Benchmarking UAQ Solvers

Alessandro Armando
alessandro.armando@unige.it
DIBRIS, Università di Genova
Genova, Italia

Giorgia A. Gazzarata
giorgia.gazzarata@unige.it
DIBRIS, Università di Genova
Genova, Italia

Fatih Turkmen
f.turkmen@rug.nl
University of Groningen
Groningen, Netherlands

ABSTRACT

The User Authorization Query (UAQ) Problem is key in systems offering permission level user-system interaction, where the system automatically determines the roles that need to be activated in order to enable the requested permissions. Finding a solution to the problem amounts to determining an optimum set of roles to activate in a given session in order to obtain some permissions while satisfying a collection of authorization constraints, most notably Dynamic Mutually-Exclusive Roles (DMER) constraints. Even if the UAQ Problem is NP-hard, a number of techniques to solve the UAQ problem have been put forward along with encouraging experimental results based on different sets of synthetic benchmarks. We propose a methodology for designing parametric benchmarks for the UAQ problem and introduce and make publicly available a novel suite of parametric benchmarks that allows for the systematic assessment of UAQ solvers over a number of relevant dimensions. By running three prominent UAQ solvers against our benchmarks we provide a comprehensive and comparative analysis of unprecedented breadth from which it can be concluded that currently available benchmarks are not adequate to the task and that the reduction to PMaxSAT is currently the most effective approach to tackling the UAQ problem.

1 INTRODUCTION

The User Authorization Query (UAQ) Problem for Role-Based Access Control (RBAC) amounts to determining an optimum set of roles to activate in a given session in order to obtain some permissions while satisfying a collection of authorization constraints, most notably Dynamic Mutually-Exclusive Roles (DMER) constraints. The UAQ problem is key in systems offering *permission level* user-system interaction (where the system automatically determines the roles that need to be activated in order to enable the requested permissions), as opposed to *role level* interaction (where it is the user who explicitly determines and tells the system which roles must be activated).

The requested permissions come in two sets: a lower bound P_{lb} and an upper bound P_{ub} such that $P_{lb} \subseteq P_{ub} \subseteq P$, where P is the complete set of permissions. The permissions in P_{lb} are those that *must* be granted, whereas those in $P_{ub} \setminus P_{lb}$ are additional permissions that *can* be granted. It is then possible to either minimize or maximize the number of permissions in $P_{ub} \setminus P_{lb}$ to be granted depending on which objective (safety or availability, respectively) needs to be prioritized. If safety (availability) is chosen, then the number of permissions from $P_{ub} \setminus P_{lb}$ needs to be minimized (maximized, resp.). Notice that a certain degree of safety is achieved even if availability is preferred over safety, since no permission in $P \setminus P_{ub}$ can be granted. DMER constraints are of the form $DMER(\{r_1, \ldots, r_m\}, t)$. They constrain activation of roles by requiring that no user can activate t or more roles in $\{r_1, r_2, \ldots, r_m\}$.

Although the UAQ Problem is NP-hard [4, 6], a number of techniques to solve the UAQ problem have been put forward along with encouraging experimental results. These approaches can be broadly classified in two classes: *search-based techniques* [13, 18], whereby the problem is solved by systematically exploring a suitably defined search space, and *SAT-based techniques* [2, 12, 13, 18], i.e. techniques that leverage an encoding of the problem into either *(i)* a sequence of propositional satisfiability problems (SAT) [12, 13, 18] or *(ii)* a partial maximum propositional satisfiability problem (PMaxSAT) [2, 18] and then use SAT solvers and PMaxSAT solvers respectively as workhorses to find a solution to the problem (if any).SAT-based approaches leverages the fast paced advancements achieved by lively and dedicated research communities that organize competitions[1] and evaluations[2] of state-of-the-art solvers on a yearly basis.

Most of the techniques proposed in the literature have been experimentally evaluated by running them against different benchmark problems. These benchmarks are usually parametric in some relevant dimension of the problem (e.g. number of roles, number of DMER constraints, number of requested permissions) and aim at evaluating the scalability of the proposed techniques along them. The current state of affairs is nevertheless unsatisfactory for a number of reasons. The available benchmarks do not cover (and thus do not test the solvers against) all the relevant aspects of the problem. For instance, the problems used in [13] do not consider the case where the number of roles to be activated is maximized (i.e. obj=*max*) and therefore do not allow the evaluation of the respective UAQ instances. Furthermore, solvers are often evaluated against problems proposed by the same authors and this does not permit to assess the relative merits of the proposed techniques.

By leveraging the asymptotic complexity analysis of the solving procedures provided in [13], in this paper we propose a methodology for designing parametric benchmarks for the UAQ problem. As we will see, the methodology leads to benchmarks capable to *(i)* stress test solvers along dimensions of the problem for which no polynomial-time technique is known but also *(ii)* to check their effectiveness, by determining whether they efficiently solve problems that are known to be solvable in polynomial time.

By using our methodology we introduce and make publicly available a novel suite of parametric benchmarks that allows for the

AsiaCCS '20, June 01–05, 2020, Taipei, Taiwan
© 2020 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

[1]http://www.satcompetition.org
[2]https://maxsat-evaluations.github.io/

systematic assessment of UAQ solvers over a number of relevant dimensions. These include problems for which no polynomial-time algorithm is known as well as problems for which polynomial-time algorithms do exist. Guided by the asymptotic complexity analysis results given [13] we indicate, for each benchmark, its purpose and the expected behavior of solvers. The suite consists of 27 parametric benchmarks grouped in 13 families: 11 benchmarks (from 7 families) have the safety objective and 16 (from 6 families) have the availability objective. For each benchmark and each considered value of the parameter the suite includes 10 different problem instances.

