Received March 2, 2021, accepted April 2, 2021, date of publication April 12, 2021, date of current version April 19, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3072635

An Improved Genetic Algorithm for Safety and Availability Checking in Cyber-Physical Systems

ZHENG WANG[1,2], **YANAN JIN**[3], **SHASHA YANG**[4], **JIANMIN HAN**[5], **AND JIANFENG LU**[5]

[1]Faculty of Science and Engineering, The Open University of China, Beijing 100039, China
[2]Department of Computer Science, Zhejiang Radio & Television University Haiyan College, Jiaxing 314300, China
[3]School of Information Management and Statistics, Hubei University of Economics, Wuhan 430205, China
[4]Xingzhi College, Zhejiang Normal University, Jinhua 321004, China
[5]Department of Computer Science and Engineering, Zhejiang Normal University, Jinhua 321004, China

Corresponding author: Yanan Jin (jinyanan@yhcrt.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 62072411 and Grant 61872323, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LR21F020001, in part by the Zhejiang Provincial Department of Education under Grant Y202043497, and in part by the Social Development Project of Zhejiang Provincial Public Technology Research under Grant 2017C33054.

ABSTRACT Cross-IoT infrastructure access frequently occurs when performing tasks in a distributed computing infrastructure of a cyber-physical system (CPS). The access control technology that ensure secure access cross-IoT infrastructure usually automatically establish relationships between user-attribute/role-permission. How to efficiently determine whether an automatic authorization access control state satisfies the safety and availability requirements of a system is a huge challenge. Existing work often focuses on a single aspect of safety or availability, while ignoring the differences between permissions and the differences between users. In this paper, we first propose a fine-grained personalization policy that takes into account the specificity of permissions/users and describes the safety, availability and efficiency requirements of an access control system in CPS. Second, we define a Personalization Policy Checking (PPC) Problem to determine whether a given personalization policy is satisfied in an access control state. We give the computational complexity of the PPC problem in different subcases, and show that it is NP-complete in general. Third, we design a binary genetic search algorithm, whose improvements mainly include continuous update and selection of the best chromosomes in the population for iteration, and exploring and determining the optimal crossover and mutation probabilities, thereby improving the convergence efficiency of the algorithm. Finally, simulation results show the effectiveness of our proposed algorithm, which is especially fit for the case that the computational overhead is even more important than the accuracy in a large-scale CPS system.

INDEX TERMS Access control, personalization policy, genetic algorithm, cyber-physical system.

I. INTRODUCTION

Cyber-physical system is a controllable, credible, scalable and heterogeneous distributed cyber-physical equipment system. It acquires information based on the IoT perception environment, and processes the information through deeply integrated computing, communication and control capabilities to complete a given task [1], [2]. CPS can bring huge economic benefits and is widely used in digital medical instruments and systems adopting automatic acquisition and control technology, distributed energy systems, aerospace and aircraft control, industrial control, etc [3]–[5]. CPS has aroused great interest of industry investment and researchers.

The associate editor coordinating the review of this manuscript and approving it for publication was Po Yang.

In the CPS environment, if users in local nodes or nodes across IoT infrastructure access sensitive data without authorization, huge losses will occur [2], [6]. For cyber-physical systems, safety is facing increasing challenges, because illegal access may also come from various networks and physical interfaces in an increasing number of non-local IoT infrastructures [7]–[9]. Due to the heterogeneity of different IoT infrastructures, traditional access control are less effective in protecting sensitive data across IoT infrastructures. In the field of distributed cyber-physical systems, the research of access control is becoming more and more important for CPS designers and users.

The autonomy, heterogeneity and distribution of CPS nodes make access control mainly focus on multi-entity access control between different trust domains, while taking into account geographic location and resource ownership [10], [11]. The subject and object of access control are highly dynamic in the CPS environment and there exists a huge number of terminals and users. Therefore, the authorization relationship between users and permissions cannot be presented in advance, and the system authorization can only be performed automatically [11], [12]. However, whether this automatically authorized access control state satisfies the safety and availability requirements of the access control system needs to be determined by corresponding access control policies. Therefore, the study of access control policies cross-IoT infrastructure in the CPS environment has practical theoretical significance and application value.

Access control policies restrict the assignment of permissions to ensure the safety and availability of the access control system [13], [14]. However, there are still shortcomings in the existing research on access control policies. (1) Access control policies often focus on security or availability, and cannot effectively balance these two [11]. Multiple CPS nodes or even cloud nodes may be involved when performing tasks. These autonomous nodes have their own role-permission relationship and may not be able to accurately satisfy task requirements. The redundant permissions generated by ensuring availability will bring security risks to the system. If security is strictly ensured, it may lead to insufficient permissions and affect the smooth execution of tasks. (2) Existing access control policies consider a large number of access permissions with negligible impact on task execution, which will increase the scale of the problem and reduce the efficiency of access control decision-making. Ignoring the difference in nature importance between permissions and treating important permissions as ordinary permissions will also bring unpredictable risks to the system. (3) Determining whether a certain access control policy is satisfied in a system state is the key issue to efficient access control decision-making. However, this problem is difficult to solve, especially for access control systems authorized across autonomous domains in CPS environments. This is because the access control subjects and objects involved in the execution of a task may come from different CPS nodes with a large number of users and permissions. It is necessary to determine whether the access control state composed of these nodes satisfies the goals and constraints of considering the weight. This greatly increases the computational complexity. For example, an access control policy may require mutually disjoint user groups that can perform tasks independently to satisfy a certain number, and the weight of the permissions owned by a single user is less than a certain threshold.

It can be seen that existing access control policies are difficult to effectively ensure the safety and availability in CPS, and it is intractable to improve the decision-making efficiency of access control under a large amount of data. For this reason, this paper introduces the concept of weight of users and permissions, which expresses the importance of permissions/users from the attributes of operations, the sensitivity of objects and user attributes. Subsequently, we propose a refined personalization policy based on weights to improve the efficiency of access control decision-making while enhancing the safety and availability of the system. Then, we analyze the computational complexity of the problem that a given access control state satisfies the requirements of a personalization policy. To address this problem of general case, we design an efficient solution based on the idea of genetic algorithm. Generally, a given access control policy is the minimum requirement of the system. For example, there are three groups of mutually disjoint users in an access control system, and each group has all the permissions to perform sensitive tasks. But the access control policy requires two groups of mutually disjoint users to ensure the availability of the system. Therefore, verifying that the policy is satisfied only needs to find two sets of mutually disjoint users. It can be seen that this solution is more effective when the parameters required by the policy are smaller than the actual parameters of the system.

Briefly, the main contributions of this paper can be summarized as follows:

* We propose a personalization policy that considers the different natural importance of permissions and users. This policy describes the safety, availability and efficiency requirements of the access control system in a fine-grained way.
* We give a formal definition of the PPC problem which determines whether an access control state in CPS environment satisfies a given personalization policy, and present the computational complexity analysis of PPC problem in different subcases. In particular, we show that this problem is NP-complete in the general case.
* We design a Binary Genetic Search (BGS) algorithm, which first considers the efficiency of solving PPC problems. This algorithm improves the selection operation and crossover and mutation probability of genetic algorithm.
* Simulation results further demonstrate the effectiveness of the BGS algorithm, which is especially fit for the case that the computational overhead is even more important than the accuracy in a large-scale CPS system.

