Optimal Employee Recruitment in Organizations under Attribute-Based Access Control

ARINDAM ROY, Goa Institute of Management, India
SHAMIK SURAL and ARUN KUMAR MAJUMDAR, Indian Institute of Technology, Kharagpur, India
JAIDEEP VAIDYA and VIJAYALAKSHMI ATLURI, Rutgers University, USA

For any successful business endeavor, recruitment of a required number of appropriately qualified employees in proper positions is a key requirement. For effective utilization of human resources, reorganization of such workforce assignment is also a task of utmost importance. This includes situations when the underperforming employees have to be substituted with fresh applicants. Generally, the number of candidates applying for a position is large, and hence, the task of identifying an optimal subset becomes critical. Moreover, a human resource manager would also like to make use of the opportunity of retirement of employees to improve manpower utilization. However, the constraints enforced by the security policies prohibit any arbitrary assignment of tasks to employees. Further, the new employees should have the capabilities required to handle the assigned tasks. In this article, we formalize this problem as the Optimal Recruitment Problem (ORP), wherein the goal is to select the minimum number of fresh employees from a set of candidates to fill the vacant positions created by the outgoing employees, while ensuring satisfiability of the specified security conditions. The model used for specification of authorization policies and constraints is Attribute-Based Access Control (ABAC), since it is considered to be the *de facto* next-generation framework for handling organizational security policies. We show that the ORP problem is NP-hard and propose a greedy heuristic for solving it. Extensive experimental evaluation shows both the effectiveness and efficiency of the proposed solution.

Categories and Subject Descriptors: H.1.2 [**Models and Principles**]: User/Machine Systems

General Terms: Employee Assignment Optimization, Separation of Duty

Additional Key Words and Phrases: Role-based access control (RBAC), statically mutually exclusive roles (SMER) constraint, graph coloring, greedy algorithm

Research reported in this publication was supported by the National Science Foundation under awards CNS-1624503 and CNS-1747728 and the National Institutes of Health under awards R01GM118574 and R35GM134927. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.
Authors' addresses: A. Roy, Big Data Analytics, Goa Institute of Management, Sanquelim, Goa, 403505, India; email: roy.arindam469@gmail.com; S. Sural and A. K. Majumdar, Department of Computer Science and Engineering, IIT Kharagpur, West Bengal, 721302, India; emails: {shamik, akmj}@cse.iitkgp.ernet.in; J. Vaidya and V. Atluri, Management Science and Information Systems Department, Rutgers University, Newark, New Jersey, 07102, USA; emails: jsvaidya@rbs.rutgers.edu, atluri@rutgers.edu.

2158-656X/2021/01-ART6 $15.00
https://doi.org/10.1145/3403950

ACM Reference format:
Arindam Roy, Shamik Sural, Arun Kumar Majumdar, Jaideep Vaidya, and Vijayalakshmi Atluri. 2021. Optimal Employee Recruitment in Organizations under Attribute-Based Access Control. *ACM Trans. Manage. Inf. Syst.* 12, 1, Article 6 (January 2021), 24 pages.
https://doi.org/10.1145/3403950

1 INTRODUCTION

Operational efficiency of any commercial organization is highly dependent on its proficiency to manage the workforce. A key task towards optimizing workforce utilization involves recruitment of the right number of proper personnel to fill the vacant positions. Moreover, if need arises, underperforming employees have to be replaced with better substitutes. In order to keep the organization more productive and profitable, the opportunity to fill the vacancies created by a number of employees who drop out or retire with new candidates is used to upgrade the overall manpower deployment scenario. However, different policies prevailing in an organization to govern access rights may often constrain assignments of such type.

Access control models are used to mediate access of the secured resources to the subjects. Development of a variety of access control models depending on the specificities of different environments has taken place as the outcome of a significant amount of work in this field [Miller and Baldwin 1990; Bell and Lapadula 1976; Sandhu et al. 1996; Wang and Li 2010]. Although Role-Based Access Control (RBAC), which grants access to objects based on the roles assigned to its users, has been successful in terms of its popularity and is still being used by diverse commercial information systems [Zhao et al. 2015; Fuchs et al. 2011; Zhu and Zhou 2008], it falls short in certain situations.

There are several important limitations of RBAC that have been identified over the years including those mentioned by Hu et al. in a NIST special publication [Hu et al. 2014] and in a white paper by *Axiomatics*, a leading vendor of fine-grained access control solutions [Axi 2016]. Specifically, RBAC is not suitable for systems where an access decision is needed without having prior knowledge of the subjects and when the context of an access request can influence the decision. For example, the access decision could depend on the role of an employee, object type, and location and time from which the access has been requested. With the continually increasing size of resource sets used in various information systems, the problem of permission explosion may arise in RBAC systems as individual objects are used to specify the permissions. RBAC is generally designed for a scenario where role assignments are based on largely fixed organizational human resource positions. Trying to deploy RBAC for systems where dynamic access decisions are needed would require creation of numerous *ad hoc* roles with limited membership, leading to a potential role explosion. It is observed from business cases that RBAC may also lead to "Toxic Combinations" of role assignments. For example, one user could be assigned a role that allows her to create a purchase order and another that allows her to approve it, posing a significant business risk if not managed properly.

The Attribute-Based Access Control (ABAC) model, which has recently been proposed, overcomes these limitations [Hu et al. 2014; Axi 2016] of RBAC. The request by a user to perform a particular operation on an object is governed in an ABAC system on the basis of environmental conditions as well as the attribute values assigned to the users and the object. Policies to mediate the access requests are defined as authorization rules in terms of attribute values and conditions [Hu et al. 2014]. ABAC provides support for access control decision making without *a priori* knowledge of the object by the subject or knowledge of the subject by the object owner. Being based only on the notion of subject and object attributes, ABAC does not require any form of direct assignment of authorizations to individual subjects prior to an access request, thus overcoming

the problem of role and permission explosion. The problem of toxic combination of roles in RBAC is also eliminated through dynamic authorizations supported in ABAC [Axi 2016]. Moreover, by avoiding management of access control lists, roles, and groups, ABAC brings a lot of flexibility in large enterprises [Hu et al. 2014].

Traditional access control models including RBAC may be viewed as special cases of ABAC as they can be specified in terms of its components. A significant body of related work exists that looks into various facets of efficient deployment of ABAC in different commercial systems [Hüffmeyer and Schreier 2016; Benkaouz et al. 2016; Hsu and Ray 2016; Xu and Stoller 2015; Ferraiolo et al. 2016].

From the perspective of the utilization of human resources to the fullest extent, organizations would like to allow employees to handle all the tasks they are capable of. However, this may violate the security constraints and pose a contradiction. Specifically, two of the widely used constraints in access control systems are Separation of Duty (SoD) [Clark and Wilson 1987; Saltzer and Schroeder 1975] and Binding of Duty (BoD) [Tan et al. 2004; Schefer et al. 2012; Strembeck and Mendling 2011] constraints. SoD restricts an employee from carrying out multiple tasks that may lead to a conflict of interest, whereas BoD requires an employee to handle all or none of a specified set of tasks. For example, an SoD constraint in a banking system can specify that the same employee should not be able to issue a check and authorize it. A BoD constraint in an academic institute can specify that a full professor has to be a member of the institute senate. Often, model-specific security constraints are used on top of SoD and BoD constraints. The Cardinality constraint, Capability constraint [Roy et al. 2016], Prerequisite constraint [Sandhu et al. 1996], and Statically Mutually Exclusive Roles (SMER) constraint [Li et al. 2007] are some of those used in RBAC. The user attribute values that a user can hold are specified for each user in an ABAC system. Intuitively, it follows that the assignment of tasks will be carried out in an organization according to the capabilities of such users. An ABAC system can also have SoD constraints in terms of environmental conditions.

Increasing global competition forces companies to work smarter. Organizations, including those using specific access control mechanisms, require timely reorganization of their human resources to enable more effective use of their resources. At the same time, an aging workforce is often more expensive though not necessarily more productive [Imhoff and Henkens 1989; Dumay and Rooney 2011]. Reorganization of a workforce on the basis of productivity has proved to be effective in certain scenarios [Smith and Rutigliano 2001]. It is often desirable, as a cost-saving measure, to supplant a set of retiring personnel with a lower number of compensating workers [Goren 2008]. Shedding the unproductive workforce with factors other than age and recruiting a fresh set of employees has also been carried out. However, the set of candidates out of whom a proper selection has to be done is typically quite large. During situations like economic slowdowns, while the number of applicants for a given position tends to increase significantly, organizations have to take up the strategy of "doing more with less" [Tolan n.d.; Starich 2019]. Thus, the importance of selecting an optimum number of new recruits increases several-fold. Also, recruitment may be even more important in a dire economy where companies need to equip themselves with any and every competitive edge to be in the best position in order to survive and "weather the storm" [Goldbeck 2019]. Several workforce optimization techniques have been proposed in the literature and implemented in practice [Bergh et al. 2013; Schmenner and Lackey 1994; Ryan and Macky 1998; Franco 2013]. However, to the best of our knowledge, none of the studies take into account the underlying access control policies and mechanism, which is a significant deficiency given the ubiquity and necessity of access control in organizations of all sizes. In this article, we deal with the problem of optimizing workforce by replacement of a section of personnel with a minimum number of eligible candidates in an enterprise where ABAC is used as the access control mechanism.

A potential application of the current work would be in situations similar to the use case presented by *Axiomatics* [Frisch 2014], which relates to an organization in which the access permissions depend on the project phase. The organizational policy is as follows: "Project members should have access to project specification documents. If the project is in the planning phase, they can create, read, edit and delete documents. If the project is in the production phase, a project steering committee must take a formal decision before a change can be made to project specifications. Project documents tagged as Public Information should be generally available." The policy requirements make the authorization rules overly complex if role-based access control is used. Instead, ABAC can be deployed to effectively handle this. Any recruitment taking place in such an organization in the event of an economic slump would require solutions for optimizing the number of recruited workers. Deployment of solutions proposed in this article will be able to solve this realistic need for recruitment optimization.