We have used the new suite of benchmark problems as well as the benchmarks introduced in [13] to experimentally evaluate three solvers: 2D-Opt-Search [13] a search-based solver, 2D-Opt-CNF [13] that combines the reduction of the UAQ Decision Problem to SAT, a state-of-the-art SAT solver and a binary search, and UAQ-Solve [2], a SAT-based solver that implements a reduction of the UAQ Problem to PMaxSAT and uses any state-of-the-art PMaxSAT solver to tackle the problem. The experiments provide a comprehensive and comparative analysis of unprecedented breadth from which it is possible to draw a number observations:

- 2D-Opt-CNF and UAQ-Solve quickly solve all benchmark problems taken from [13]. This result indicates that this suite of benchmarks does not represent adequately the complexity of the problem and it is therefore of limited use to assess the effectiveness of the solvers.
- 2D-Opt-CNF and 2D-Opt-Search scale poorly even for (most) benchmarks in our suite for which polynomial-time procedures exist. This results is in stark contrast with the experimental results given in [13] from which one could be led to believe that these two solvers scale well in practice.
- For most benchmarks (2 out of 27), the behavior of UAQ-Solve meets the expectations that can be drawn from the known asymptotic complexity analysis results; this results is a strong indication of the validity of the methodology we propose in this paper, but it also indicates that there is still room for further investigation.
- UAQ-Solve outperforms both 2D-Opt-Search and 2D-Opt-CNF in the vast majority of the benchmark considered. This is not surprising since PMaxSAT solver implements sophisticated search algorithms specifically tailored to tackle optimization problems; in contrast, 2D-Opt-CNF tackles the optimization problem through a fairly naive binary search strategy and 2D-Opt-Search simply enumerates the solutions in order to find an optimum. Yet, prior to our experimental analysis, there was little evidence (if any) to support this conclusion.

Structure of the paper. In the next section we introduce the UAQ problem. In Section 3 we provide an overview of the techniques for solving the problem. In Section 4 we present our methodology for the generation of parametric benchmark problems and introduce the new suite of benchmarks. In Section 5 we present and discuss the experimental results. In Section 6 we discuss the related work and in Section 7 we conclude the paper with some final remarks.

2 THE UAQ PROBLEM

An *RBAC policy* is a tuple $RP = (U, R, P, UA, PA, \geq, C)$, where U is a set of users, R a set of roles, and P a set of permissions; users are associated to roles by the *user-assignment* relation $UA \subseteq U \times R$ and roles are associated to permissions by *permission-assignment* relation $PA \subseteq R \times P$; $\geq$ is a partial order on R, modeling the hierarchy between roles, i.e. $r_1 \geq r_2$ means that r_1 is *more senior than* (has all permissions of) r_2 for $r_1, r_2 \in R$; and C is a set of *dynamic mutually exclusive role (DMER)* constraints of the form $\mathrm{DMER}(\{r_1, \ldots, r_m\}, t)$, with $t \leq m$, stating that no user can simultaneously activate t or more roles in the set $\{r_1, \ldots, r_m\}$.

A user u is a *member* of role r when $(u, r) \in UA$. By R_u we denote the set of roles assigned to user u, i.e. $R_u = \{r \in R : (u, r) \in UA\}$. A user u *has permission* p if there exists a role $r \in R$ such that $(p, r) \in PA$ and u is a member of r. All the RBAC policies considered in this paper are assumed to be *finite*, i.e. U, R, and P have finite cardinality (and thus UA, PA, and $\geq$ have finite cardinality too). We treat permissions as if they are opaque (i.e. we do not consider the internal structure of permissions) and mutually independent (i.e. the possession of one or more permissions does not imply the possession of another permission). Let $p \in P$, we define $R_p = \{r \in R : (r, p) \in PA\}$ and $R_{P'} = \bigcup_{p \in P'} R_p$ for any $P' \subseteq P$. Similarly, let $r \in R$, then we define $P_r = \{r \in R : (r, p) \in PA\}$ and $P_{R'} = \bigcup_{r \in R'} P_r$ for any $R' \subseteq R$. Let $\varrho \subseteq R$, then we say that ϱ *satisfies* $\mathrm{DMER}(\{r_1, \ldots, r_m\}, n)$ iff $|\{r_1, \ldots, t_m\} \cap \varrho| < n$ and that ϱ *satisfies* C iff ϱ satisfies c for all $c \in C$.

Let S be a set of sessions and $user : S \longrightarrow U$ a function that associates each session $s \in S$ with the corresponding user.

A *User Authorization Query (UAQ)* is a tuple $q = (s, P_{lb}, P_{ub}, obj)$, where $s \in S$, $P_{lb} \subseteq P_{ub} \subseteq P$, and $obj \in \{\text{any}, \text{min}, \text{max}\}$.

Definition 2.1 (UAQ Problem). The *UAQ Problem for* $q = (s, P_{lb}, P_{ub}, obj)$ *in RP* is the problem of determining a set of roles $\varrho \subseteq R_{user(s)}$ such that *(i)* ϱ satisfies C, *(ii)* $P_{lb} \subseteq P_\varrho \subseteq P_{ub}$ and *(iii)* any other $\varrho' \subseteq R_{user(s)}$ that satisfies C and $P_{lb} \subseteq P_{\varrho'} \subseteq P_{ub}$ is such that

- $P_\varrho \subseteq P_{\varrho'}$, if $obj = \text{min}$;
- $P_{\varrho'} \subseteq P_\varrho$, if $obj = \text{max}$.