The rest of this paper is organized as follows. In Section II, we start with an overview of previous literature. Section III presents the formal definition of the personalization policy and the PPC problem, and studies computational complexity of its variants subcases. We present an algorithm for the PPC problem in Section IV. In Section V, we implement the proposed algorithm. We conclude this paper in Section VI.

II. RELATED WORK

The unique technical requirements and constraints of CPS make the existing research on automatic authorization of access control focuses on the discovery method of attribute-permission association in attribute-based access control (ABAC). And provides flexible control and management through the mapping mechanism of user-role and role-permission in role-based access control (RBAC) [11].

ABAC regards attributes as the key element of access control, which effectively solves the problem of large-scale, dynamic and private fine-grained access control in the CPS. ABAC first establishes the attribute set and describes the access control policy, and then responds to the access control request and updates the access control policy during execution [12]. RBAC guarantees flexible control and management of objects through a dual authority mapping mechanism, and provides inter-domain role mapping and constraint verification methods in cross-entity access control of CPS [15], [16]. When constructing attribute set and permission mapping, usually use role engineering or attribute engineering top-down or bottom-up method to mine roles or attributes to further authorize users. However, the automatically authorized access control state may not necessarily satisfy the safety and availability requirements of the access control system.

The access control policy which used to restrict permission assignment to ensure safety and availability in access control system is a main research for several decades [17]. The research of access control policy originated from the safety analysis of access control system, which determines whether an access control system can reach a state in which an unsafe access is allowed [18]. In the earliest work, the safety of the access control system is the focus of consideration, its purpose is to ensure the safety of the access control system when performing tasks and prevent abuse of authority. The separation of duty (SOD) policies is a typical policy used to ensure safety [19]. It prevent a set of users less than a certain threshold from being fully authorized to perform sensitive tasks [15], [16], [20]. Excessive pursuit of system safety may lead to unavailability of the system. For example, an access control state that satisfies strict safety requirements does not have the full permissions to perform tasks. Therefore, subsequent research also focuses on the availability of the access control system. Resiliency policy requires that absent any s users, there is still exist d mutually disjointed set of users which number is less than t and each set has all permissions in P to perform tasks to ensure availability of system [21], [22].

The problem of determine whether a certain access control state satisfies a given access control policy in the CPS environment is difficult to solve. For example, the problem of checking whether an access control state satisfies a resiliency policy is intractable (NP-hard) in the general case, and is in the Polynomial Hierarchy (in $\mathrm{coNP}^{\mathrm{NP}}$) [21]. In this paper, although we have comprehensively optimized the description of the policy to ensure that it is easier to solve while enhancing the safety and available effect. However, the policy proposed in this paper takes into account the weight of users and permissions, which obviously increases the difficulty of analyzing the problem.

The policy checking problem is difficult to solve under general case. The existing access control policy checking problem is to reduce the system scale through preprocessing, and then solve it by a satisfiability problem (SAT) solver [22]. However, due to the massive data scale of the CPS environment, which makes the implementation of this scheme require a great system overhead. Genetic algorithm has been proved to be effective in dealing with many problems, especially in dealing with NP-complete problems [23]–[29]. This is because the fitness value of the optimal solution can be calculated for this type of problem. The optimization goal of genetic algorithm is to make the solution set convergence to the optimal solution with higher efficiency. For example, the literature [24] proposed multi-granularity genetic algorithm that adopts a multi-granularity space strategy based on a random tree, which accelerates the searching speed of the algorithm in the multi-granular space. The literature [25] optimized crossover and mutation operations were devised to make the algorithm converge more quickly in solving the multi-processor scheduling problem in cloud data-centers. Aiming at the policy checking problem, this paper optimizes the genetic algorithm in many aspects to achieve the ideal solution effect.

In summary, the existing access control policy describes the safety or availability of the access control system, but it does not give a good balance between these two aspects, and it is difficult to apply in a distributed CPS environment. This paper proposes an access control policy applied in the CPS environment, defines and analyzes the computational complexity of the weighted policy check problem. Through the analysis of genetic algorithm, it can be seen that the algorithm can efficiently obtain the approximate solution of the problem. Therefore, this paper improves the algorithm to obtain better efficiency and accuracy.

III. PROBLEM FORMULATION

The individuality of every permission/user means that it has different nature and importance. It is a key topic that should be introduced to access control policy of CPS environment, but ignored. In this section, we propose a personalization policy that takes into account the specificity of permissions/users and be used to ensure the safety, availability, and efficiency of the access control system.

A. PERSONALIZATION POLICY

The personalization policy considers the particularity of permissions that have different natural and importance. In financial institution's access control systems, for example, the permission writes asset data is more important than the permission reads asset data. The weight is a value attached to a permission/user representing its importance and we introduce it to personalization policy. Here, we present an example to motivate the new features of the notation about the weight of permission/user to optimize the access control policy. Let us assume that the permission set is p_1, p_2, p_3, p_4, permissions p_1 and p_2 are assigned to u_1, permissions p_3 and p_4 are assigned to u_2, permissions p_1 and p_3 are assigned to u_3, permissions p_2 and p_4 are assigned to u_4. It is obvious that both $\{u_1, u_2\}$ and $\{u_3, u_4\}$ are the solutions and each solution has all permissions to perform tasks. However, it may not make any sense for choosing $\{u_1, u_2\}$, if the permissions $p_1$and p_2 are more important resulting weighted u_1 beyond a certain threshold. This is because that it is easier to put the system at unpredictable risk if a user has too important permissions. Furthermore, certain permissions that may be more critical for system can only be owned by special users, other users cannot be authorized in the process of performing sensitive tasks. Safety is an important factor that we consider, and availability also needs to be considered because it is related to the smooth execution of the task. For example, in the previous example, there are two mutually disjoint user sets to perform sensitive task, this means that even if any one of the users is absent, the task can still be executed.

There are a lot of resources in the CPS environment. If the access permissions of these resources are all taken into account in the access control system, it will bring great system overhead and affect the efficiency of access control decision-making. Therefore, in order to enhance the availability of the system, we do not consider non-essential permissions into the access control system. We use weights to indicate the importance of these resource access permissions to the system. We set a threshold according to the importance of the task, and do not add permissions with a weight less than a certain threshold to the access control policy. This is because the abuse of these permissions with lower weights has a tolerable impact on the smooth execution of tasks, and the deficiencies of these permissions can be resolved through temporary authorization.

The weight of permissions/users is a value between 0-1 that weighs the importance of permissions/users from the attributes of operations, the sensitivity of objects, and user attributes [30]. In this section, formal definition of the weight of permission and methods of calculating them is not discussion. We assume that the weight of permissions is determined by the system and the weight of users is the sum of the weighted user's permissions. The personalization policy is defined as follows.