As we use ABAC, this work is also applicable for *ad hoc* environments like freelancing platforms where the subjects are not static and may not be known to the system a priori. A project is floated on such platforms and application or bid from freelancers is invited. The freelancers are not contract bound and may leave the project in between or may be asked to step down due to nonperformance. Hence, new freelancers have to be deputed intermittently for different tasks of the project based on their capabilities and environment conditions (configuration of devices used, location, etc.). It is to be noted that such scenarios cannot be modeled using RBAC due to its limitations, as discussed before. However, because ABAC is a generalized model, specifications of other models including DAC, MAC, and RBAC are subsumed in it.

To summarize, the key aspects of the problem studied in this article are as follows:

- We aim to minimize the number of new recruits needed to fill the vacancies from among a pool of candidates in an organization having a deployed access control mechanism.
- The selection of the new recruits is done in the presence of the following security constraints:
  - Separation of Duty
  - Binding of Duty
  - User Capability

The problem and the constraints handled in this work are discussed in detail in the subsequent sections. Some of the key reasons behind identifying and attempting to solve this particular problem are as follows:

- Appropriate selection of new recruits is an important strategy towards enabling workforce optimization.
- The problem of optimizing recruitment while handling the access control requirements has so far not been studied in the literature, to the best of our knowledge.

The organization of the rest of this article is as follows. ABAC is formally defined in Section 2. We provide the definitions of various constraints in ABAC that are later considered in this work in Section 3. Section 4 introduces the Optimal Recruitment Problem and provides a detailed explanation using an illustrative example. Complexity analysis of the introduced problem is carried out in Section 5, and a solution is proposed in Section 6. Results obtained from the experimental evaluation are presented in Section 7. Research related to the work carried out in the current article is discussed in Section 8. Finally, the conclusion is drawn and directions for future work are discussed in Section 9.

2 ATTRIBUTE BASED ACCESS CONTROL (ABAC)

In ABAC, access rights on objects are accorded to users by means of authorization policies based on the attributes of the user placing the request, the object being requested, and the environment conditions. In this section, we re-brief a formal definition of ABAC also used in our previous work [Roy et al. 2019].

Following are the basic components constituting ABAC [Hu et al. 2014]:

- U, O, and E: Represent a set of users, sensitive objects, and environment specifications, respectively. Members of these sets are represented as u_i, o_i, and e_i, respectively, for $1 \leq i \leq |X|$, where X represents U, O, or E respectively.
- U_A, O_A, and E_A: Represent a set of user attributes, object attributes, and environment attributes, respectively. Members of these sets are represented as ua_j, oa_j, and ea_j, respectively, for $1 \leq j \leq |X_A|$ (as earlier, X represents U, O, or E, respectively). A value can be acquired by each attribute ua_j, oa_j, or ea_j from a corresponding set of attribute values. For instance, let a user attribute *Research Group*, object attribute *Storage Location*, and environment attribute *Location* be associated with the following respective sets of values:
  - {*Robotics*, *Data Sciences*, *Artificial Intelligence*, *Computing Systems*, *Security*}
  - {*DB1*, *DB2*, *DB3*}
  - {*Philadelphia*, *Washington D.C.*, *Chicago*}

  Here, each user (u_i), object (o_i), and environment specification (e_i) can have its attributes *Research Group*, *Storage Location*, and *Location* assigned to any of the possible values from the associated sets.
- Three set of functions, $F_U: U \times U_A \rightarrow \{v_j | v_j$ is a user attribute value$\}$, $F_O: O \times O_A \rightarrow \{v_j | v_j$ is an object attribute value$\}$, and $F_E: E \times E_A \rightarrow \{v_j | v_j$ is an environment attribute value$\}$. An instance of each of these respective functions is as follows:
  - F_U(*Sophia*, *Research Group*) = *Data Sciences*, if an R&D employee *Sophia* is part of the research group Data Sciences.
  - $F_O(f_i$, *Storage Location*) = *DB2*, if a file f_i is located in the database *DB2* of an organization.
  - $F_E(e_i$, *Location*) = *Chicago* represents that an environment e_i has its attribute *Location* as *Chicago*.
- A: Represents a set of all possible operations (actions) denoted as a_i that can be carried out in the system. For example, the set A = {*insert*, *create*} denotes that the only possible operations on a file are *insert* and *create*.
- P: Represents a set of authorization policies (also called authorization rules) denoted as pa_i represented by a 4-tuple, for $1 \leq i \leq |P|$. The members of the 4-tuple of a construct $\langle uc, oc, ec, a\rangle$ are user conditions, object conditions, environment condition, and action $a \in A$, respectively [Jha et al. 2018]. A user condition, object condition, or environmental condition is denoted by equalities of construct $n = c$, where n is an attribute name and c is either *any* or an attribute value. It is to be noted that, if an attribute name n is assigned to a value *any*, it becomes irrelevant from the perspective of making access decisions. An authorization policy thus denotes that an operation a can be executed on the objects with object condition oc by the users with the user condition uc in an environment with the environment condition ec.

While a user invokes an access request on an object, a search is carried out in the set P of authorization rules. The access is then granted or denied on the basis of whether a suitable rule facilitating it is found or not. For instance, suppose an organization has a condition that only a user from location *Philadelphia* and project type *Communications* can insert entries in the activity log for the office in *Philadelphia* of the company *Comcast* from a computer at his office. This requirement can be encoded as a rule in an ABAC system as follows: $\langle$\{(Project Type = *Communications*)\},

{(type = *Activity Log*), (Company = *Comcast*), (Office = *Philadelphia*)}, {(Location = *Philadelphia*)}, *insert*⟩.

A permission p that can be defined as a tuple (a_i, oc) in an ABAC system imperatively provides authority to execute actions on specified varieties of objects. Here, the action a_i can be executed over the objects with oc as its object condition. The set Per consists of all such specified permissions.

3 SECURITY CONSTRAINTS IN ABAC

To enforce certain security policies, ABAC has to embrace different constraints. In this section, we discuss such important constraints that have been considered in this work.

While involving a least specified number of employees to execute a task is enforced by an SoD constraint, a BoD constraint coheres an employee to a specified set of permissions. For an ABAC system, utilizing the recently enunciated idea of permission, SoD and BoD can be defined as follows.

Definition 1 (Separation of Duty (SoD) Constraint). In an ABAC system, an SoD constraint is denoted by $\langle\{p_1, \ldots, p_n\}, l\rangle$, where $\{p_1, \ldots, p_n\} \subseteq Per$ and l is an integer. It is considered to be satisfied if a minimum of l users is required to perform a job that requires obtaining all the permissions in the set $\{p_1, \ldots, p_n\}$.

Definition 2 (Binding of Duty (BoD) Constraint). In an ABAC system, a BoD constraint is denoted by a set $\{p_1, \ldots, p_n\}$ of permissions. It is considered to be satisfied if a user gains admittance to all or none of the permissions in the set.

Considering the set EC of environmental conditions, a separation of duty can also be dependent of the environment attributes.

Definition 3 (Environment Dependent Separation of Duty (EDSoD) constraint). An EDSoD constraint in an ABAC system is denoted by $\langle\{p_1, \ldots, p_n\}, \{ec_1, \ldots, ec_m\}, k\rangle$, where $\{p_1, \ldots, p_n\}$ is a set $PR \subseteq Per$ of permissions, $\{ec_1, \ldots, ec_m\}$ is a set $E \subset EC$ of environmental conditions, and k is an integer. It is said to be satisfied if at least k users are required to perform a task that requires all the permissions in the set PR and access to the set E of environmental conditions.

In other words, an EDSoD constraint $\langle PR, E, k\rangle$ is said to be satisfied in an ABAC system if the SoD constraint $\langle PR, k\rangle$ is satisfied for all the users who have access to the environmental conditions in set E. An EDSoD constraint $\langle PR, EC, k\rangle$ is equivalent to an SoD constraint $\langle PR, k\rangle$ as the set EC considers all possible environmental conditions in the system.

It has been shown by Jha et al. [2018] that verification of SoD constraints for ABAC systems is intractable and hence they have developed efficient techniques to convert an SoD constraint to Mutually Exclusive Roles (MER) constraints. The verification of a set of MER constraints is proven to be solvable in polynomial time. The definition of MER constraints is as follows.

Definition 4 (Mutually Exclusive Roles (MER) Constraint). An MER constraint in an ABAC system, denoted by $\langle R, k\rangle$, where $R \subseteq P$ is a set of authorization rules, and k is an integer where $1 < k \leq |R|$, is said to be satisfied if no user in the system has access to k or more rules of the set R.

For each SoD constraint, a set of MER constraints is generated in two steps. First, the SoD is converted into a corresponding set of Separation of Rules (SoR) using a procedure *GenerateSoR*. Second, the intermediate SoRs are converted into a set of MERs using procedure *GenerateMER*, thus completing the process. It is to be noted that this work can even be used with minor modification to

convert an EDSoD constraint to a set of MER constraints in two similar steps. *GenerateSoR* takes the SoD constraint and the set P of rules as an input. First, this procedure is used with a modification in the inputs to convert an EDSoD constraint to a set of SoRs, and the modifications in the inputs for an EDSoD $\langle PR, E, k\rangle$ are as follows. Let P be the rule base of the considered ABAC system. Only the set $P' \subset P$ of the rule base is considered where any of the environmental conditions from the set E are mentioned. The SoD considered as input is $\langle PR, k\rangle$. Thus, the input $\langle PR, k\rangle$ and P' can be fed to the *GenerateSoR* procedure to convert each EDSoD into a set of SoRs. Once a set of SoRs is obtained, it can be given as input to the procedure *GenerateMER* to generate the corresponding set of MER constraints. Hence, it is a fair assumption that MERs can be used in an ABAC system to enforce EDSoD constraints.