Definition 2.2 (UAQ Decision Problem). Let $w = (s, P_{lb}, P_{ub}, k_p)$ where $k_p \in \{\leq, \geq\} \times [0, |P_{ub} \setminus P_{lb}|]$. The *UAQ Decision Problem for* w *in RP* is the problem of determining a set of roles $\varrho \subseteq R_{user(s)}$ such that *(i)* ϱ satisfies C, *(ii)* $P_{lb} \subseteq P_\varrho$ and
(iii.a) $|P_\varrho \setminus P_{lb}| \leq n$ if $k_p = (\leq, n)$,
(iii.b) $|P_\varrho \setminus P_{lb}| \geq n$ if $k_p = (\geq, n)$.

3 SOLVING THE UAQ PROBLEM

In this section, we provide a systematic overview of search-based and SAT-based techniques for solving the UAQ problem. This will also give insights into the complexity of the problem.

3.1 Search-based Techniques

A naive two-step algorithm that first obtains a set of roles covering the desired permissions minimally and then checks whether the set of roles satisfies the constraints is proposed in [20]. However, the algorithm may not find a combination of roles that satisfies the constraints since the first step does not take any constraints into account.

An alternative approach, discussed again in [20] and improved in [18], amounts to

Procedure 1

(1) enumerating all possible role activations for the user,
(2) checking (in polynomial time as shown in [13]) whether the selected roles grant the requested permissions (i.e. fall between P_{lb} and P_{ub}), and satisfy the DMER constraints, and
(3) keeping the best (according to the security objective considered) solution encountered, if any.

The algorithm is clearly in $O(2^{|R|})$. The DPPL-based procedures proposed in [18] and the DFS-based algorithms proposed in [9] are optimized versions of this algorithm with additional preprocessing and pruning steps.

An alternative approach to solving the UAQ problem (adapted from [13]) is as follows:

Procedure 2

(1) enumerate all sets of roles $S_p \subseteq R_p$ for each $p \in P_{ub}$,
(2) check in polynomial time whether $S = \bigcup_{p \in P_{ub}} S_p$ is such that $P_{lb} \subseteq P_S \subseteq P_{ub}$ and S satisfies the DMER constraints, and
(3) keep the best (according to the security objective considered) solution encountered, if any.

The above algorithm is in $O(2^{\widehat{R_P}|P_{ub}|})$, where $\widehat{R_P} = \max_{p \in P} |R_p|$. To see this it suffices to observe that for each $p \in P_{ub}$ there are at most $2^{|R_p|}$ subsets S_p of R_p and thus there are in total at most $2^{\widehat{R_P}|P_{ub}|}$ candidate solutions to consider. When it is sufficient to activate "at most one role per permission", [13] shows that the above procedure is in $O(\widehat{R_P}^{|P_{ub}|})$ and hence is also fixed-parameter polynomial (FPP) in $|P_{ub}|$, i.e. it is polynomial-time if $|P_{ub}| \leq c$ for some constant c.

If the objective is min or any, then it is sufficient to activate at most one role per permission and this leads to the following, more efficient version of the algorithm:

Procedure 3

(1) enumerate all roles $r_p \in R_p$ for each $p \in P_{lb}$,
(2) check in polynomial time whether $S = \{r_p \in R_p : p \in P_{lb}\}$ is such that $P_S \subseteq P_{ub}$ and S satisfies the DMER constraints, and
(3) keep the best (according to the security objective considered) solution encountered, if any.

If the objective is min or any, it is in fact possible to consider the activation of individual roles granting the permissions in P_{lb} and the algorithm is in $O(|\widehat{R_P}|^{|P_{lb}|})$ [13].

If the optimization objective is max, then the "at most one role per permission" assumption does not hold and hence the FPP result cannot be applied in general. To illustrate consider a UAQ problem with $R = \{r_1, r_2, r_3\}$, $P = \{p_1, p_2, p_3, p_4\}$, PA such that $P_{r_1} = \{p_1, p_3\}$, $P_{r_2} = \{p_2, p_4\}$, $P_{r_3} = \{p_2, p_3\}$, $P_{lb} = \{p_1\}$, $P_{ub} = P$ and obj = max. The solution $\{r_1\}$ satisfies the "at most one role per permission" assumption but it is not optimal. In fact both $\{r_1, r_2\}$ and $\{r_1, r_2, r_3\}$ activate a larger (actually maximal) set of permissions, namely P. An alternative approach to tackling the problem for the max case is put forward in [13]:

Procedure 4

(1) enumerate all sets of possible role activations $R_a = R_1 \cup \cdots \cup R_n \cup R_{free}$, where $R_i \subseteq rs_i$ and $|R_i| < t_i$, for all constraints $DMER(rs_i, t_i)$ in C and $i = 1, \ldots, |C|$, and $R_{free} \subseteq R$ is the set roles that do not occur in the DMER constraints (and can thus be freely activated),
(2) check in polynomial time whether $P_{lb} \subseteq P_{R_a} \subseteq P_{ub}$ and R_a satisfies the DMER constraints.
(3) keep the maximum sized $|P_{R_a}|$ solution encountered, if any.

