rd International Conference on Informative and Cybernetics for Computational Social Systems (ICCSS)*, pages 331–335.

Du, X. and Chang, X. (2014). Performance of ai algorithms for mining meaningful roles. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2070–2076. IEEE.

Gallaher, M., O'Connor, A., Kropp, B., and Tassey, G. (2002). The economic impact of role-based access control. Technical report, National Institute of Standards and Technology (NIST).

Guntsch, M. (2004). *Ant algorithms in stochastic and multi-criteria environments*. PhD thesis, Karlsruhe Institute of Technology.

Huang, H., Shang, F., Liu, J., and Du, H. (2015). Handling least privilege problem and role mining in rbac. *Journal of Combinatorial Optimization*, 30(1):63–86.

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

Mitra, B., Sural, S., Vaidya, J., and Atluri, V. (2016). A survey of role mining. *ACM Computing Surveys*, 48(4):1–37.

Molloy, I., Chen, H., Li, T., Wang, Q., Li, N., Bertino, E., Calo, S., and Lobo, J. (2008). Mining roles with semantic meanings. In Ray, I. and Li, N., editors, *Proceedings of the 13th ACM symposium on Access control models and technologies - SACMAT '08*, pages 21–30, New York, New York, USA. ACM Press.

Molloy, I., Li, N., Li, T., Mao, Z., Wang, Q., and Lobo, J. (2009). Evaluating role mining algorithms. In Carminati, B. and Joshi, J., editors, *Proceedings of the 14th ACM symposium on Access control models and technologies - SACMAT '09*, pages 95–104, New York, New York, USA. ACM Press.

Ouelhadj, D. and Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling*, 12(4):417–431.

Saenko, I. and Kotenko, I. (2017). Reconfiguration of rbac schemes by genetic algorithms. In Badica, C., El Fallah Seghrouchni, A., Beynier, A., Camacho, D., Herpson, C., Hindriks, K., and Novais, P., editors, *Intelligent Distributed Computing X*, volume 678 of *Studies in Computational Intelligence*, pages 89–98. Springer International Publishing, Cham.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E. (1996). Role-based access control models. *Computer*, 29(2):38–47.

Schlegelmilch, J. and Steffens, U. (2005). Role mining with orca. In Ferrari, E. and Ahn, G.-J., editors, *Proceedings of the tenth ACM symposium on Access control models and technologies - SACMAT '05*, pages 168–176, New York, New York, USA. ACM Press.

Vaidya, J., Atluri, V., and Guo, Q. (2007). The role mining problem. In Lotz, V. and Thuraisingham, B., editors, *Proceedings of the 12th ACM symposium on Access control models and technologies - SACMAT '07*, pages 175–184, New York, New York, USA. ACM Press.

Vaidya, J., Atluri, V., Warner, J., and Guo, Q. (2010). Role engineering via prioritized subset enumeration. *IEEE Transactions on Dependable and Secure Computing*, 7(3):300–314.

Verizon (2019). Data breach investigations report 2019. *Computer Fraud & Security*, 2019(6):4.

Xu, Z. and Stoller, S. D. (2012). Algorithms for mining meaningful roles. In Atluri, V., Vaidya, J., Kern, A., and Kantarcioglu, M., editors, *Proceedings of the 17th ACM symposium on Access Control Models and Technologies - SACMAT '12*, pages 57–66, New York, New York, USA. ACM Press.