Definition 1 (Personalization Policy): Given a set U of users, a set P of permissions, the personalization policy satisfy the following constraints:

* ***Safety constraint:*** *A safety constraints is denoted as $PP\langle \omega, U_F(p_f)\rangle$, where $\omega \geq 0$. p_f is very important to the system and can only be assigned to users in the user set U_F.We say that $PP\langle U, P, \omega, U_F(p_f)\rangle$ is satisfied if and only if the following conditions hold:*
  - *$\exists p_f \in P(U_F)$ and $p_f \notin P(U_n)$ where $U_n = U - U_F$, $P(U_F)$ denotes all permissions assigned to the users set U_F.*
  - *$\exists u_i \in U_P$ and $U_P = \bigcup_{W(u_j)<\omega} u_j$, where $u_j \in U$ and $W(u_j)$ denotes the weight of the u_j.*
* ***Efficiency constraint:*** *A efficiency constraints is denoted as $PP\langle \omega_0\rangle$, where $1 \geq \omega_0 \geq 0$, We say that $PP\langle U, P, \omega_0\rangle$ is satisfied if and only if the following conditions hold:*
  - *$\exists p_i \in P_P$ and $P_P = \bigcup_{(W(p_j)>\omega_0)} p_j - P_F$, where $p_j \in P$ and $P_F = \bigcup p_f$ denotes the permissions set of all p_f.*
* ***Available constraint:*** *A available constraints is denoted as $PP\langle U, P, \kappa\rangle$, where $0 \leq \kappa \leq n$ are positive integer. We say that $PP\langle \kappa\rangle$ is satisfied if and only if the following conditions hold:*
  - *$\exists\{P(U_i), P(U_j)\}$, and $P(U_i) = P(U_j) = P_P$, $U_i \cap U_j = \phi$, and $U_i, U_j \subseteq U_P$. where $0 \geq i, j \geq \kappa$, $i \neq j$.*

In order to distinguish different types of permissions and user groups. We define P_P, U_P, P_F, U_F as pivotal permissions, pivotal users, fixed authorized permissions and fixed authorized users respectively, as shown in definition 1. We define $P_N = P/P_F$ as non-fixed authorized permissions. We define the permissions with a weight less than ω_0 as general permission, denoted as P_G, and users with a weight greater than ω_0 are dangerous users, denoted as U_D.

To specify a subcase of the personalization policy, we combine the three constraints and write it followed by the list of constraints within a pair of braces. For instance, $PP\langle P, U, \kappa, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$. An access control state satisfies such a personalization policy if and only if fixed authorized permissions $\{p_i, \ldots, p_j\}$only belongs fixed authorized users $\{U_{f1}, \ldots, U_{fn}\}$ respectively, exist at least κ mutually disjoint sets of users such that each set has all authorized pivotal permissions and total weight of permissions authorized by each users is no more than ω.

Suppose we now give a personalization policy as $PP\langle P, U, \kappa, \omega_0, \omega, \{Mike(Ratify)\}\rangle$. This policy requires that fixed authorizations permission *ratify* only assigned to user *Mike*. If $\kappa = 2$ and $\omega_0 = 0$ is set, the policy requires that overall permissions except *ratify* are assigned to at least two mutually disjointed sets of users. If $\kappa = 2$ and $\omega_0 = 0.35$ is set, the permission excepted not only *ratify* but also permissions with a weight less than 0.35. If $\omega = 1.2$ is set, this means that the weight of each user in each mutually disjointed user groups is no more than 1.2. If we set $\omega = \infty$, this means that the weight of users is unrestricted.

Example 1: Given the access control state shown in Figure 1, all permissions in a fund publishing task are $P = \{input, issue, view, ratify\}$ and weighted to 0.7,0.5,0.3 and 0.9, respectively. All users are $U = \{Alice, Bob, Ed, Mik, Harry, Jack\}$.

As shown in Figure 1, the personalization policy $PP\langle P, U, 2, 0, 1.2, \{Mike(Ratify)\}\rangle$ is satisfied, because existence of $U_1 = \{alice, ed\}$ and $U_2 = \{bob, jack\}$ have full pivotal permissions and weighted each user no more than 1.2. However, $PP\langle P, U, 2, 0, 1, \{Mike(Ratify)\}\rangle$ is not satisfied, because the weight of U_2 alone does not exceed 1. $PP\langle P, U, 3, 0, \infty, \{Mike(Ratify)\}\rangle$ is not satisfied, because this access control state has only two mutually disjoint sets of pivotal users with all pivotal permissions. But $PP\langle P, U, 3, 0.35, \infty, \{Mike(Ratify)\}\rangle$ is satisfied, because this access control state has three mutually disjoint sets of pivotal users has pivotal permissions *input* and *issue*, the weight of permission *view* is less than 0.35 means that it's not importance for the task, so the access control system is not considered.

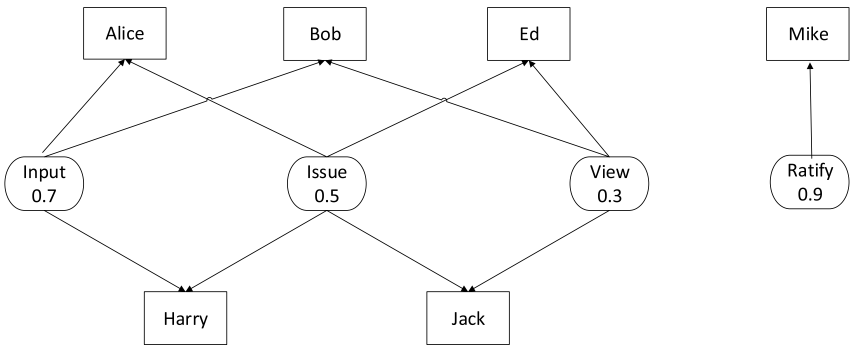


FIGURE 1. An example of access control state.

The parameters κ requires that existing κ mutually disjoint sets of users can be perform tasks respectively, mean that any $\kappa - 1$ pivotal users to be absent in emergency situations, there is still exist one independent team of users to perform tasks. Such as in the example 1, the access control state satisfies $\kappa = 2$, mean that the system can be able to tolerate any one pivotal user absent. Furthermore, even if absents any number of pivotal users in $\kappa - 1$ user sets, the system can still perform tasks. The parameters ω requires that the weight of a single user in any user set is no more than ω, which prevents a single user has more importance permissions to ensure the system safety. Obviously, if the parameters ω is given, then the number of users in each sets is no less than $\lceil W(P_P)/\omega \rceil$, where $W(P_P)$ is weight of all pivotal permissions. Such as in the Example 1, if given $\omega = 0.8$ then $\lceil W(P_P)/\omega \rceil = 2$, it means the number of users in each sets is no less than 2.

B. PERSONALIZATION POLICY CHECKING PROBLEM

In access control system, U represents all users and P represents all permissions, assignment relationship between the user and the permission is represented as $UP \subseteq U \times P$. How to efficiently determine whether the existing access control state UP satisfies a given access control policy is the key to the access control decision. For this reason, we now give a formal definition of the problem and analyze its computational complexity.

Definition 2 (Personalization Policy Checking (PPC) Problem): Given a personalization policy PP and an access control state UP, UP satisfies PP is denoted as the $sat_{PP}(UP)$. Determining whether $sat_{PP}(UP)$ is true is called Personalization Policy Checking Problem.

In special cases, the parameters of personalization policy *PP* are not always fully consider. For example, a personalization policy in the subcase $PPC\langle\kappa = 1\rangle$ has the form $PP\langle P, U, 1, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$ which means determines whether there exists a set of users have all pivotal permissions in P and weight of each user no more than ω. The subcase $PPC\langle\omega = \infty\rangle$ determines whether exist κ sets of users and each set has all pivotal permissions in P. The computational complexity results for PPC problem and it's various subcases are given as following theorem.