For the sake of simplicity but without loss of generality we assume that no role in R is assigned permissions which are not contained in P_{ub}. (It is easy to see that any UAQ problem in which one or more roles in R are assigned permissions in P_{ub} can be turned into an equisatisfiable UAQ problem that meets our assumption.) We now note that for each constraint $DMER(rs_i, t_i)$ in C there are $\sum_{k=1}^{t} \binom{|rs_i|}{k}$ subsets R_i of rs_i such that $|R_i| < t_i$. The number of sets $R_1 \cup \cdots \cup R_n$ is $\left(\sum_{k=1}^{t} \binom{|rs_i|}{k}\right)^{|C|}$, from which it easily follows that the enumeration of the set of roles $R_a = R_1 \cup \cdots \cup R_n \cup R_{free}$ grows as $O(\widehat{rs}^{|C|\hat{t}})$, where $\widehat{rs} = \max_{DMER(rs,t) \in C} |rs|$ and $\widehat{t} = \max_{DMER(rs,t) \in C} t$. This improves the upper bound $O(|R|^{|C|\widehat{t}})$ given in [13].

As shown in [13], the role hierarchy, $\succeq$, does not contribute to the computational complexity of the problem.

A summary of the results is given in Table 1.

Table 1: Complexity of solution techniques

Procedure	Objective	Complexity
1	any, min, max	$O(2^{\vert R\vert})$
2	any, min, max	$O(2^{\widehat{R_P}\vert P_{ub}\vert})$
3	any, min	$O(\widehat{R_P}^{\vert P_{lb}\vert})$
4	max	$O(\widehat{rs}^{\vert C\vert\hat{t}})$

3.2 SAT-based Techniques

Let $RP = (U, R, P, UA, PA, \succeq, C)$ be an RBAC policy with constraints and $q = (s, P_{lb}, P_{ub}, obj)$ a UAQ query for RP. Since RP is finite (i.e. the set U of users, R of roles, and P of permissions are all finite), the UAQ problem can be tackled by leveraging SAT solvers. This can be done in a variety of ways.

A first approach [13] amounts to reducing the UAQ Decision Problem to SAT and solving the optimization problem through binary search that leverage the SAT solver as an oracle for the decision problem. The second approach ([2, 17, 18]) eliminates the need for the binary search by directly encoding UAQ problems into PMaxSAT.

3.2.1 Reducing the UAQ Decision Problem to SAT. Let $w = (s, P_{lb}, P_{ub}, k_p)$ where $k_p \in \{\leq, \geq\} \times [0, |P_{ub} \setminus P_{lb}|]$. The reduction of the UAQ Decision Problem for w in RP to SAT amounts to generating a set of clauses $C^w_{RP} = C_{RP} \cup C^q$, where C_{RP} and C^w are defined below. We assume the existence of a propositional variable $\overline{r}$ for each $r \in R$ and a propositional variable $\overline{p}$ for each $p \in P$.

C_{RP} is the smallest set of propositional clauses satisfying the following conditions.

Core RBAC.

(1) for all $r \in R$ if $(user(s), r) \notin UA$ then $\neg\overline{r} \in C_{RP}$;
(2) for all $p \in P$ and $r \in R$ such that $(p, r') \in PA$ with $r \geq r'$, $(\neg\overline{r} \vee \overline{p}) \in C_{RP}$;
(3) for all $p \in P$, $(\neg\overline{p} \vee \bigvee\{\overline{r}$: exists $r' \in R, r \geq r'$, $(p, r') \in PA\}) \in C_{RP}$.

It is easy to see that the number of clauses above is in $O(|R||P|)$ and the number of propositional variables in $O(|R| + |P|)$.

MER Constraints. For all $MER(rs, n) \in C$, a CNF of the following formula is in C_{RP}:

$$\sum_{r \in rs} \overline{r} \leq n - 1 \tag{1}$$

As shown in [16], inequalities of the form $\sum_{x \in X} x \leq n$ can be succinctly encoded into CNF with $7|X|$ clauses and $2|X|$ additional propositional variables. Thus, constraints of the form (1) can be encoded with a number of variables and clauses in $O(|R|)$.

C^w is the smallest set of propositional clauses satisfying the following conditions.

Query.

- a unit clause $\overline{p} \in C_{RP}$ for each $p \in P_{lb}$;
- a unit clause $\neg\overline{p} \in C_{RP}$ for each $p \in P \setminus P_{ub}$;
- A CNF of the following formula in C_{RP}:

$$\sum_{p \in P_{ub} \setminus P_{lb}} \overline{p} \leq n \text{ if } k_p = \langle \leq, n \rangle$$
$$\sum_{p \in P_{ub} \setminus P_{lb}} \overline{p} \geq n \text{ if } k_p = \langle \geq, n \rangle$$

It can be shown that any solution to C^w_{RP} corresponds to a solution of the corresponding UAQ Decision Problem and vice versa.

3.2.2 Reducing the UAQ Problem to PMaxSAT. A PMaxSAT problem is given by a pair $\langle \mathcal{H}, \mathcal{S} \rangle$, where $\mathcal{H}$ and $\mathcal{S}$ are two finite sets of clauses, called "hard" and "soft" respectively. A solution to a PMaxSAT problem $\langle \mathcal{H}, \mathcal{S} \rangle$ is any truth-value assignment to the variables in $\mathcal{H}$ and $\mathcal{S}$ that satisfies all clauses in $\mathcal{H}$ and the maximum number of clauses in $\mathcal{S}$.