The notion of an attribute-value pair (av pair) is used in this work [Roy et al. 2019]. A user u is considered to be assigned to an av pair $av_{ij} \in AV$ if $F_U(u, a_i) = v_j$. A matrix $UA \subseteq U \times AV$ is used to denote a many-to-many user to av pair assignment relation. Here, $UA(u, av) = 1$ if av is assigned to the user u, and $UA(u, av) = 0$ otherwise. Using these notions, the User Capability constraint is defined as follows.

Definition 5 (User Capability (UC) Constraint). A set of UC constraints is represented by a matrix $UC \subseteq U \times AV$, where $UC(u, av_{ij}) = 1$ if the user u is capable of acquiring the av pair av_{ij}. Likewise, $UC(u, av_{ij}) = 0$ denotes that the user u does not have the capability of getting assigned to av_{ij}.

A fresh set Y' of users can *fill* the vacant positions created by a set $X \subseteq U$ of users if the set Y' of new users can acquire all the av pairs allocated to the set X of users such that all the specified constraints in the ABAC system are satisfied. That is, $\forall k, \forall i, \exists j,\ UA(x_i, av_k) \implies UA(y_j, av_k)$, where $x_i \in X$, $y_i \in Y'$, and $av_k \in AV$.

4 PROBLEM DEFINITION

We now present an introduction to the problem of filling the vacant positions with a minimum number of employees from a given set of candidates with different capabilities in an ABAC system satisfying the MER, BoD, and capability constraints. It is worth noting that a set of SoD constraints or a set of EDSoD constraints can be efficiently converted to a set of MER constraints using the procedure proposed in Jha et al. [2018]; hence, for simplicity, we define the ORP problem in terms of MER constraints. The formal definition of the problem is as follows:

Definition 6 Optimal Recruitment Problem (ORP). Given a set U of users, find the minimum number of candidates from the set Y required to *fill* the vacancies made by the set $X \subseteq U$ of employees, satisfying a set MER of MER constraints, set BC of BoD constraints, and the user capability constraint UC in an ABAC system.

In other words, an instance of the ORP problem seeks to obtain the minimum number of candidates from the set Y needed to *fill* the vacancies created by the set X of employees in the ABAC system such that all the specified constraints in the ABAC system are satisfied. It is important to remark that the unique condition that allows a user to get allocated to an av pair is satisfying the specified security constraints that are not in terms of environment attributes as in this problem. Hence, for the simplicity of the solution, it is justified to consider authorization policies devoid of environment conditions and use a static authorization matrix AM to represent them. The notion of authorization matrix AM is also used in Roy et al. [2019] and is described as a many-to-many user-condition-to-permission assignment relation. Formally, it can be written as $AM \subseteq UCond \times Per$, where $UCond$ denotes the set consisting of all user conditions used in the authorization rules. However, the definition of the AM matrix can be easily tweaked to consider the environment conditions as follows. AM can be defined as a many-to-many relation of tuple

Table 1. UA Matrix for Example 4.1

	av_{11}	av_{12}	av_{13}	av_{21}	av_{22}	av_{31}	av_{32}	av_{33}
u_1	0	1	0	0	1	0	1	0
u_2	1	0	1	1	0	0	0	0
u_3	0	1	1	1	0	1	0	0
u_4	0	1	0	0	1	1	1	0
u_5	0	0	0	0	0	0	0	1

Table 2. UC Matrix for Example 4.1

	av_{11}	av_{12}	av_{13}	av_{21}	av_{22}	av_{31}	av_{32}	av_{33}
u_{n1}	1	1	1	1	0	1	0	1
u_{n2}	1	0	0	0	0	0	0	0
u_{n3}	1	1	1	1	1	1	1	0
u_{n4}	0	0	1	1	0	0	0	0
u_{n5}	0	1	0	0	1	1	1	1

Table 3. AM Matrix for Example 4.1

	p_1	p_2	p_3	p_4	p_5
uc_1	1	0	1	0	1
uc_2	1	0	1	0	0
uc_3	0	1	0	1	0
uc_4	0	1	0	1	0
uc_5	0	0	0	0	1

Table 4. UA' Matrix for Example 4.1

	av_{11}	av_{12}	av_{13}	av_{21}	av_{22}	av_{31}	av_{32}	av_{33}
u_{n1}	0	1	1	1	0	1	0	1
u_{n3}	1	1	1	1	1	1	1	0
u_{n5}	0	1	0	0	1	1	1	0

$(UCond, EC)$ of all possible pairs of user conditions and environment conditions considered in the authorization policies to permission assignment relation, that is, $AM \subseteq (UCond, ec) \times Per$.

We next take an illustrative example for the ORP problem as defined above.

Example 4.1. Consider an ABAC system in which a set $X = \{u_1, \ldots, u_5\}$ of employees is stepping down such that the user-to-attribute-value pair relation UA for these employees is as shown in Table 1. The set $Y = \{u_{n1}, \ldots, u_{n5}\}$ is composed of the candidates with different capabilities as shown in Table 2, from whom an optimal selection has to be done to fill in the vacancies. The AM matrix with the set $R = \{r_1, \ldots, r_{10}\}$ of 10 rules is shown in Table 3. Here, $AV = \{av_{11}, av_{12}, av_{13}, av_{21}, av_{22}, av_{31}, av_{32}, av_{33}\}$, $UCond = \{uc_1, \ldots, uc_5\}$, and the definitions of the user conditions are as follows: $uc_1 = \{av_{11}\}$, $uc_2 = \{av_{12}, av_{31}\}$, $uc_3 = \{av_{13}, av_{21}\}$, $uc_4 = \{av_{32}, av_{22}\}$, and $uc_5 = \{av_{33}\}$. Let $MER = \{mer_1, mer_2, mer_3\}$, where $mer_1 = \langle\{r_2, r_4, r_7, r_{10}\}, 4\rangle$ and $mer_2 = \langle\{r_9, r_{10}\}, 2\rangle$. Also, let $BC = \{bc_1, bc_2\}$, where $bc_1 = \{p_4, p_3\}$ and $bc_2 = \{p_3, p_1\}$. The objective

is to find the set of minimum number of candidates from Y sufficient to cover the assignments in UA. Table 4 shows a possible UA' relation that covers all the assignments in UA.

5 COMPLEXITY ANALYSIS

This section presents a computational complexity analysis of the ORP problem. We establish a theorem to show that even a special case of the ORP problem without the MER and BoD constraints is NP-hard. The theorem establishes a polynomial time reduction of the Minimum Set Cover Problem to the current problem. The Minimum Set Cover Problem is one of Karp's 21 NP-complete problems. To initiate the proof, we first define the decision version of the ORP Problem and re-visit the optimization and decision versions of the Minimum Set Cover Problem.

Definition 7 (Decision Optimal Recruitment Problem (DORP)). Given a set U of employees, a set Y of candidates, and an integer t, can the vacancies created by the set $X \subseteq U$ be *filled* by a set $Y' \subseteq Y$ such that $|Y'| \leq t$, satisfying set MER of MER constraints, set BC of BoD constraints, and the user capability constraint UC in an ABAC system?

Definition 8 (Minimum Set Cover Problem (MSCP)). Given a universal set UV and a collection ST of subsets of UV, find a sub-collection with minimum number of subsets $st_1, st_2, \ldots, st_m$, where each $st_i \in ST$ and $st_1 \cup st_2 \cup \cdots \cup st_m = U$.

Definition 9 (Decision Version of MSC Problem (D-MSCP)). Given a universal set UV, a collection ST of subsets of U, and an integer v, does there exist a sub-collection of subsets $\{st_1, st_2, \ldots, st_m\} \subseteq ST$ that covers all the elements in U and $m \leq v$?

Theorem 5.1. *D-ORP is NP-hard.*

Proof. The reduction procedure takes a D-MSCP instance with sets $UV = \{uv_1, \ldots, uv_n\}$ and $ST = \{st_1, st_2, \ldots, st_m\}$ and a positive integer v to build an instance of D-MSCP and establish the fact that the D-ORP problem is at least as hard as D-MSCP. The reduction goes as follows:

- Users: A user u_i is added to the set U and X for each element uv_i in UV. Therefore, $U = X = \{u_1, \ldots, u_n\}$ and $|U| = |X| = |UV|$.
  For the set Y of new users, add an element un_j for each set in the collection ST. Therefore, $Y = \{un_1, \ldots, un_m\}$ and $|Y| = |ST|$.
- Attribute-value pair: An av pair av_i is unioned to the set AV for each element uv_i in UV. Therefore, $AV = \{av_1, \ldots, av_n\}$ and $|AV| = |UV|$.
- User Condition: A user condition is considered in the set $UCond$ for each av pair That is, $\forall (a, v) \in AV, a = v \in UCond$ and $UCond = \{uc_1, \ldots, uc_n\}$.
- Permission: For each element uv_i in UV, an element p_i is unioned to the set Per of permissions. Therefore, $Per = \{p_1, \ldots, p_n\}$ and $|Per| = |UV|$.
- User-to-attribute-value pair relation (UA): A diagonal matrix with all non-zero entries as 1 is set as the UA matrix. Formally, $\forall i, j$ if $i = j$, $UA(u_i, av_j) = 1$, else $UA(u_i, av_j) = 0$.
- Authorization matrix: Likewise, a diagonal matrix with all non-zero entries as 1 is set as the AM matrix. Formally, $AM \subseteq UCond \times Per$, where $\forall i, j$ if $i = j$, $AM(uc_i, p_j) = 1$, else $AM(uc_i, p_j) = 0$.
- The positive integer $t = v$.
- Constraints:
  - UC constraint: A new user $un_i \in Y$ that corresponds to the set $st_i \in ST$ is capable of handling an attribute-value pair av_j if the corresponding $uv_j \in st_i$. That is, the set of capable attribute-value pairs for a particular new user corresponds to a set in the collection ST. Formally, $\forall i, j$ if $uv_j \in st_i$, $UC(u_i, av_j) = 1$, else $UC(u_i, av_j) = 0$.