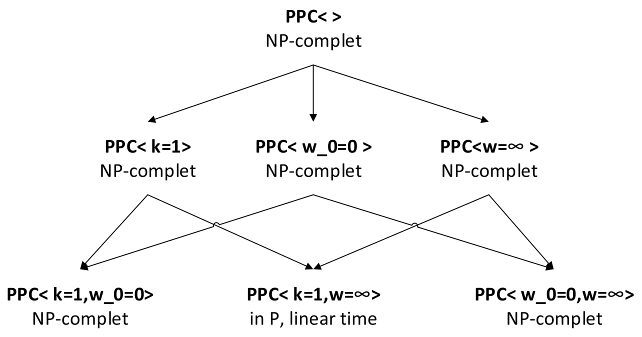


FIGURE 2. Computational complexity results for PPC problem in various subcases.

Theorem 1: The computational complexity of PPC problem and its subcases is shown in Figure 2.

We study the computational complexity of PPC problem in various subcases. The following lemma shows that the $PPC\langle\kappa = 1\rangle$, $PPC\langle\omega = \infty\rangle$, $PPC\langle\quad\rangle$ are NP-complete.

Lemma 1: $PPC\langle\kappa = 1\rangle$ is NP-complete

Proof: We prove that the $PPC\langle\kappa = 1\rangle$ is an NP problem: given a solution of the $PPC\langle\kappa = 1\rangle$ problem, it can be verified in polynomial time whether the solution is correct.

Next, we convert the NP-complete weighted set covering decision problem [31] to $PPC\langle\kappa = 1\rangle$ problem in Polynomial time, and show $PPC\langle\kappa = 1\rangle$ is NP-complete. In the weighted set covering problem, given a finite set S, a family $F = \{S_1, \ldots, S_m\}$ of subsets of S, and a budget B, the goal is to determine whether the weight of each S_i is less than B, where the union of S_i is S. Given an instance of the weighted set cover problem, we now construct an instance of $PPC\langle\kappa = 1\rangle$ in the following way: We create permissions $p_1, \ldots, p_m$ for each element in S, let $\omega = B$, m is the cardinality of the set S. we create $PP\langle P, U, 1, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$ and create an access control state: For each different subset $S_i (1 \leq i \leq m)$ in F, create a user u_i, so that all permissions and their weight values in S_i are assigned to u_i. Then whether $PP\langle P, U, 1, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$ is true if and only if there is a union of subsets in F that covers S, and the weight of any set in the subset is less than B.

Therefore, the PPC problem when $\kappa = 1$ is NP-complete problem. □

Lemma 2: $PPC\langle \omega = \infty \rangle$ is NP- complete

Proof: We prove that the $PPC\langle \omega = \infty \rangle$ is an NP problem: given a solution of the $PPC\langle \omega = \infty \rangle$ problem, it can be verified in polynomial time whether the solution is correct.

Next, we reduce the NP-complete DOMATIC NUMBER problem [32] to $PPC\langle \omega = \infty \rangle$. Given a graph $G(V, E)$, the DOMATIC NUMBER problem asks whether V can be partitioned into κ mutually disjoint sets $V_1, V_2, \ldots, V_k$ such that each V_i is a dominating set for G. V`is a dominating set for $G(V, E)$ if for every node u in $V - V^{`}$, there is a node v in V` such that $(u, v) \in E$. An instance of $PPC\langle \omega = \infty \rangle$ asks whether an access control state UP satisfies a policy $PP\langle P, U, \kappa, \omega_0, \infty, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$. Given a graph $G = (V, E)$, we now construct an instance of $PPC\langle \omega = \infty \rangle$ in the following way: We construct an access control state UP with n users $u_1, u_2, \ldots, u_n$ for n nodes in G and n permissions $p_1, p_2, \ldots, p_n$. $v(u_i)$ denotes the node corresponding to user u_i. In UP, user u_i is authorized for the permission p_j if and only if either $i = j$ or $(v(u_i), v(u_j)) \in E$. Let P denote the set $\{p_1, p_2, \ldots, p_n\}$. A dominating set in G corresponds to a set of users that together have all permissions in P. UP satisfies $PP\langle P, U, \kappa, \omega_0, \infty, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$ if and only if V contains κ mutually disjoint dominating sets.

Therefore, the PPC problem when $\omega = \infty$ is NP-complete problem. □

Lemma 3: $PPC\langle \quad \rangle$ is NP-complete

Proof: An instance consists of an access control state UP and a policy $PP\langle P, U, \kappa, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$. UP satisfies $PP\langle P, U, \kappa, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$ if and only if there exist at least κ mutually disjoint sets of users such that each set has all authorized pivotal permissions and total weight of permissions authorized by each user is no more than ω. If these κ sets are given, they can be verified in polynomial time. Therefore, $PPC\langle \quad \rangle$ is in NP, and the subcase of $PPC\langle \quad \rangle$ is NP-complete, then the $PPC\langle \quad \rangle$ is NP-complete. □

IV. THE BINARY GENETIC SEARCH ALGORITHM FOR PPC

The fact that PPC problem is intractable, as shown in Theorem 1, means that there exist difficult problem instances that take exponential time in the worst case. Therefore, we propose a Binary Genetic Search (BGS) algorithm to approximate solve PPC problems, which is inspired by the idea of the Genetic algorithm.

First, this algorithm performs preprocessing to reduce the system scale. Second, this algorithm execute optimized genetic algorithm and search algorithm within T seconds of system tolerance time. During this time, the number of mutually disjointed user sets which found in the first half of the population satisfy the parameters κ of policy, then stop and output result: true. If not, save the mutually disjoint user groups, randomly generate new chromosomes, and continue to iterate until κ groups are found. If the running time more than the system tolerance time of T seconds, it is uncertain whether the policy is satisfied, and the output result: false. This algorithm has a time complexity of $O(lmn)$, where l, m and n denote the number of actually performed iterations, the size of population and the number of all available users, respectively. The main notations used in this paper are shown in Table 1. Algorithm 1 shows the process of BGS for PPC problem.

TABLE 1. Main notations used in this algorithm.

Notation	Description
O[k]	The Optimal solution of k points.
E[m]	The population of m points.
E[i].fit	The fitness of E[i].
E[i].relfit	The relative fitness of E[i].
E[i].U[j]	The j th user in i th point of the population.
p_c	The probability for crossover.
p_m	The probability for mutation.
P_s	The size of the total pivotal permissions in the system after preprocessing.
U_s	The size of the total pivotal users in the system after preprocessing.
wep	The weight of extra permissions in perform tasks.
wmp	The weight of missing permissions in perform tasks.
W(E[i])	The weight of E[i].

Algorithm 1 BGS for PPC

```
Data: $UP[m][n]$, $W(p_i)$, $PP$, $P_m$, $P_c$, $T$
Result: $O[m][n]$, $Str$
1  Preprocessing();
2  while $runtime < Tsecond$ and $\kappa < max$ do
3      OGA( ) ;
4      Search( );
5      if $\kappa \geq max$ then
6          Str=True ;
7          exit(0);
8      end
9  end
10 if $\kappa < max$ and $runtime \geq Tsecond$ then
11     Str=False ;
12 end
13 return: Str;
```

This algorithm is optimized based on the idea of genetic algorithm, and has the characteristics of rapid convergence and evolution to the optimal solution. At the same time, because the PPC problem is an NP-complete problem, it can be determined in polynomial time whether the obtained solution is optimal.The algorithm is divided into three parts as shown in Algorithm 1. The first part is preprocessing, as shown in Algorithm 2; The second part performs optimized genetic algorithm as shown in Algorithm 3; the third part is to find mutually disjoint user groups such as algorithm 4 shown.