A UAQ Problem for $q = (s, P_{lb}, P_{ub}, obj)$ can be reduced to a PMaxSAT problem $\langle \mathcal{H}^q_{RP}, \mathcal{S}^q \rangle$, where $\mathcal{H}^q_{RP}$ is obtained from C_{RP} (as defined in Section 3.2.1) by adding

- a unit clause $\overline{p}$ for each $p \in P_{lb}$ and
- a unit clause $\neg\overline{p}$ for each $p \in P \setminus P_{ub}$;

and $\mathcal{S}^q$ comprises

- a unit clause $\neg\overline{p}$ if obj = min
- a unit clause $\overline{p}$ if obj = max

for each $p \in P_{ub} \setminus P_{lb}$. No soft clauses are included if obj = any.

It can be shown that any solution to C^w_{RP} (C^q_{RP}) corresponds to a solution of the corresponding UAQ Decision Problem (UAQ Problem, resp.) and vice versa.

In can be readily seen that for both reductions the number of clauses is in $O(|R||P|)$ and the number of propositional variables is in $O(|R| + |P|)$.

4 BENCHMARKS

Designing benchmarks suitable for the systematic assessment of UAQ solvers is not easy. A common approach [2, 9, 13, 18] is to focus on families of problems that are parametric on aspects of the problem that may contribute to its complexity. All other aspects are either set to a predefined, constant value or are randomly chosen in a given interval or according to some criterion. By running a solver against the instances corresponding to increasing values of the parameter, it is thus possible to obtain an estimation of how the solver scales along the dimension represented by the parameter. Unfortunately, the adequacy of the benchmarks proposed in the literature is seldom discussed. For instance, as we will see, some benchmarks have been reported to be solved efficiently by proposed solvers, e.g., in linear time along some parameter for which no polynomial-time algorithm is known. When this is the case, it seems likely that the benchmarks do not represent the complexity of the problem. (The alternative being that the solver used in the experiments improves over the known complexity results.)

The sheer number of elements that contribute to the definition of the UAQ problem complicates the selection of the parameters. The elements characterizing the RBAC policy include the number of roles $|R|$, the number of permissions $|P|$, the number of DMER constraints $|C|$ as well as their specific features (e.g. $\widehat{rs}$ and $\hat{t}$). One may even consider features of the PA relation, such as the maximum number of roles that contain any given permission (referred as $\widehat{R_P} = \max_{p \in P} |R_p|$ in Section 3). The components of the query also contribute to the complexity of the problem. These include the security objective (any, min, max), the number of requested permissions that *must* be granted, i.e. $|P_{lb}|$, and the number of requested permissions that *can* be granted, i.e. $|P_{ub}|$.

In previous work (see, e.g., [2, 9, 13, 18]), various benchmark problems parametric in any of these aspects have been put forward. To the best of our knowledge, the most extensive collection of parametric benchmarks so far is presented in [13] and summarized in Table 2. The benchmarks are parametric in $|R|$, $\widehat{R_P}$, $|D|$, $|rs|$, t, $|P_{lb}|$, $|P_{ub}|$, and obj. To illustrate, consider the benchmark problems named "roles" in the table. They are parametric in $|R|$ (with $|R|$ ranging from 25 to 200), have 500 permissions ($|P|$) with every permission being assigned to exactly 3 roles (and thus $\widehat{R_P} = 3$), and 10 MER constraints ($|C|$); each MER constraint contains 10 roles ($\widehat{rs}$) and the value of t is set to 3. The cardinality of P_{lb} and P_{ub} are set to 7 and 20 respectively. Only the optimization objective min is considered.

While the benchmarks in [13] provide a first attempt to provide a comprehensive evaluation along a number of significant dimensions, they still suffer from the following shortcomings:

(1) only the optimization objective min is considered, and they are therefore not suitable for evaluating the performance of the solvers when different optimization objectives, most notably max, are considered;
(2) it is not always clear if and how these benchmarks represent the complexity of the UAQ problem.

Table 2: Parametric Benchmarks from [13]

Name	SAT	$\|R\|$	$\|P\|$	$\widehat{R_P}$	$\|C\|$	$\widehat{rs}$	$\hat{t}$	$\|P_{lb}\|$	$\|P_{ub}\|$
roles	F	25..200	500	3	10	10	3	7	20
d	F	100	500	3	10..100	10	3	7	23
rolesPerConstr	F	300	1000	3	20	10..100	3	5	30
t	F	100	500	3	20	25	2..12	6	10
plb	F	100	500	3	10	10	3	1..11	$20 - \|P_{lb}\|$

The complexity results introduced in Section 3 can be used to validate and even guide the design of the benchmarks. If a family of UAQ problems is known to be solvable in polynomial time, then by running a solver against this family of problems we can check how the performance of the solver compares with that of the known algorithms: if the time spent by the solver grows, e.g., exponentially as the size of the problem increases, than the solver is clearly inefficient against this family of problems. Dually, if a family of UAQ problems is known to be NP-hard, but the time spent by any solver has a polynomial growth as the size of the problem increases, then this means that the family of the problems (i.e. benchmarks) considered does not represent adequately the complexity of the problem.

The above methodology can be used to understand and validate the benchmarks in Table 2. We start by observing that in all problems except plb, the value of $|P_{lb}|$ is fixed. Procedure 3 is therefore insensitive to the value of the respective parameter (R for roles, $\widehat{R_P}$ for rpp, etc.) and so should be any reasonably efficient solver when applied to these problems. All benchmark problems but plb can thus be used to check whether UAQ solvers are as effective as Procedure 3 as the value of the respective parameter increases. Benchmark plb is instead parametric in $|P_{lb}|$ and we therefore expect the solving time of Procedure 3 (and of any other solver) to increase exponentially as $|P_{lb}|$ increases. Notice that Procedure 1 is unlikely to be efficient here, since $|R|$ is set to a fairly large value, namely 100.