—BoD constraint: $BC = \phi$.
—MER constraint: $MER = \phi$.

Now, we can use the solution of the reduced DORP problem instance to get a solution to the corresponding DMSCP problem instance. We deduce this claim by proving the validity of the two cases as follows:

—If the reduction yields a "Yes" instance of the DORP problem, the correlated DMSCP instance is a "Yes" instance as well.
—If the reduction yields a "No" instance of the DORP problem, the correlated DMSCP instance is a "No" instance as well.

Let the reduction yield a "Yes" instance of the DMSCP problem. Therefore, the vacancies created by the set X of users can be filled in by at most t new users from the set Y. The set of t new users from the user-to-attribute-value pair assignment relation UA' obtained after solving the DORP instance can be used to identify the sets from the collection ST of the corresponding DMSCP instance, which covers all the elements of the set UV. The corresponding sets $\{st_1, \ldots, st_t\}$ form the sub-collection which covers the set UV. It is to be noted that because the attribute-value pairs correspond to the set UV of the DMSCP instance, and the UA relation is a diagonal matrix, the rows in the resultant UA' relation show the elements that have been covered by the corresponding sets in the sub-collection of the corresponding DMSCP instance. Since all the assignments from the UA matrix have been covered, all the elements of the set UV that correspond to the attribute-value pairs are implied to be covered. Thus, as the positive integer $v = t$, it is proven that the DMSCP instance from which the reduction is obtained is a "Yes" instance.

Let there be respective DMSCP and DORP problem instances α and β, where the proposed reduction is used to obtain β from α. Now, the second case can be posed as a contrapositive statement as follows: "if α is a 'Yes' instance, β will also be a 'Yes' instance." In that case, the sub-collection $\{st_1, \ldots, st_v\}$ can be used to construct the instance of UA' with $t = v$ new users. Each set st_i will correspond to a new user un_i and $(un_i, av_j) = 1$ in UA' if $uv_j \in st_i$ and $\forall st_x$, where $x = \{1, \ldots, i-1\}$, $uv_i \notin st_x$. With this construction, we can clearly obtain a UA' relation with t new users where every attribute-value pair will be assigned to at most one new user. Thus, the theorem is proved. □

6 SOLVING ORP

As the ORP problem is NP-hard, assuming $P \neq NP$, there is no polynomial time algorithm to solve it. Hence, we propose an algorithm based on a heuristic to obtain a close-to-optimal solution. In this section, we present the solution we have developed. We also establish an upper bound of the problem, which further is used to handle the BoD constraints.

To solve the ORP problem, first, the user-to-attribute-value pair relation UA is transformed into a user-to-user-condition relation UAC. For each user condition, if a user in the UA matrix is assigned to the corresponding attribute-value pairs, she is assigned to that particular user condition in UAC. It is to be noted that for a user, an attribute-value pair that is not covered by any user condition can be ignored. It is only a user condition that can give access of a permission to the user that is essential to perform a task. As the objective of this problem is to cover the functions of the replaced users, this reduction is apt. For instance, the UAC matrix for the UA matrix in Example 4.1 is given in Table 5. The alphabets of form a_i denote the enumerations of the corresponding assignments. This reduction is achievable in $O(|UCond| \times |UA|)$ time.

In order to avoid unnecessary complexities in the implementation, rather than considering a user-to-attribute-value pair relation, a user-to-user-condition relation UAC is considered as an

Table 5. Corresponding UAC Matrix for the UA Matrix in Example 4.1

	uc_1	uc_2	uc_3	uc_4	uc_5
u_1	0	a_1	0	a_2	0
u_2	a_3	0	a_4	0	0
u_3	0	a_5	a_6	0	0
u_4	0	a_7	0	a_8	0
u_5	0	0	0	0	a_9

ALGORITHM 1: Procedure to solve an ORP instance

1: **procedure** SolveORP(UAC, AM, BC, MER, UC)
2: Input: Set the User to User condition assignment relation UAC, Authorization matrix AM, and the set BC of BoD constraints,set MER of MER constraints, and relation UC of capability constraints.
3: F = BoDForest(UAC, AM, BC) ▷ Built BoD Forest
4: UAC' = AssignCandidates(F, MER, UC)
5: ▷ Assign candidates to the components of F
6: Output UAC'
7: **end procedure**

input to the ORP problem. It is assumed that this polynomial time reduction of UA to UAC for an organization is carried out before solving the ORP instance. Similarly, for the user capability constraint UC, the corresponding candidate-to-user-condition relation is considered.

Now, the problem is solved in two steps. First, a forest is constructed based on the AM matrix and the BoD constraints. Then the components of the forest are assigned with the candidates from set Y satisfying the MER constraint. Procedure *SolveORP* in Algorithm 1 makes the function calls for the procedure solving these two steps in Line no. 3 and Line no. 4, respectively.

Going into the details of the first step, we now introduce the construction named *BoD forest* shown in Algorithm 2, taking UAC, AM, and BC as inputs. We use this construction to build a forest with the assignments of UAC as nodes. The construction is described using the algorithm as follows. Lines 4 and 5 of the procedure *BoDForest* in Algorithm 2 iterate over the steps for each pair of permissions p_x, p_y in each BoD constraint. Line 6 iterates over the steps for each pair of assignments, one from the set having access to p_x and another from the set having access to p_y. Lines 8 and 10 add the edge joining the pair of assignments to the forest if its corresponding user conditions in UAC are not the same and the pair of assignments is not connected in the partially built forest. If an edge is formed in the forest, the corresponding pair of permissions (p_x, p_y) in Line 5 is recorded in Line 11. The procedure in Algorithm 3 is called in Line 17 to remove all the redundant edges from the partially built forest. For each edge (a, b) in the forest, the procedure *RemoveRedundantEdges* removes it if there exists a pair of nodes c, d with edges (a, c) and (b, d) caused due to the same pair of permissions (p_x, p_y) in Line 5. Line 18 finally returns the constructed BoD forest F.

The steps for constructing the BoD forest for Example 4.1 are illustrated in Figure 1. Figure 1(a) shows the BoD forest created before removing the redundant edges. The nodes represent assignments in the UAC matrix. An edge is formed between a pair of nodes a_i and a_j if they correspond to two different permissions (on the basis of the AM matrix) in a BoD constraint. However, no edge is formed for the BoD constraint bc_1 because both the permissions p_4, p_3 in bc_1 correspond to the

ALGORITHM 2: Algorithm to construct a *BoD Forest*

```
1: procedure BoDForest(UAC, AM, BC)
2:     Input: User to User condition assignment relation UAC, Authorization matrix AM, and set
       BC of BoD constraints
3:     Initialize an empty forest F
4:     for each BoD constraint bc_i ∈ BC do
5:         for each pair of permissions p_x, p_y ∈ bc_i do
6:             for each pair of assignments a ∈ assign(p_x) and b ∈ assign(p_y) do
7:                     ▷ assign(p_x) is the set of assignments that have access to p_x through AM
8:                 if Ucond(a) ≠ Ucond(b) and b is not reachable from a in F then
9:                             ▷ Ucond(a) is the corresponding user condition in the UAC matrix
10:                    E(F) = E(F) + (a, b)          ▷ E(F) is the set of edges in the BoD forest F
11:                    Per(a, b) = (p_x, p_y)
12:                ▷ Per(a, b) returns the tuple of permissions for which the edge (a, b) is formed
13:                end if
14:            end for
15:        end for
16:    end for
17:    F = RemoveRedundantEdges(F)
18:    Output F
19: end procedure
```

ALGORITHM 3: Procedure to remove the redundant edges from the BoD Forest

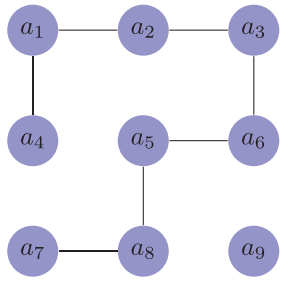
```
1: procedure RemoveRedundantEdges(F, AM, BC)
2:     Input: Set the BoD forest F, Authorization matrix AM, and the set BC of BoD constraints
3:     for each edge (a, b) ∈ E(F) do
4:         if ∃c, d where, Per(a, c) = Per(a, b) and Per(b, d) = Per(a, b) then
5:             Remove (a, b) from E(F)
6:         end if
7:     end for
8: end procedure
```

same set of user conditions, violating the condition check in Line 8 of Algorithm 2. All the edges formed correspond to the pair of permissions (p_3, p_1) of bc_2. Figure 1(b) shows the BoD forest after removing the redundant edges. For instance, the edge (a_1, a_2) is removed because there exists a pair of nodes a_4, a_3 with edges (a_1, a_4) and (a_2, a_3) caused due to the pair of permissions (p_3, p_1), satisfying the condition in Line 4 of Algorithm 2.

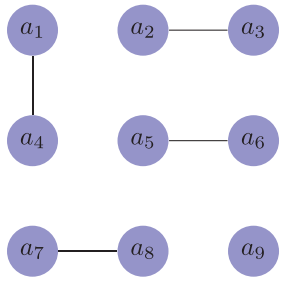
We use this forest to group the assignments of the *UAC* relation so as to enforce the BoD constraints. Before proceeding, we introduce the definition of a set of *Overlapping BoD constraints*.

Definition 10 (Overlapping BoD Constraints). A set *OB* of BoD constraints is said to be overlapping if $\forall bc_i \in OB$, $\exists bc_j \in OB$, where $bc_i \neq bc_j$, $bc_i \cap bc_j \neq \phi$ and $OB - bc_i$ is overlapping.

It is to be noted that if a user obtains access to any one of the permissions that belongs to a set $\{bc_1, \ldots, bc_n\}$ of overlapping BoD constraints, the user will have to be given access to all the permissions in the set $\bigcup_{i=1}^{n} bc_i$. Now, we establish an upper bound of the minimum set of candidates required to *fill* the vacancies in an ORP instance.