A. PREPROCESSING

We first determine whether the fixed authorization permissions in $PP\langle P, U, \kappa, \omega_0, \omega, \{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}\rangle$ in the preprocessing part only belongs to the fixed authorized user, that is, determine whether $\{U_{f1}(p_i), \ldots, U_{fn}(p_j)\}$ is true, if it is false, the policy is not satisfied. Secondly, we perform static pruning of users and permissions based on PP to reduce the scale of problem solving, which is of great help to improve the access control decision-making efficiency of CPS. Finally, we transform the PPC problem into the chromosome of genetic algorithm through coding. The preprocessing process in this section is shown in Algorithm 2.

Algorithm 2 Preprocessing Function Algorithm for PPC Problem

```
Data: $UP[m][n]$, $W(p_i)$, $PP$
Result: $U_pP_p[m][n]$, $Str$
1  if $U_{f1}(p_i), \ldots, U_{fn}(p_j) == \emptyset$ then
2      Str=False ;
3      exit(0);
4  else
5      foreach $p_i$ do
6          if $p_i \in P_f$ then
7              $P = P/p_i$;
8          end
9          if $W(p_i) < \omega_0$ then
10             $P = P/p_i$;
11         end
12     end
13     foreach $W(u_i) > \omega$ do
14         $U = U/u_i$;
15     end
16 end
17 return: Str;
```

1) STATIC PRUNING

The access permissions of large-scale resources in the CPS environment are taken into account in the access control decision system, which causes a large system overhead. Therefore, this section uses static pruning to delete users and permissions that do not need to be considered during the execution of the algorithm to improve the decision-making effectiveness of the access control system. Users and permissions in the following situations do not need to be considered.

* Fixed authorization permissions: For safety reasons, fixed authorization permissions can only be owned by specific users, while other users cannot be authorized, so we need to exclude these permissions when considering availability.
* Permission with weight less than ω_0: The importance of the permission is less than a certain threshold, so the permission does not need to be considered to improve the efficiency of access control decision-making. During task execution, the lack of such permissions can be obtained through temporary authorization.
* Users whose weight is greater than ω: If a selected user's weight is greater than ω, it does not satisfy the requirements of the access control policy, so there is no need to consider it.

2) ENCODING

After static pruning of users and permissions, a sub-state of the access control state composed of pivotal users and permissions is formed. Next, we optimize the genetic algorithm to discover the user group containing all the pivotal permissions. The genetic algorithm coding rules are as follows:

Given an access control state U_PP_P, U_P represents a set of m pivotal users, and P_P represents a set of n pivotal permissions. We use m-bit chromosomes to represent m users. When the i-th chromosome is 1, it means that user u_i is selected.

B. OPTIMIZATION GENETIC ALGORITHM

In this section, we introduce the optimized genetic algorithm (OGA). The core idea of the OGA function is to carry out genetic iterations according to the optimal crossover and mutation probabilities determined by experiments, updated optimal half of the population after Each iteration completes, and continue iterating with this population. Until the fitness of the first half of the population is the same and it is equal to the maximum value of fitness, and the value of relative fitness is also in a reasonable range. This means that the user set selected by each chromosome in the first half of the population covers all pivotal permissions. The execution steps of the optimized genetic algorithm (OGA) function are as follows, and Algorithm 3 gives the detailed execution process.

step i Select a population of m points $x_1, \ldots, x_m$ to represent the users set at random.

step ii Compute fitness: Compute the fitness and relative fitness of the role set using the evaluation function respectively.

step iii Replacement: Sort the m points according to the fitness value from large to small, sort the points with the same fitness according to the relative fitness, and then replace the latter half with the front half.

step iv Mutate: For each point x_i that $m/2 < i \leq m$ in the population and for each bit in x_i, with probability p_m, alter its value.

step v Crossover: For each y_j in the pair points x_i and $x_{(i+1)}$ from the $x_{m/2}, \ldots, x_m$, with probability p_c, exchange $x_i.y_j$ with $x_{(i+1)}.y_j$.

step vi Stop: If the front half of the population has the same fitness and equal to the maximum fitness, at the same time, the value of relative fitness is also in a reasonable range, stop.

C. FINDING MUTUALLY DISJOINT USER GROUPS

This section finds whether there are κ groups of mutually disjoint user groups in the solution $E[i]$ of the optimized

Algorithm 3 OGA Function Algorithm for PPC Problem

```
Data: $U_pP_p[m][n]$, $W(p_i)$, $PP$, $P_m$, $P_c$
Result: $E[i]$
1  $Rand(E[m])$;
2  while $E[0].fit \neq E[m/2-1].fit \neq P_s$ and
     $E[m/2-1].relfit \geq W(P_i) * 2$ do
3      foreach $i \in [0, n)$ do
4          $E[i].fit \leftarrow P_s - wep - 100wmp$ ;
5          $E[i].relfit \leftarrow W(E[i])$ ;
6      end
7      Sort($E[m].fit$,$E[m].relfit$) ;
8      foreach $i \leq m/2$ do
9          $E[m/2+i] \leftarrow E[i]$;
10     end
11     foreach $i > m/2$ do
12         foreach $j \in [0, n)$ do
13             if $rand() < p_m$ then
14                 $E[i].U[j] \leftarrow (E[i]\bar{.}U[j])$;
15             end
16         end
17         foreach $i\%2 == 0$ do
18             foreach $j \in [0, n)$ do
19                 if $rand() < p_c$ then
20                     $E[i].U[j] \leftrightarrow E[i+1].U[j]$;
21                 end
22             end
23         end
24     end
25 end
26 return:$E[i]$;
```

genetic algorithm. This paper proposes two methods. The first one is to encode the obtained solution $E[i]$ and use the optimized genetic algorithm to solve it again. The second method is to find whether there are κ groups mutually disjoint user groups. This method is an approximate solution method. The process is: If the obtained solution $E[i]$ belongs to a subset of any solution in the mutually disjoint solution set $O[k]$, replace it. If it is a solution that does not intersect with $O[k]$, add into the solution set. For the simulation experiment, we use the second method, which is shown in Algorithm 4.

The BGS algorithm is optimized based on the idea of genetic algorithm, and the improvements are as follows: First, in the process of execution, the optimal half of the population obtained by evolution is always updated, and the population is used to continue iterating. This is because the evolution based on the best solution has a high probability to get the better solution. Second, the mutation and crossover probability are determined through experiments. In the experiment, the value of the mutation probability is an integer multiple of the reciprocal of the population size. Third, the crossover operation selects the chromosome with the closest fitness.

Algorithm 4 Search Function Algorithm for PPC Problem

```
Data: $E[i]$
Result: $k$
1 foreach $i < m/2$ and $k \leq max$ do
2     if $E[i] \subseteq O[k]$ then
3         $O[k] \leftarrow E[i]$;
4     end
5     if $E[i] \cap O[k] == \phi$ then
6         $O[++k] \leftarrow E[i]$;
7     end
8 end
9 return:$k$ ;
```

These improvements greatly improve the efficiency of the algorithm converging to the optimal solution.