Driven by the above methodology, we propose two new families of parametric UAQ problems, one with obj = min and one with obj = max. The benchmarks with obj = min are summarized in Table 3:

- Plb_bigR and Plb_smallR are both parametric in $|P_{lb}|$. Plb_bigR can be used to stress test solvers for increasing values of $|P_{lb}|$: for large values of $|R|$ (here set to 100), the best known algorithm (i.e. Procedure 3) is exponential in $|P_{lb}|$ and thus we expect any solver to exhibit the same behavior. Plb_smallR can instead be used to check the effectiveness of solvers: Procedure 1 is in $O(2^{|R|})$ and thus we know that the problem can be solved efficiently for sufficiently small values of $|R|$ (here set to 10).
- R_bigPlb and R_smallPlb are both parametric in $|R|$ and are dual to Plb_bigR and Plb_smallR respectively. R_bigPlb can be used to stress test solvers for increasing values of $|R|$: for large values of $|P_{lb}|$ (here set to 100), the best known algorithm (i.e. Procedure 1) is exponential in $|R|$ and thus we expect any solver to exhibit the same behavior. R_smallPlb can be used to check the efficiency of solvers: Procedure 3 is in $O(\widehat{R_P}^{|P_{lb}|})$ and thus we know that the problem can be solved efficiently for sufficiently small values of $|P_{lb}|$ (here set to 1).
- RPhat_bigPlb, RPhat_medPlb, RPhat_smallPlb are parametric in $\widehat{R_P}$ and can be used to check the effectiveness of solvers: Procedure 3 is in $O(\widehat{R_P}^{|P_{lb}|})$ and thus we know that the problem can be solved efficiently for sufficiently small values of $|P_{lb}|$ (here set to 12, 4 and 1 respectively). Note that $|P_{lb}|$ is the degree of the polynomial and therefore the time spent by the solver may differ significantly (for the values of $|P_{lb}|$ considered) as $\widehat{R_P}$ increases.
- Pub, C, rshat, and that, are parametric in P_{ub}, $|C|$, $\widehat{rs}$ and $\widehat{t}$ respectively. Since these parameters do not contribute to the asymptotic complexity of any procedure presented in Section 3.1, these benchmarks can be used to check the effectiveness of solvers.

Notice that we do not include benchmarks parametric in the "size" of the role hierarchy since, as already pointed out in Section 3, it does not contribute to the complexity of the UAQ problem.

The benchmarks with obj = max are summarized in Table 4:

- Pub_bigRCt, Pub_smallR and Pub_smallCt are parametric in $|P_{ub}|$. Pub_bigRCt can be used to stress test solvers for increasing values of $|P_{ub}|$: for large values of $|R|$ and $|C|\widehat{t}$, Procedure 2 is to be preferred to Procedure 4. Since Procedure 2 is exponential in $|P_{ub}|$, we expect solvers to exhibit the same behavior. Pub_smallR (Pub_smallCt) can instead be used to check the efficiency of solvers: Procedure 1 (Procedure 4, resp.) is in $O(2^{|R|})$ ($O(\widehat{R_P}^{|C|\widehat{t}})$, resp.) and thus we know that the problem can be solved efficiently for sufficiently small values of $|R|$ ($|C|\widehat{t}$, resp.) (here set to 10, in both cases).
- RPhat_bigRCt, RPhat_smallR, RPhat_smallCt, parametric in $\widehat{R_P}$, are analogous to the previous case.
- C_bigRPub, C_smallR and C_smallRpPub are parametric in $|C|$. C_bigRPub can be used to stress test solvers for increasing values of $|C|$: for large values of $|R|$ and $|P_{ub}|$ Procedure 4 is to be preferred to Procedure 1 and Procedure 2. Since Procedure 4 is exponential in $|C|$, we expect solvers to exhibit the same behavior. C_smallR (C_smallRpPub) can be used to check the efficiency of solvers: Procedure 1 (Procedure 2, resp.) is in $O(2^{|R|})$ ($2^{\widehat{R_P}|P_{ub}|}$), resp.) and thus the problem can solved efficiently for sufficiently small values of $|R|$ ($\widehat{R_P}|P_{ub}|$, resp.).
- that_bigRPub, that_smallR and that_smallPub, parametric in $\widehat{t}$, are analogous to the previous case.

Table 3: Benchmark specifications for obj = min

Name	$\|R\|$	$\|P_{ub}\|$	$\widehat{R_P}$	$\|C\|$	$\widehat{rs}$	$\widehat{t}$	$\|P_{lb}\|$	TO (sec.)
Plb_bigR	200	400	5	0	-	-	5..50	600
Plb_smallR	10	400	5	0	-	-	5..50	30
R_bigPlb	10..100	400	5	0	-	-	100	600
R_smallPlb	10..100	400	5	0	-	-	2	30
RPhat_bigPb	200	400	2..12	0	-	-	10	600
RPhat_medPlb	200	400	2..12	0	-	-	4	600
RPhat_smallPlb	200	400	2..12	0	-	-	1	600
Pub	200	100..1000	5	50	8	3	10	600
C	200	400	5	10..100	8	3	10	5
rshat	200	400	5	10	5..50	3	10	5
that	200	400	5	40	8	2..8	10	5