(a) STEP 1: Intermediate BoD forest before removing redundant edges.



(b) STEP 2: Final BoD forest after removing redundant edges from the forest in STEP 1.

Fig. 1. Two steps to create a BoD forest for the UAC matrix in Table 5 (Example 4.1). Here, all the edges in both the steps correspond to the tuple of permissions (p_3, p_4) from bc_2. No edges are formed due to bc_1.

Theorem 6.1. *$|Y'| \leq |assign(UAC)| - |E(F)|$, where Y' is the minimum set of candidates required to* fill *the vacancies in the ORP problem, where $assign(UAC)$ is the set of assignments in UAC, and $E(F)$ is the set of edges in the* BoD forest.

Proof. Each assignment in a component of the *BoD forest* has to be covered using a single candidate from the set Y in the ORP problem to enforce the set of BoD constraints. This is because, from the construction in Algorithm 2, the BoD constraints for which the corresponding edges are formed in a particular component of the *BoD forest* F are overlapping (as defined in Definition 10). The total number of components in the *BoD forest* F is $|V(F)| - |E(F)|$, where $V(F)$ is the set of vertices in the forest. The set of assignments in each component of F has to be assigned to a single user to enforce the BoD constraints. Therefore, at least $|assign(UAC)| - |E(F)|$ number of users are required to cover all the assignments in an ORP problem instance. □

Therefore, the set of assignments grouped by each component of the *BoD forest* now has to be covered using a single new user so as to enforce the BoD constraints. This assignment is carried out in the second step based on a heuristic in which for each component, the smallest candidate that satisfies the given MER and UC constraints is assigned to it. The candidates are sorted in decreasing order of the number components that they are capable of getting assigned to. The procedure *AssignCandidates* in Algorithm 4 describes the assignment of the candidates (new users) to the components of the computed *BoD forest*. Line 4 computes the connected components of the forest F. Line 6 sorts the candidates Y in decreasing order of the number components that they are capable of getting assigned to. Line 7 sorts the components of F in decreasing order of the number of MER constraints that it has to abide with. It is to be noted that not all MER constraints have to be verified when an assignment takes place for a component, only the ones with the rules that the corresponding user conditions of a component is part of has to be satisfied. Now, for each component the smallest candidate in the order that can be assigned to it is computed from Lines 8 to 32. Lines 10 to 14 find whether the current candidate is capable of handling the current component. MER constraints are verified in Lines 15 to 21. Components that have any common corresponding user conditions cannot be assigned with the same candidate, and the same is taken care of in Lines 22 to 26. Lines 27 to 30 assign the corresponding user conditions of the current component to the current candidate if all the conditions are satisfied. The generated UAC' is returned in Line 33.

ALGORITHM 4: Procedure to assign candidates to the connected components of F

```
1: procedure AssignCandidates(F, MER, UC, Y)
2:     Input: Set Bod Forest F, MER constraints MER,
   user capability constraints UC of candidates Y
3:     Initialize an empty relation UAC'
4:     Components_F = GetComponents(F)
5:                                   ▷ Getting connected components of the forest F
6:     CandOrder ← Y in decreasing order of the no. of capable components
7:     CompOrder ← Components_F in decreasing order of the no. of affecting MER constraints
8:     for comp ∈ CompOrder do
9:         for cand ∈ CandOrder do
10:            if cand capable of assigning comp then
11:                capable ← True
12:            else
13:                capable ← False
14:            end if
15:            if cand is assigned to comp then
16:                if no MER is violated then
17:                    satisfy_MER ← True
18:                else
19:                    satisfy_MER ← False
20:                end if
21:            end if
22:            if cand is assigned to a conflicting component w.r.t comp then
23:                conflict ← True
24:            else
25:                conflict ← False
26:            end if
27:            if capable and satisfy_MER and not conflict then
28:                UConds = Ucond(comp)          ▷ corresponding user conditions
29:                UAC'[cand, UConds] = 1
30:            end if
31:        end for
32:    end for
33:    Output UAC'
34: end procedure
```

The individual procedures in Algorithms 2 and 4 have the following worst-case time complexities.

- *BoDForest*: $O(|BC| \times |Per|^2 \times |UAC|^2)$.
- *AssignCandidates*: $O(|UAC| \times |Y| \times |MER|)$, here $|UAC|$ denotes the total number of assignments in the UAC matrix, respectively.

Thus, the overall running time complexity of the procedure *SolveORP* in Algorithm 1 happens to be $O(|UAC| \times (|BC| \times |Per|^2 \times |UAC| + |Y| \times |MER|))$.

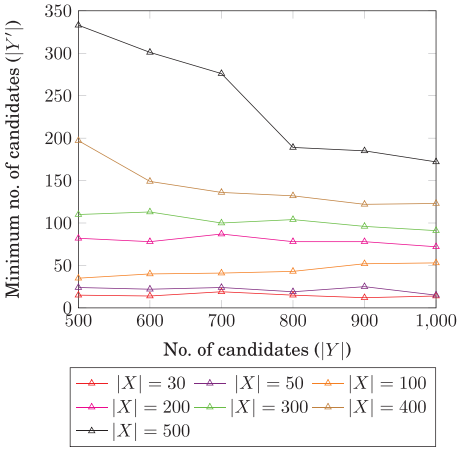


Fig. 2. Variation of minimum number of candidates with respect to $|Y|$ for different values of $|X|$.

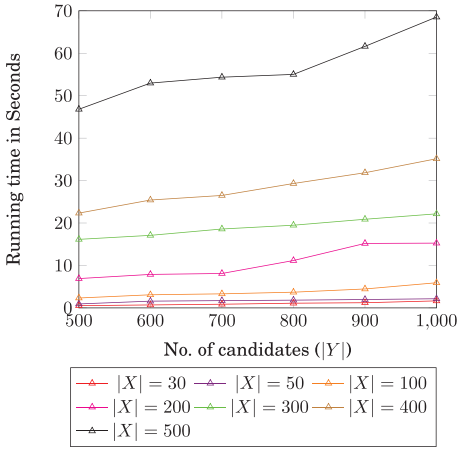


Fig. 3. Variation of running time with respect to $|Y|$ for different values of $|X|$.

7 EXPERIMENTAL EVALUATION

Python has been used to implement the algorithms proposed in Section 6. The experimentation to test the implementation was carried out on an Intel Core i5 (processor speed of 1.7 GHz) machine with 8 GB of RAM running Linux. Study of the minimum number of candidates and time required to execute the solution for changing the number of BoD constraints, number of MER constraints, number of new and retiring employees, number of user conditions, permissions, and rules for an ABAC system is carried out. Twenty different runs on randomly generated synthetic data were executed for each combination of parameters and medians over the different runs are reported.

Figure 2 presents the variation in the minimum candidate requirement $|Y'|$ to fill the vacancies of the number of employees $|X|$ varying from 30 to 500 for the size of set Y of candidates from 500 to 1,000. The other parameters are kept fixed as follows: number of user conditions as 20, number of permissions as 20, number of rules as 50, and number of BoD and MER constraints as 10 each. Figure 3 presents the corresponding variation in the running time. It is observed that generally for a lower value of $|X|$, there is not much variation in $|Y'|$ with the increase in the number of candidates. The number of candidates varies from 500 to 1,000, which is quite high compared to $|X|$; hence, this variation does not affect the choice of candidates for assignment in Algorithm 4. However, for bigger values of $|X|$, the minimum number of candidates required $|Y'|$ decreases with the increase in the number candidates. This variation occurs because with a higher number of candidates that have varied capabilities due to the user capability constraint, a chance of getting a better candidate that can be assigned to a higher number of BoD forest components increases. This variation is more evident in the plot for $|X| = 300$ or more as the difference between $|X|$ and $|Y|$ decreases. Further, with an increase in the number of retiring employees, the minimum candidates requirement $|Y'|$ increases because of the increase in the number of allocations in the UAC matrix.

From Figure 3, it is noticed that variation in the running time increases with the increase in the number of retiring employees $|X|$ and the number of candidates $|Y|$. With an increase in X, the number of allocations in the UAC matrix increases, increasing the execution time of the procedure in Algorithms 2 and 4. Also, with more $|Y|$, the for loop in Lines 8 to 15 of Algorithm 4 have to run for more iterations, thus increasing the running time.

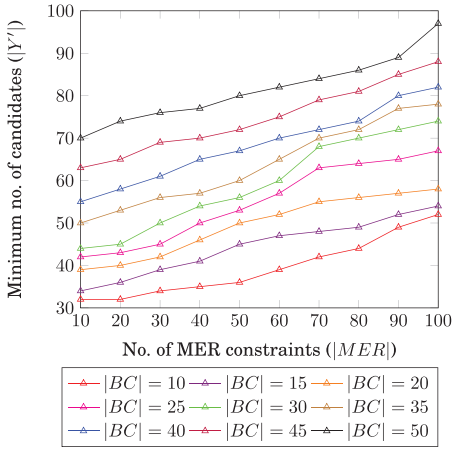


Fig. 4. Variation of minimum number of candidates with respect to |*MER*| for different values of |*BC*|.

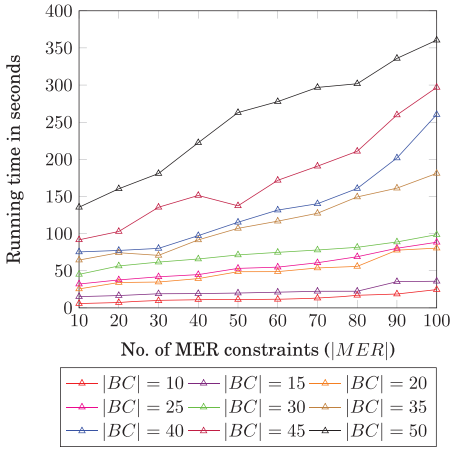


Fig. 5. Variation of running time with respect to |*MER*| for different values of |*BC*|.