V. IMPLEMENTATION AND EVALUATION

In order to verify the effectiveness of the BGS algorithm, we have implemented it and performed several experiments using randomly generated instances. The implementation of our algorithm was written in C. Experiments have been carried out on a PC with an Intel(R) Core(TM) i5-8500T CPU running at 2.11 GHz, and with 4GB memory, running windows 10. In order to get closer to the real access control environment, we add two interference permissions that are not related to the task. It is assumed that the fixed authorization permissions satisfy the policy requirements and are pruned to generate instances. For each instance, 10 randomly generated test cases are run, the averages time of the test results are used to generate the runtime graphs, and the number of satisfaction in ten instances are used to generate another graph.

The evaluation function is used to evaluate the solutions. The fitness function is defined as $P_s - wep - 100wmp$, for more details, please refer to [23]. The relative fitness function is defined as follows.

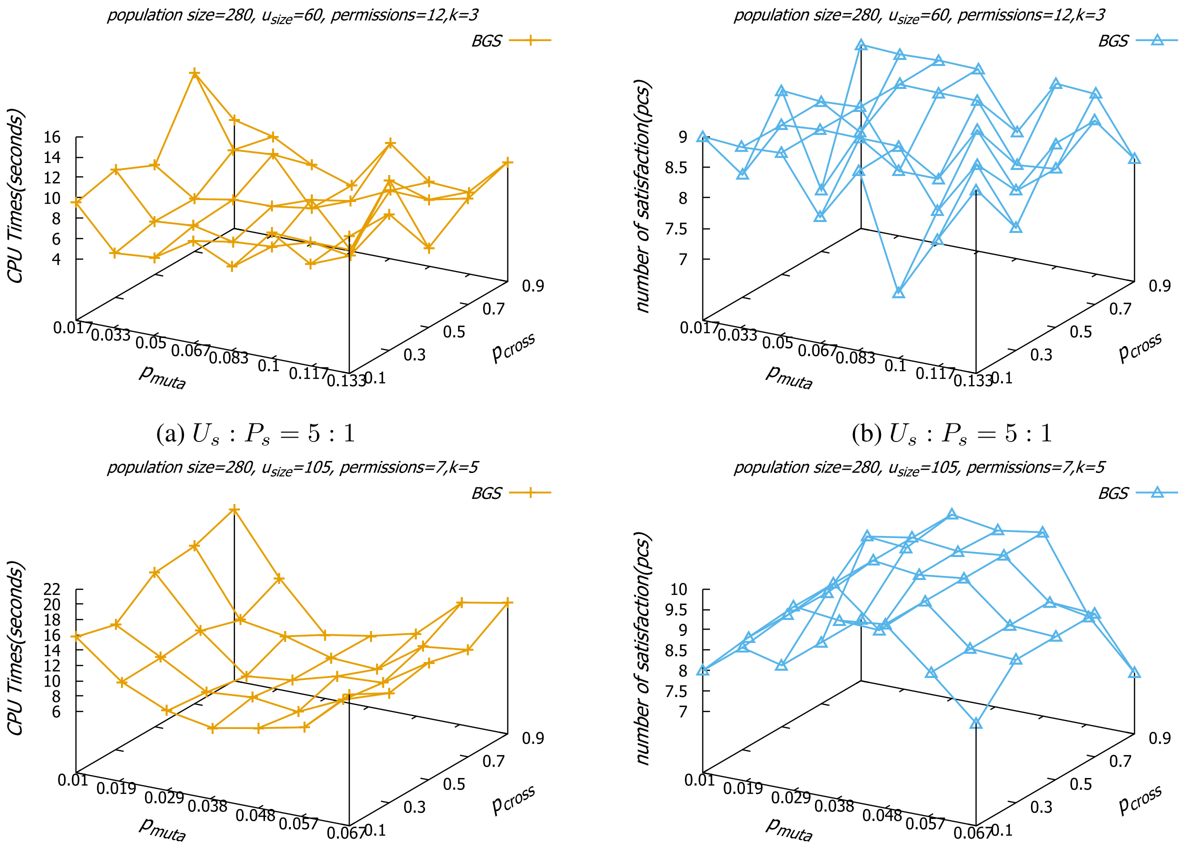
Definition 3 (Evaluation Function of Relative Fitness): Relative fitness of $E[i]$ is defined as:

$$E[i].relfit = \sum_{j=0}^{j=U_s} WP_{af}(E[i].U[j]), (ifE[i].U[j] = 1)$$

where $WP_{af}(E[i].U[j])$ represents weight of permissions only owned by $E[i].U[j]$.

A. EFFECTIVENESS OF MUTATION AND CROSSOVER

Figure 3 shows the average CPU times and number of satisfaction under different probability of crossover and mutation for the two test case (1) $U_{size} = 60, permissons = 12$ and $\kappa = 3$; (2) $U_{size} = 105, permissons = 7$ and $\kappa = 5$, the size of population $m = 280$, and the system tolerance time $t = 30$. The x-axis denotes the probability of mutation, and we fix its value as $1/U_{size}, \ldots, 8/U_{size}$ respectively. It can be clearly seen from Figure3(a) and (c) that the average CPU times is least when we choose the parameter $P_m = 3/U_{size}$



(a) $U_s : P_s = 5 : 1$ (b) $U_s : P_s = 5 : 1$

(c) $U_s : P_s = 15 : 1$ (d) $U_s : P_s = 15 : 1$

FIGURE 3. The runtime and number of satisfaction for different probability of mutation and crossover.