Table 4: Benchmark specifications for obj = max

Name	$\|R\|$	$\|P_{ub}\|$	$\widehat{R_P}$	$\|C\|$	$\widehat{rs}$	$\widehat{t}$	$\|P_{lb}\|$	TO (sec.)
Pub_bigRCt	200	100..1000	5	50	8	3	10	600
Pub_smallR	10	100..1000	5	50	8	3	10	5
Pub_smallCt	200	100..1000	5	5	8	2	10	5
RPhat_bigRCt	200	400	20..60	50	25	4	4	600
RPhat_smallR	10	400	20..60	50	25	4	4	600
RPhat_smallCt	200	400	20..60	5	25	2	4	600
C_bigRpPub	200	400	5	10..100	8	3	10	600
C_smallR	10	400	5	10..100	8	3	10	5
C_smallRpPub	200	50	5	10..100	8	3	10	5
that_bigRPub	200	400	5	40	8	2..8	10	300
that_smallR	20	400	5	40	8	2..8	10	5
that_smallPub	200	50	5	40	8	2..8	10	5
rshat_bigRCt	200	400	5	10	5..50	3	10	100
rshat_medRCt	200	400	5	3	5..50	3	10	100
rshat_smallRCt	200	400	5	1	5..50	3	10	100
Plb	200	400	5	20	5	2	10..100	5

- rshat_smallCt, rshat_medCt and rshat_bigCt are parametric in $\widehat{rs}$. rshat_smallCt can be used to check the effectiveness of solvers. Procedure 4 is in $O(\widehat{rs}^{|C|\widehat{t}})$ and thus the problem can solved efficiently for sufficiently small values of $|C|\widehat{t}$. rshat_medCt and rshat_bigCt can be used to see how the values of $C|\widehat{t}$ affect the complexity of the problem.
- Plb is parametric in P_{lb}. Since this parameter does not contribute to the asymptotic complexity of any procedure presented in Section 3.1, this benchmarks can be used to check the effectiveness of solvers.

The benchmarks in Table 3 and in Table 4 are obtained by randomly generating a RBAC policy with the specified number of roles and permissions (ensuring that each permission is assigned to exactly $\widehat{R_P}$ roles), the specified number of DMER constraints of the form DMER($\{r_1, \ldots, r_m\}, t$) with $m = \widehat{rs}$, $t = \widehat{t}$ and a query with with P_{lb} with $|P_{lb}|$ randomly selected permissions and $P_{ub} = P$. It must be noted any UAQ problem can be readily reduced to an equivalent UAQ problem by eliminating all permissions in P that are not in P_{ub} and all roles that have assigned those permissions. By setting $P_{ub} = P$ we are thus eliminating the generation of problems that are *de facto* equivalent to problems of smaller size.

5 EXPERIMENTAL RESULTS

We used the benchmarks introduced in Section 4 to experimentally evaluate the performance of the following solvers:

- 2D-Opt-Search [13]: a search-based solver leveraging the FPP result;
- 2D-Opt-CNF [13]: a SAT-based solver that leverages the reduction of the UAQ Decision Problem to SAT, zChaff [1] as SAT solver a state-of-the-art SAT solver and a two-dimensional binary search to solve UAQ problems;
- UAQ-Solve [2], a SAT-based solver that employs a reduction of the UAQ Problem to PMaxSAT and employs any state-of-the-art PMaxSAT solver to tackle the problem. In the experiments presented in this paper we used the Loandra PMaxSAT solver [3].

Since the first two solvers perform a joint optimization of permissions and roles, i.e. minimizing (or maximizing) the number of roles and permissions in a solution, and the last solver optimizes only permissions we disabled the role optimization in those solvers. For each benchmark problem we generated 10 instances and ran the solvers against them. All data points in our plots represent the median value of the time spent by the corresponding solver. The experiments have been conducted on a PC with 2 64-bit Intel Xeon CPU X7350 (8 core) @ 2.93GHz and 47 GB RAM running Linux (Ubuntu 16.04.5 LTS).

For the benchmarks of Table 2 we set the timeout to 600 seconds. For the benchmarks of Table 3 and Table 4 the timeout is set to the values indicated in the rightmost column of the tables (TO).

The experimental results obtained by running the solvers against the benchmarks of Table 2 are shown in Figure 1. The experiments indicate that UAQ-Solve outperforms both 2D-Opt-Search and 2D-Opt-Search in most cases. But, most importantly, all problems are solved by 2D-Opt-Search and UAQ-Solve in a fraction of a second even for very large values of the parameter. We can thus conclude that these benchmarks do not represent the complexity of the problem and they are of limited value for assessing the effectiveness of UAQ solvers.