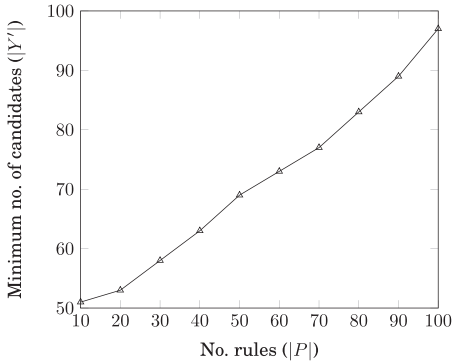


Fig. 6. Variation of minimum number of candidates with respect to |*P*|.

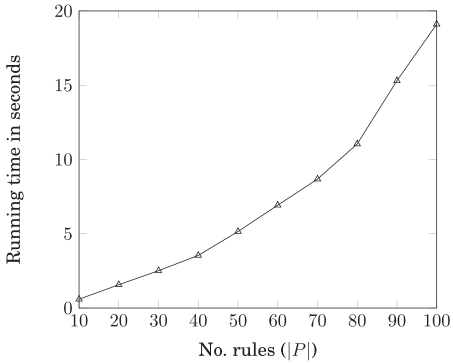


Fig. 7. Variation of running time with respect to |*P*|.

Figure 4 demonstrates the change in the minimum candidates requirement $|Y'|$ with altering the number of BoD constraints from 10 to 50 and MER constraints from 10 to 100. The other parameters are kept fixed as follows: $|X|$ as 100, $|Y|$ as 500, number of user conditions as 20, number of permissions as 20, and number of rules as 50. Figure 5 presents the corresponding variation in the running time. It is noticed that, due to the increase in restrictions imposed by the increasing number of BoD and MER constraints, $|Y'|$ increases. The execution time is also observed to increase because of the increase in the number of iterations required to build the BoD forest in Algorithm 2 and time required for verification of the MER constraints in Line 10 of Algorithm 4.

Figure 6 demonstrates the variation in $|Y'|$ with changing the number of rules from 10 to 100, keeping X as 100, Y as 500, number of permissions as 10, number of user conditions as 10, number of MER constraints as 10, and number of BoD constraints as 10. Figure 7 presents the corresponding change in the running time. It is observed that $|Y'|$ increases with the increasing number of rules. This is because, with an increase in the number of allocations in the AM matrix, the number of user conditions assigned to the permission sets of the overlapping BoD constraints increases, thus increasing the number of allocations in the randomly generated UAC matrix, which satisfies the

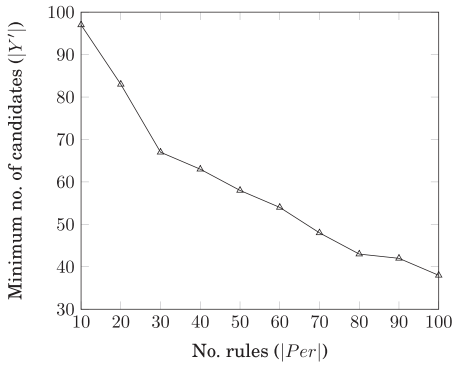


Fig. 8. Variation of minimum number of candidates with respect to |*Per*|.

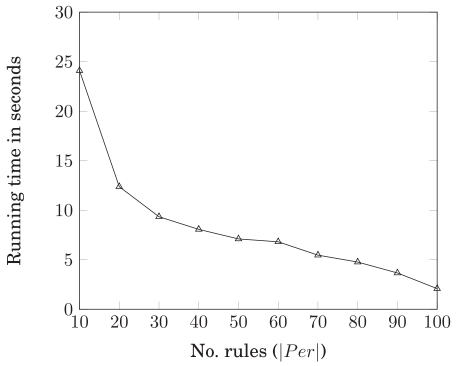


Fig. 9. Variation of running time with respect to |*Per*|.

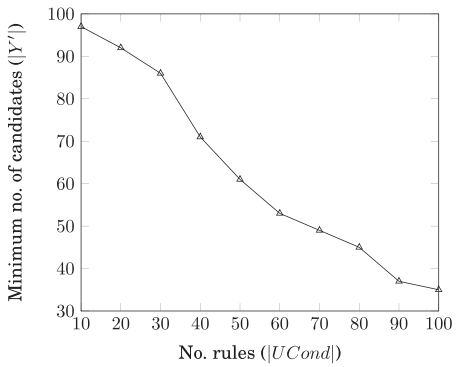


Fig. 10. Variation of minimum number of candidates with respect to |*UCond*|.

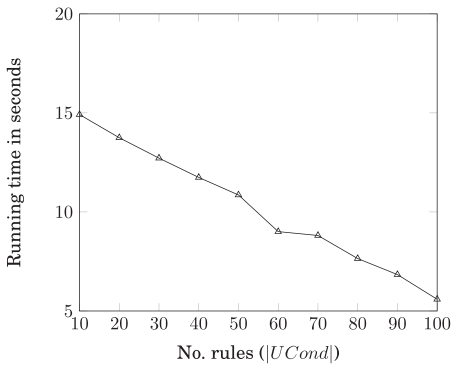


Fig. 11. Variation of running time with respect to |*UCond*|.

constraints. It is also noticed that the running time increases because of the increasing number of allocations in the *UAC* matrix.

Figure 8 demonstrates the change in $|Y'|$ with changing the number of permissions from 10 to 100, keeping X as 100, Y as 500, number of rules as 100, number of user conditions as 10, number of MER constraints as 10, and number of BoD constraints as 10. Figure 9 shows the corresponding change in the running time. It is observed that $|Y'|$ decreases with the increasing number of permissions. This is due to the decrease in the number of user conditions assigned to the permission sets of the overlapping BoD constraints, which causes the number of allocations in the *UAC* matrix to decrease during data generation. It is to be noted that, due to the fixed number of allocations (rules) in the *AM* matrix, this decrease is observed. The running time decreases for the same reason.

Figure 10 shows the change in $|Y'|$ with changing the number of user conditions from 10 to 100, keeping X as 100, Y as 500, number of rules as 100, number of permissions as 10, number of MER constraints as 10, and number of BoD constraints as 10. Figure 11 shows the corresponding change in the running time. The minimum candidate requirement decreases with the increasing number of user conditions due to the same reason described for the experiment varying the number of permissions. Because the number of allocations (rules) in the *AM* matrix is fixed, the number of user conditions assigned to the permission sets of the overlapping BoD constraints decreases, which causes the number of allocations in the *UAC* matrix to decrease during data generation.

Thus, this trend is observed. The execution time also decreases due to a decrease in the number of allocations in the UAC matrix.

8 RELATED WORK

Extensive research has been carried out in the field of access control and workforce optimization that also has some relation with the current work. However, none of them deal with the problem of optimal recruitment in the presence of security constraints. This section discusses such works and shows the essential differences with the current study.

Study and implementation of a variety of access control models for enforcing secure access of resources in business information systems have been carried out [Harrison et al. 1976; Miller and Baldwin 1990; Snyder 1977; Bell and Lapadula 1976; Sandhu et al. 1996]. Although a fairly large number of enterprises still use RBAC [Zhao et al. 2015; Fuchs et al. 2011; Zhu and Zhou 2008], since it is not expansible and has other drawbacks, ABAC has recently proliferated [Hu et al. 2014]. Based on ABAC, a variety of access control models for different environments have been proposed over recent years. A framework designed by Qi et al. [2015] dealt with distributed access control architecture combining role and attribute. An ABAC-based location-aware model that can be of use in social networking web applications to detect security infringements has been proposed in Hsu and Ray [2016]. Jin et al. proposed a role-centric ABAC model named RABAC [Jin et al. 2012a]. The property of attribute inheritance was introduced in ABAC in a hierarchical framework by Servos and Osborn [2014].

A scheme to define the access control structures using an access tree made up of logic expressions over attributes was designed by Chatterjee et al. [2014]. A Label-Based Access Control model for policy enumeration was proposed by Biswas et al. [2016]. Riad et al. discussed an ABAC model for the cloud environment [Riad et al. 2015]. A unified ABAC model for configuring older versions of access control models including RBAC was proposed in Jin et al. [2012b], thus formally establishing the feasibility of posing older access control models as special cases of ABAC. This work also establishes that solutions for other access control models are intrinsically developed if ABAC is considered. The publication by NIST [Hu et al. 2014] comprehensively defines the ABAC model. Further research done to achieve more efficient operationalization of ABAC in different varieties of business information systems can be found in Hüffmeyer and Schreier [2016], Benkaouz et al. [2016], Xu and Stoller [2015], Ferraiolo et al. [2016], and Servos and Osborn [2017].

Access control models use different kinds of constraints to enforce various security policies in information systems. Two of the most important constraints used are SoD and BoD constraints. SoD requires that no user carries out multiple conflicting tasks. It was introduced by Clark and Wilson in order to control errors and fraud in information systems [Clark and Wilson 1987]. A general definition of SoD has been considered, taking a cue from its traditional definition. The considered SoD constraint makes it necessary to involve a minimum of a specified number of subjects for completion of a task that involves a specific set of permissions [Li et al. 2007]. Jha et al. [2018] showed that verifying such constraints in an ABAC system is a hard problem and thus proposed a form of constraint named MER to which the system's SoD constraints can be converted. Verification of MER constraints has been proved to be in polynomial time. In our work, we use the MER constraints to deal with the complexities that SoD constraints can pose to the current problem.

A BoD constraint makes sure that a subject performs either all or none of the tasks in a set that has been specified, and thus a relation is defined between them [Strembeck and Mendling 2010]. Alternatively, it can be said that the same subject has to perform the set of "bound tasks" in order to complete a job. The BoD constraints we dealt with in this work compel a subject who requires any one permission in a set to get assigned to all the permissions in that set.