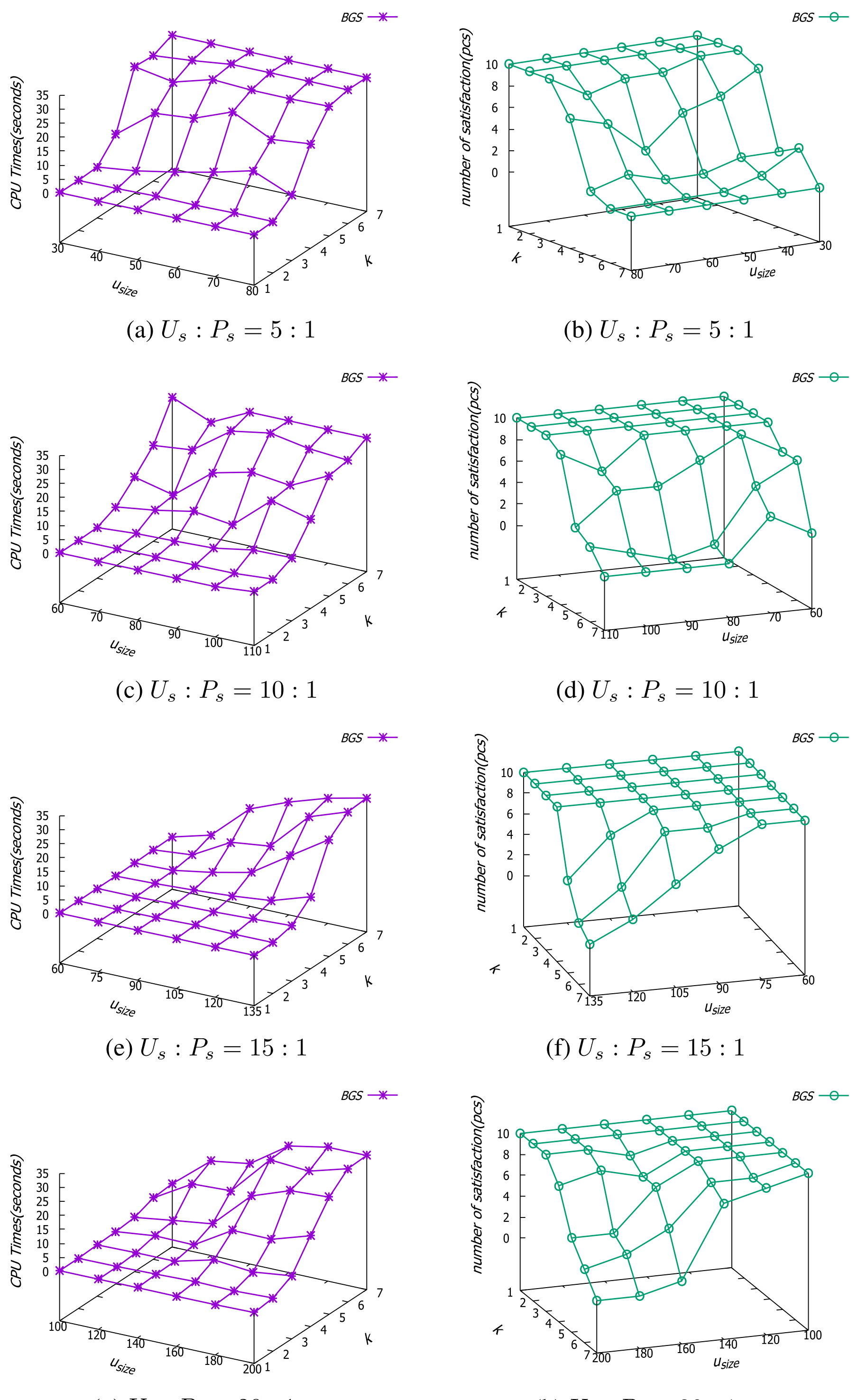
with fixed P_c. This means that it's easier to obtain a solution quickly by simultaneously mutating 3 bits in a chromosome. The average CPU times increases with the maximal P_c for the fixed P_m, because the less P_c will save the CPU times. As shown in Figure3(b)(d), the number of satisfaction is maximum when we choose the parameter $P_m = 3/U_{size}$ or $P_m = 5/U_{size}$ with fixed P_c, and when we choose the parameter P_c is close to 0.5 with fixed P_m. Together with the observation, we choose the parameters $P_m = 3/U_{size}$ and $P_c = 0.5$ for the remainder experiments.

Figure3(c)(d) shows longer CPU time and higher number of satisfaction than Figure3(a)(b). This is because when the ratio of users to permissions is large, it is easier to obtain mutually disjoint user groups, and the CPU time consumed will be reduced. The Figure3(c)(d) is clearer than Figure3(a)(b) on the curve trend of CPU time and the number of satisfaction. This is because if the ratio of users to permissions is small, the number of mutually disjoint user groups in the system is also small. In this case, it is difficult for the system to obtain a solution that satisfies the policy, and it may even not have a solution that satisfies the requirements of the policy. Therefore, if the ratio of users to permissions is small, the running time and the number of satisfactions of different random instances are very different.

B. RUNTIME AND NUMBER OF SATISFACTION FOR BGS ALGORITHM

Figure4 shows the results of running the experiments for the four test case (a) $U_{size} : P_{size} = 5 : 1$; (b) $U_{size} : P_{size} = 10 : 1$; (c) $U_{size} : P_{size} = 15 : 1$;(d) $U_{size} : P_{size} = 20 : 1$. The runtime and number of satisfaction depend on the total number of the users U_{size}, pivotal authorized permissions P_{size}, and parameter κ of the personalization policy.

In Figure4, as the parameter κ increases for the fixed U_{size}, the number of satisfaction reduces and the overall CPU time increases. This is because the larger the parameters required by the policy, the more difficult to satisfy for the system. As the total number of the users U_{size} increases for the fixed parameter κ, the number of satisfaction reduces and the overall CPU time increases, this change is not obvious when the value of κ is small. But the change is obvious when the value of κ becomes large, as shown in Figure4(g)(h). This is because as the policy parameter κ increases, it is more difficult for the system to obtain a solution that satisfies the policy, and the running time of some instances may reach the system tolerance time. The number of satisfaction increases also with the maximal $U_{size} : P_{size}$ for the fixed U_{size} and κ. The reason is that the more value of $U_{size} : P_{size}$ the more number of mutually disjoint sets of users. In Figure4(f)(h),



(a) $U_s : P_s = 5 : 1$ (b) $U_s : P_s = 5 : 1$

(c) $U_s : P_s = 10 : 1$ (d) $U_s : P_s = 10 : 1$

(e) $U_s : P_s = 15 : 1$ (f) $U_s : P_s = 15 : 1$

(g) $U_s : P_s = 20 : 1$ (h) $U_s : P_s = 20 : 1$

FIGURE 4. The runtime and number of satisfaction for different users and the parameters κ of policy.

as the number of U_{size} increases, the number of satisfaction reduced when the parameter κ more than 3. The reason is that the BGS algorithm will stop when CPU times are over the system tolerance time 30 second. Therefore, if we want to obtain the better number of satisfaction, we can increase tolerance time of the system.

Consequently, for the case that the system tolerance time is more important, we can make the BGS algorithm obtain the best possible solution within the system tolerance time. The BGS algorithm is able to solve the PPC problem even though in a larger scale system.

VI. CONCLUSION

In this paper, we have proposed a personalization policy that has reflected in the particularity of permissions/users and has described the safety, availability and efficiency requirements of the access control system in a fine-grained way. We have introduced the definition of PPC problems and have studied the computational complexity analysis of various subcases. We have shown that most instances of PPC problems are intractable. In particular, we have proposed a BGS algorithm to solve PPC problems. This algorithm has greatly improved the efficiency of the algorithm converging to the optimal solution of the PPC problem within the tolerance time of the system.

REFERENCES

[1] M. Wankhade and S. V. Kottur, "Security facets of cyber physical system," in *Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Tirunelveli, India, Aug. 2020, pp. 359–363.

[2] Y. Javed, M. Felemban, T. Shawly, J. Kobes, and A. Ghafoor, "A partition-driven integrated security architecture for cyberphysical systems," *Computer*, vol. 53, no. 3, pp. 47–56, Mar. 2020.

[3] T. Wang, D. Zhao, S. Cai, W. Jia, and A. Liu, "Bidirectional prediction-based underwater data collection protocol for end-edge-cloud orchestrated system," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4791–4799, Jul. 2020.

[4] T. Wang, H. Luo, X. Zeng, Z. Yu, A. Liu, and A. K. Sangaiah, "Mobility based trust evaluation for heterogeneous electric vehicles network in smart cities," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1797–1806, Mar. 2021.

[5] X. Liu, M. S. Obaidat, C. Lin, T. Wang, and A. Liu, "Movement-based solutions to energy limitation in wireless sensor networks: State of the art and future trends," *IEEE Netw.*, vol. 35, no. 2, pp. 188–193, Mar. 2021.