Other than BoD and SoD constraints, the user capability constraint is dealt with in this article. We used this constraint in our previous work [Roy et al. 2016]; however, the definition of this constraint differed qualitatively. The user capabilities in Roy et al. [2016] were defined using roles in the given RBAC system. Though the term "capabilities" is not alien to access control literature, its contextual use has been completely different. A capability-based access control system for a distributed environment was proposed by Gusmerolia et al. [2013]. In their work, the subjects have to present their "authorization capabilities" (the term used for permissions) to the service provider to perform the requested operation on the specified resources. Similarly, access privileges in the possession of a user are called "capabilities" in the capability-based computer system proposed in Levy [2014]. In this work, we use the term "capability" to refer to the abilities that an employee has gained through training and experience before responsibilities are assigned.

Some of our previous works deal with the problem of workforce optimization in systems embracing access control models [Roy et al. 2012, 2014, 2015, 2016]. While algorithms to determine the minimum number of employees required to impose SMER constraints in RBAC were proposed in Roy et al. [2012] and Roy et al. [2014], it has considerable differences with the current problem. First, in our previous work we considered the situation where an RBAC system is being deployed *ab initio*, whereas the current work deals with a change in employee assignment scenario in an already running system. Second, in our previous work, new assignments were introduced to deploy the system with a minimum number of users, whereas in the ORP problem being addressed in the current article, we cover a section of the old assignments with a fresh set of employees. Finally, in the current work we consider an ABAC system, while our previous work considered an RBAC system. With ABAC emerging as a new standard for access control in commercial organization, it is important to consider its implication in workforce optimization.

In Roy et al. [2015], the problem of computing the smallest number of users needed to execute a given set of jobs was considered in the presence of mutually exclusive task and cardinality constraints. However, no specific access control model was considered in this work. It dealt with a problem of optimal deployment of a system *ab initio* as in Roy et al. [2012] and Roy et al. [2014]. The problem of maximizing workforce utilization in an RBAC system with relevant security constraints was studied in Roy et al. [2016]. Again, it considers a scenario where the system is getting set up for the first time and the number of allocations for employees to the roles was maximized. On the other hand, in the current article, we do not attempt to maximize the assignments as they are fixed; rather, we minimize the number of replaced employees. Thus, none of the above-mentioned articles considered the problem of optimal recruitment of the replacements of the retiring employees, and most importantly the ABAC system was not taken into account.

In Roy et al. [2019], as in the current work, an ABAC environment was considered and a solution to an important workforce optimization problem (Employee Replacement problem (ERP)) in the presence of security constraints was proposed. Although both problems deal with a situation where a set of employees is leaving the organization and they are to be compensated with new recruits, the basic objectives of the two problems differ in a fundamental way, requiring unique algorithms for solving them. While in Roy et al. [2019] the ERP problem only verifies whether the new set of employees can actually compensate the functions of the set of relieved employees, the ORP problem as studied in the current article deals with minimizing the compensating set of employees by identifying the best set of recruits from the given candidates. It thereby solves a lot more critical problems in the field of workforce optimization. As the structure of the problems suggests, the current problem is an optimization problem, whereas ERP is a decision problem. Another important aspect of the current work that was not addressed in Roy et al. [2019] is that it considers MER constraints in place of the raw SoD constraints, thus giving it a capability to handle

environmental conditions, one of the unique features of ABAC that sets it apart from RBAC, as discussed in Section 3.

In Bertino et al. [1999], Wang and Li [2010], and Crampton et al. [2013], some previous work pertaining to workflow management that deals with assigning users to their authorized rights over resources can be found. Crampton [2005] was the first to analyze the computational complexity of the problem to decide if a user-to-task-allocation assignment satisfies an authorization policy for a given workflow. All the above-mentioned work verify whether an employee-to-authorization-privilege assignment can be configured in a way that satisfies the specified security conditions, thus addressing an entirely different problem. Furthermore, because an ABAC model was not considered in place, most of them were decision problems and the hierarchical arrangement of the tasks in these works, the current problem in different in its entirety.

The work by Basin et al. [2012] appears most similar to the current work. The authors deal with the problem of optimization of the transition of an existing users-to-tasks-assignment relation that may not satisfy all the given security constraints to a fresh assignment relation that satisfies all the constraints while allowing execution of the workflow successfully. The cost of transition, including costs related to administrative or maintenance, and risk are the factors considered for optimization. Even the current problem deals with a transition of an existing assignment relation to a new one; however, in Basin's problem the set of users remained the same, whereas we covered the previous assignments using a completely different set of users. While the assignments in Basin's problem may change to satisfy the security constraints, in ORP the same assignments are covered by the new set of users. The initial assignment relation in ORP always satisfies the security constraints, which might not be the case in the Basin et al. problem. The notion of optimality differs substantially, as we deal with the optimization of the number of employees instead of cost of transformation. While k-n SoD constraints of any size can be handled in our work, Basin et al. can only deal with a 2-2 form of SoD constraints. Finally, our work considers the ABAC model and is independent of any particular workflow execution.

The issue of downsizing is dealt with in some articles in the area of management research [Schmenner and Lackey 1994; Ryan and Macky 1998; Franco 2013]. The authors have broadly classified the strategies of downsizing used by different organizations as workforce reduction, work redesign, and systemic strategy. The workforce reduction approach is being considered as the most preferred downsizing strategy [Schmenner and Lackey 1994]. Considering different tactics for workforce reduction, the approach considered in this article is quite close to succession planning. However, nowhere in the literature has the development of an algorithmic or mathematical tool to facilitate downsizing been addressed.

In Table 6, we provide a summary of how the current contribution differs from all of the existing work in the literature. In the columns, we enumerate various factors that can be used to compare different approaches. These include the type of access control model considered, nature of the problem, how the existing set of employees are affected, consideration of new employees, types of security constraints handled, and optimization of new recruits. The rows list all of the relevant existing work in chronological order. A ✓ mark denotes that the corresponding existing work in the row is similar to the current work in terms of the factor mentioned in the column (e.g., Roy et al. [2019] and the current work both consider the ABAC model). Similarly, a × mark implies they are dissimilar (e.g., Roy et al. [2014, 2015, 2016] all considered RBAC, while the current work considers ABAC). A – mark means that the work does not consider this parameter (e.g., Schmenner and Lackey [1994] and Ryan and Macky [1998] do not consider access control at all). From the comparison chart, it is clear that this article deals with a problem that is unique and the approach is novel. We would like to emphasize that while our prior work also uses optimization methods, the

Table 6. Comparison of ORP (This Work) to Existing Work in the Literature

	Type of access control	Nature of the problem	Affect existing employees	Involve new employees	Security constraints	Optimize recruits
Schmenner and Lackey [1994]	–	–	✓	✓	–	×
Ryan and Macky [1998]	–	–	✓	–	–	–
Bertino et al. [1999]	×	×	✓	×	×	×
Wang and Li [2010]	×	×	✓	×	×	×
Basin et al. [2012]	×	✓	✓	×	×	×
Roy et al. [2012]	×	✓	×	✓	×	×
Crampton et al. [2013]	×	×	✓	×	×	×
Franco [2013]	–	–	✓	×	–	×
Roy et al. [2014]	×	✓	×	✓	×	×
Roy et al. [2015]	×	✓	×	✓	×	×
Roy et al. [2016]	×	✓	×	✓	×	×
Roy et al. [2019]	✓	×	✓	✓	×	×

current work continues the same line of research and answers an important challenge significant to management information systems.

9 CONCLUSION AND FUTURE WORK

In this article, a scenario was considered where a section of human resources is stepping down from an organization due to retirement or downsizing attempts and recruitment of new capable employees has to be done from the set of aspiring candidates. Separation of duty constraints, which were represented as MER constraints for efficiency, BoD constraints, and capability constraints, along with an ABAC environment was considered as a means to enforce security policies. The problem has been formally posed as the Optimal Recruitment Problem and proved to be NP-hard. To come by a solution, a greedy heuristic was proposed and an upper bound to the problem was derived. We have experimentally verified our implementation and presented the results, which show the efficiency of the solution.

Due to its consideration of ABAC, an important contribution of the current work is that it can also handle recruitment optimization for other access control mechanisms like DAC, MAC, and RBAC. This is due to the ability of ABAC to specify the policies of all such models. Since the proposed approach uses a heuristic for solving the problem, the results are obtained quite efficiently. However, a flip side is that an exact solution may not be obtained. Given the nature of application for this work, an approximate solution can be fine-tuned through human intervention, and hence, this trade-off is worth examining. Also, the algorithm is designed in such a way that security constraints would never get violated, thereby ensuring safety. Although we handle most of the important security constraints that an access control system deals with in a real-world scenario, a

limitation of this work is that any constraint that is not expressible in an SoD, BoD, MER, or user capability form cannot be modeled with the ORP problem. Developing a generalized framework encompassing all possible constraints would be an interesting future task in this context. Another limitation of the proposed approach is that it does not take into consideration any historical data of the candidates while selecting the recruits.

Taking into consideration the strengths and limitations of this work, in the future, research in the following directions would be interesting to pursue:

- —A unified framework can be developed to represent any possible security or other business constraint. This would help in generalizing the current work and enhance its applicability.
- —Historical data can be used to segment the candidates on the basis of different qualitative attributes, with an optimization strategy applied on top of it. Combining descriptive, predictive, and prescriptive analytics to solve the problem of recruitment optimization would also be a prospective direction for research.
- —The current approach assumes that the vacancies created in the organization are due to a set of employees leaving the organization. However, vacancies can also be created when the organization is planning to expand. Expansion may lead to the creation of a new set of objects and attributes, which have to be handled.
- —Extension of this work to handle reorganization of employees across sister organizations, coordinated and managed in a multi-organizational setup, will also be an equally interesting aspect for investigation.

REFERENCES

Axiomatics. 2016. The Evolution of RBAC Models to Next-Generation ABAC. *Axiomatics (White Paper)*. Retrieved from www.axiomatics.com/resources/rbac-to-abac/.