[6] J. Lu, C. Tang, X. Li, and Q. Wu, "Designing socially-optimal rating protocols for crowdsourcing contest dilemma," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 6, pp. 1330–1344, Jun. 2017.

[7] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 987–1001, 2019.

[8] T. Wang, Y. Lu, J. Wang, H.-N. Dai, X. Zheng, and W. Jia, "EIHDP: Edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for IoT systems," *IEEE Trans. Comput.*, early access, Feb. 19, 2021, doi: 10.1109/TC.2021.3060484.

[9] J. Lu, Y. Xin, Z. Zhang, X. Liu, and K. Li, "Game-theoretic design of optimal two-sided rating protocols for service exchange dilemma in crowdsourcing," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2801–2815, Nov. 2018.

[10] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Des. Test. IEEE Des. Test.*, vol. 34, no. 4, pp. 7–17, Aug. 2017.

[11] J. H. Huh, R. B. Bobba, T. Markham, D. M. Nicol, J. Hull, A. Chernoguzov, H. Khurana, K. Staggs, and J. Huang, "Next-generation access control for distributed control systems," *IEEE Internet Comput.*, vol. 20, no. 5, pp. 28–37, Sep./Oct. 2016.

[12] Z. Xu and S. D. Stoller, "Mining attribute-based access control policies," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 533–545, Sep. 2015.

[13] A. Margheri, M. Masi, R. Pugliese, and F. Tiezzi, "A rigorous framework for specification, analysis and enforcement of access control policies," *IEEE Trans. Softw. Eng.*, vol. 45, no. 1, pp. 2–33, Jan. 2019.

[14] P. Yang and L. Xu, "The Internet of Things (IoT): Informatics methods for IoT-enabled health care," *J. Biomed. Informat.*, vol. 87, pp. 154–156, Nov. 2018.

[15] M. U. Aftab, Z. Qin, N. W. Hundera, O. Ariyo, N. T. Son, and T. V. Dinh, "Permission-based separation of duty in dynamic role-based access control model," *Symmetry*, vol. 11, no. 5, p. 669, May 2019.

[16] X. Ma, R. Li, Z. Lu, J. Lu, and M. Dong, "Specifying and enforcing the principle of least privilege in role-based access control," *Concurrency Comput., Pract. Exper.*, vol. 23, no. 12, pp. 1313–1331, Mar. 2011.

[17] M. Narouei, H. Takabi, and R. Nielsen, "Automatic extraction of access control policies from natural language documents," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 3, pp. 506–517, Jun. 2020.

[18] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976.

[19] D. D. Clark and D. R. Wilson, "A comparison of commercial and military computer security policies," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, Apr. 1987, pp. 184–194.

[20] T. Zhu, D. Ye, W. Wang, W. Zhou, and P. Yu, "More than privacy: Applying differential privacy in key areas of artificial intelligence," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 4, 2021, doi: 10.1109/TKDE.2020.3014246.

[21] J. Crampton, G. Gutin, and R. Watrigant, "Resiliency policies in access control revisited," in *Proc. 21st ACM Symp. Access Control Models Technol.*, Shanghai, China, Jun. 2016, pp. 101–111.

[22] N. Li, Q. Wang, and M. Tripunitara, "Resiliency policies in access control," *ACM Trans. Inf. Syst. Secur.*, vol. 12, no. 4, pp. 20:1–20:34, Apr. 2009.

[23] J. Lu, Z. Wang, D. Xu, C. Tang, and J. Han, "Towards an efficient approximate solution for the weighted user authorization query problem," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 8, pp. 1762–1769, Aug. 2017.

[24] C. Li, S. Xia, Z. Chen, and G. Wang, "A multi-granularity genetic algorithm," in *Proc. IEEE Int. Conf. Big Knowl. (ICBK)*, Beijing, China, Nov. 2019, pp. 145–151.

[25] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A Johnson's-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 597–610, Jul./Sep. 2019.

[26] L. Cui, J. Zhang, L. Yue, Y. Shi, H. Li, and D. Yuan, "A genetic algorithm based data replica placement strategy for scientific applications in clouds," *IEEE Trans. Services Comput.*, vol. 11, no. 4, pp. 727–739, Jul./Aug. 2018.

[27] A. Zhdanov, "Generation of static YARA-signatures using genetic algorithm," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Stockholm, Sweden, Jun. 2019, pp. 220–228.

[28] B. Cao, S. Fan, J. Zhao, P. Yang, K. Muhammad, and M. Tanveer, "Quantum-enhanced multiobjective large-scale optimization via parallelism," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100697.

[29] G. Lin, S. Wen, Q.-L. Han, J. Zhang, and Y. Xiang, "Software vulnerability detection using deep neural networks: A survey," *Proc. IEEE*, vol. 108, no. 10, pp. 1825–1848, Oct. 2020.

[30] X. Ma, R. Li, and Z. Lu, "Role mining based on weights," in *Proc. 15th ACM Symp. Access Control Models Technol. (SACMAT)*, Pittsburgh, PA, USA, Jun. 2010, pp. 65–74.

[31] J. Lu, J. B. D. Joshi, L. Jin, and Y. Liu, "Towards complexity analysis of user authorization query problem in RBAC," *Comput. Secur.*, vol. 48, pp. 116–130, Feb. 2015.

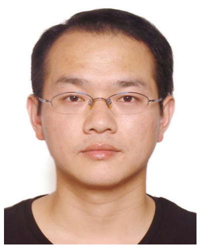
[32] G. J. Chang, "The domatic number problem," *Discrete Math.*, vol. 125, nos. 1–3, pp. 115–122, Feb. 1994.

ZHENG WANG received the M.S. degree from the Department of Computer Science and Engineering, Zhejiang Normal University, in 2017. He is currently a Teaching Assistant with Zhejiang Radio & Television University Haiyan College. His current research interests include optimizing intelligent algorithms and network security.

YANAN JIN received the Ph.D. degree in computer application technology from the Huazhong University of Science and Technology, in 2011. In 2012, he was a Visiting Researcher with Concordia University, Montréal, Canada. He is currently an Associate Professor with the School of Information Management and Statistics, Hubei University of Economics, Wuhan, China. His major research interests include web data mining and recommend systems.

SHASHA YANG received the M.S. degree from the Department of Computer Science and Engineering, Zhejiang Normal University, in 2020. She is currently a Teaching Assistant with the Xingzhi College, Zhejiang Normal University. Her research interests include mobile crowdsensing, incentive mechanism, and game theory.

JIANMIN HAN received the B.S. degree from the Department of Computer Science and Technology, Daqing Petroleum Institute, in 1992, and the Ph.D. degree from the Department of Computer Science and Technology, East China University of Science and Technology, in 2009. He is currently a Professor with the Department of Computer Science and Engineering, Zhejiang Normal University. His research interests include privacy preservation and game theory.



JIANFENG LU received the Ph.D. degree in computer application technology from the Huazhong University of Science and Technology, in 2010. In 2013, he was a Visiting Researcher with the University of Pittsburgh, Pittsburgh, USA. He is currently a Professor with the Department of Computer Science and Engineering, Zhejiang Normal University. His research interests include algorithmic game theory and incentive mechanism with applications to mobile crowdsensing and federated learning.