D. Basin, S. J. Burri, and G. Karjoth. 2012. Optimal workflow-aware authorizations. In *Proceedings of the ACM Symposium on Access Control Models and Technologies*. 93–102.

D. E. Bell and L. J. Lapadula. 1976. Secure computer system: Unified exposition and multics interpretation. Electronic Systems Division, Air Force Systems Command, Hanscom Field, Bedford, MA.

Y. Benkaouz, M. Erradi, and B. Freisleben. 2016. Work in progress: K-nearest neighbors techniques for ABAC policies clustering. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*. 72–75.

J. V. Bergh, J. Beliën, P. D. Bruecker, E. Demeulemeester, and L. D. Boeck. 2013. Personnel scheduling: A literature review. *European Journal of Operational Research* 226, 3 (2013), 367–385.

E. Bertino, E. Ferrari, and V. Atluri. 1999. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security* 2, 1 (1999), 65–104.

P. Biswas, R. Sandhu, and R. Krishnan. 2016. Label-based access control: An ABAC model with enumerated authorization policy. In *Proceedings of ACM Workshop on Attribute Based Access Control*. 1–12.

S. Chatterjee, A. K. Gupta, V. K. Mahor, and T. Sarmah. 2014. An efficient fine grained access control scheme based on attributes for enterprise class applications. In *Proceedings of International Conference on Signal Propagation and Computer Technology*. 273–278.

D. D. Clark and D. R. Wilson. 1987. A comparison of commercial and military computer security policies. In *Proceedings of the IEEE Symposium on Security and Privacy*. 184–194.

J. Crampton. 2005. A reference monitor for workflow systems with constrained task execution. In *Proceedings of the 10th ACM Symposium on Access Control Models and Technologies*. 38–47.

J. Crampton, G. Gutin, and A. Yeo. 2013. On the parameterized complexity and kernelization of the workflow satisfiability problem. *ACM Transactions on Information and System Security* 16, 1 (2013), 4:1–4:31.

J. Dumay and J. Rooney. 2011. Dealing with an ageing workforce: Current and future implications. *Journal of Human Resource Costing & Accounting* 15, 3 (2011), 174–195.

D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu. 2016. Extensible access control markup language (XACML) and next generation access control (NGAC). In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*. 13–24.

G. Franco. 2013. Workforce downsizing: Strategies, archetypes, approaches and tactics. *Journal of Management Research* 13, 2 (2013), 67–76.

F. Frisch. 2014. A technical view of the business case for Attribute Based Access Control (ABAC). *Axiomatics (Case Study)*. Retrieved from www.axiomatics.com/blog/a-technical-view-of-the-business-case-for-attribute-based-access-control-abac-part-2/.

L. Fuchs, G. Pernul, and R. Sandhu. 2011. Roles in information security–A survey and classification of the research area. *Computers & Security* 30, 8 (2011), 748–769.

H. Goldbeck. 2019. The 2020 Recession: What It Means for Recruiters. *Recruiter*. Retrieved from www.recruiter.com/i/the-2020-recession-what-it-means-for-recruiters/.

B. Goren. 2008. Five Steps to Optimizing Human Capital. *SAS Institute Inc*. Retrieved from www.sas.com/resources/asset/5stepstooptimization.pdf.

S. Gusmerolia, S. Piccionea, and D. Rotondi. 2013. A capability-based security approach to manage access control in the Internet of Things. *Mathematical and Computer Modelling* 58, 5–6 (2013), 1189–1205.

M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. 1976. Protection in operating systems. *Communications of the ACM* 19, 8 (1976), 461–471.

A. C. Hsu and I. Ray. 2016. Specification and enforcement of location-aware attribute-based access control for online social networks. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*. 25–34.

V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone. 2014. Guide to attribute based access control (ABAC) definition and considerations. NIST Special Publication.

M. Hüffmeyer and U. Schreier. 2016. RestACL - An access control language for RESTful services. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, 58–67.

E. V. Imhoff and K. Henkens. 1989. The budgetary dilemmas of an ageing workforce: A scenario study of the public sector in the Netherlands. *European Journal of Population* 14 (1989), 39–59.

S. Jha, S. Sural, V. Atluri, and J. Vaidya. 2018. Specification and verification of separation of duty constraints in attribute-based access control. *IEEE Transactions on Information Forensics and Security* 13, 4 (2018), 897–911.

X. Jin, R. Sandhu, and R. Krishnan. 2012a. RABAC: Role-centric attribute-based access control. In *Proceedings of International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, 84–96.

X. Jin, R. Sandhu, and R. Krishnan. 2012b. A unified attribute-based access control model covering DAC, MAC and RBAC. In *Proceedings of IFIP Annual Conference on Data and Applications Security and Privacy*. 41–55.

H. M. Levy. 2014. *Capability-Based Computer Systems*. Digital Press.

N. Li, M. V. Tripunitara, and Z. Bizri. 2007. On mutually exclusive roles and separation-of-duty. *ACM Transactions on Information and System Security* 10, 2 (2007), 1–36.

D. V. Miller and R. W. Baldwin. 1990. Access control by boolean expression evaluation. In *Proceedings of the Annual Computer Security Applications Conference*. 131–139.

H. Qi, H. Ma, J. Li, and X. Di. 2015. Access control model based on role and attribute and its applications on space-ground integration networks. In *Proceedings of International Conference on Computer Science and Network Technology*. 1118–1122.

K. Riad, Z. Yan, H. Hu, and G.-J. Ahn. 2015. AR-ABAC: A new attribute based access control model supporting attribute-rules for cloud computing. In *Proceedings of IEEE Conference on Collaboration and Internet Computing*. 28–35.

A. Roy, S. Sural, and A. K. Majumdar. 2012. Minimum user requirement in role based access control with separation of duty constraints. In *12th International Conference on Intelligent Systems Design and Applications*, 386–391.

A. Roy, S. Sural, and A. K. Majumdar. 2014. Impact of multiple t-t SMER constraints on minimum user requirement in RBAC. In *Proceedings of the International Conference on Information Systems Security*. 109–128.

A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri. 2015. Minimizing organizational user requirement while meeting security constraints. *ACM Transactions on Management Information Systems* 6, 3 (2015), 12:1–12:25.

A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri. 2016. On optimal employee assignment in constrained role-based access control systems. *ACM Transactions on Management Information Systems* 7, 4 (2016), 10:1–10:24.

A. Roy, S. Sural, A. K. Majumdar, J. Vaidya, and V. Atluri. 2019. Enabling workforce optimization in constrained attribute based access control systems. *IEEE Transactions on Emerging Topics in Computing (Early access)* (2019). 1–1. DOI: https://doi.org/10.1109/TETC.2019.2944787

L. Ryan and K. A. Macky. 1998. Downsizing organizations: Uses, outcomes and strategies. *Asia Pacific Journal of Human Resources* 36, 2 (1998), 29–45.

J. H. Saltzer and M. D. Schroeder. 1975. The protection of information in computer systems. In *Proceedings of the IEEE*. 1278–1308.

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

S. Schefer, M. Strembeck, and J. Mendling. 2012. Checking satisfiability aspects of binding constraints in a business process context. In *International Conference on Business Process Management*. 465–470.

R. Schmenner and C. Lackey. 1994. “Slash and burn” doesn’t kill weeds: Other ways to downsize the manufacturing organization. *Business Horizons* 37, 4 (1994), 80–87.

D. Servos and S. L. Osborn. 2014. HGABAC: Towards a formal model of hierarchical attribute-based access control. In *Proceedings of International Symposium on Foundations and Practice of Security*. 187–204.

D. Servos and S. L. Osborn. 2017. Current research and open problems in attribute-based access control. *Computing Surveys* 49, 4 (2017), 65:1–65:45.

B. Smith and T. Rutigliano. 2001. Reducing staff the right way, smart strategies for approaching a manager’s toughest challenge: Choosing who stays, and who goes. *Gallup Business Journal*. Retrieved from news.gallup.com/businessjournal/334/reducing-staff-right-way.aspx.

L. Snyder. 1977. On the synthesis and analysis of protection systems. In *Proceedings of the ACM Symposium on Operating Systems Principles*. 141–150.

M. Starich. 2019. Recruiters Preparing for the Recession of 2020. *Recruiting Daily*. Retrieved from recruitingdaily.com/preparing-for-the-recession-of-2020/.

M. Strembeck and J. Mendling. 2010. Generic algorithms for consistency checking of mutual-exclusion and binding constraints in a business process context. In *Proceedings of OTM Confederated International Conferences “On the Move to Meaningful Internet Systems”*. 204–221.

M. Strembeck and J. Mendling. 2011. Modeling process-related RBAC models with extended UML activity models. *Information and Software Technology* 53, 2 (2011), 456–483.

K. Tan, J. Crampton, and C. A. Gunter. 2004. The consistency of task-based authorization constraints in workflow systems. In *IEEE Computer Security Foundations Workshop*. 1–15.

J. Tolan. [n.d.]. 4 Ways The Next Recession Will Change Recruitment. *Spark Hire*. Retrieved from hr.sparkhire.com/tips-for-recruiters/4-ways-the-next-recession-will-change-recruitment/.

Q. Wang and N. Li. 2010. Satisfiability and resiliency in workflow authorization systems. *ACM Transactions on Information and System Security* 13, 4 (2010), 40:1–40:35.

Z. Xu and S. D. Stoller. 2015. Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing* 4, 5 (2015), 533–545.

X. Zhao, C. Liu, S. Yongchareon, M. Kowalkiewicz, and W. Sadiq. 2015. Role-based process view derivation and composition. *ACM Transactions on Management Information Systems* 6, 2 (2015), 7:1–7:24.

H. Zhu and M. Zhou. 2008. Roles in information systems: A survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews* 38, 3 (2008), 377–396.

Received November 2019; revised April 2020; accepted May 2020