The experimental results obtained by running the solvers against the benchmarks with optimization objective set to min (cf. Table 3) are shown in Figure 2:

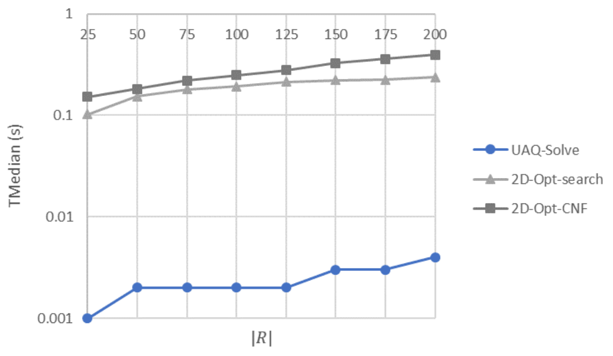
- The time spent by UAQ-Solve to solve Plb_bigR and Plb_smallR meets the expectations. For Plb_bigR the plot has a clear exponential growth, whereas for Plb_smallR is grows very slowly and even (slowly) decreases for $P_{lp} > 20$. The behavior of 2D-Opt-CNF on Plb_smallR is similar to that of UAQ-Solve, but when applied to Plb_bigR it reaches the timeout even for the smallest value of $|P_{lb}|$ (i.e. 5). 2D-Opt-Search can only solve Plb_smallR and Plb_bigR only for the smallest value of $|P_{lb}|$.
- Similar considerations hold for R_bigR and R_smallR. Notice that 2D-Opt-Search performs remarkably well for R_smallR but it reaches the timeout even for the smallest instance of R_bigR (i.e. 10).
- The results for RPhat_bigPlb, RPhat_mediumPlb, RPhat_smallPlb are of particular interest. As pointed out in Section 3 Procedure 3 is in $O(\widehat{R_P}^{|P_{lb}|})$ and if $|P_{lb}|$ is bounded, then it can solve the problem in time which grows as a polynomial of degree $|P_{lb}|$. It can be noted that UAQ-Solve quickly solves all problems in RPhat_smallPlb (for which $|P_{lb}| = 1$), while the plot is considerably more steep for RPhat_bigPlb (for which $|P_{lb}| = 12$). This is consistent with the growth of polynomials of degree 1 and 12 respectively. The plot for RPhat_mediumPlb (for which $|P_{lb}| = 4$) represents a intermediate situation. The behavior of 2D-Opt-Search on RPhat_smallPb and RPhat_medPb is similar to that of UAQ-Solve, but it reaches the timeout for Plb_bigR for a small value of $\widehat{R_P}$ (i.e. 2).
- The plot for Pub indicates that the time spent by UAQ-Solve is not insensitive to $|P_{ub}|$. This is however not surprising if we consider that in our benchmarks $P_{ub} = P$ and the size of the encoding is in $O(|R||P|)$ and thus it grows linearly with $|P|$. Notice also that the time spent by UAQ-Solve is very small even for very large values of $|P_{ub}|$. 2D-Opt-CNF reaches the timeout for the smallest value of $|P_{lb}|$ (i.e. 100), whereas 2D-Opt-Search reaches the timeout even for that value.
- As expected, UAQ-Solve quickly solves the remaining benchmarks, namely C, rshat, and that, whereas 2D-Opt-CNF and 2D-Opt-Search reach the timeout even for the smallest values of the respective parameters.

Figure 3 presents the experimental results obtained by running UAQ-Solve against the benchmarks with optimization objective set to max (cf. Table 4). We did not consider 2D-Opt-CNF nor 2D-Opt-Search since 2D-Opt-CNF appears to be unstable (it crashes on some instances)x and 2D-Opt-Search (by assuming the "at least one role per permission") may lead to sub-optimal results. From the experimental results we can draw the following observations:

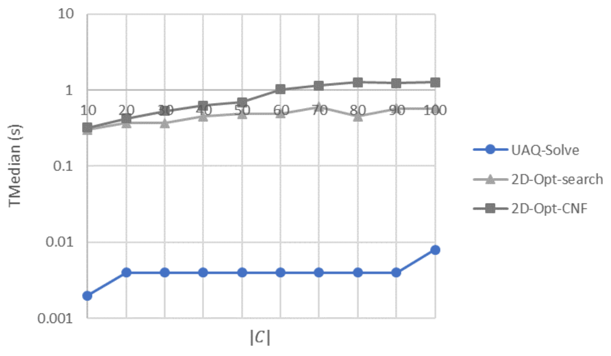
- The time spent by UAQ-Solve to solve Pub_bigRCt, Pub_smallR and Pub_smallCt are parametric in $|P_{ub}|$ meets the expectations. For Pub_bigRCt the plot has a clear exponential growth, whereas for Pub_smallR and Pub_smallCt the plots grow very slowly as $|P_{ub}|$ increases.
- Contrary to expectations the time spent by UAQ-Solve for RPhat_bigRCt does not increase (it actually decreases) as $\widehat{R_P}$ increases. We tried with with considerably larger values of $|R|$, $|C|$, and $\widehat{t}$ to no effect: UAQ-Solve remains unexpectedly fast. In this case it seems reasonable to conclude that our problem generation method does not yield instances that are representative of the complexity of the problem. Given this, it is not surprising that UAQ-Solve easily solves RPhat_smallR and RPhat_smallCt too.
- The time spent by UAQ-Solve to solve C_bigRPub, C_smallR and C_smallRpPub meets the expectations. For C_bigRPub the plot has a clear exponential growth, whereas for C_smallR and C_smallRpPub the plots grow very slowly as $|C|$ increases.
- Contrary to expectation UAQ-Solve quickly solves all benchmarks that_bigRPub, that_smallR and that_smallPub. Again this could be due to the fact that our problem generation method fails to produce instances that are representative of the complexity of the problem
- The plot for rshat_smallCt show that the problems cab be solved quickly as $\widehat{rs}$ increases as long as the value of $|C|\widehat{t}$ is sufficiently small (3 in this case). The plots for rshat_medCt and rshat_bigCt show the effect of larger values of $|C|\widehat{t}$ which is consistent with the growth of polynomials of degree 9 and 30 respectively.
- As expected, UAQ-Solve quickly solves the Plb benchmark.

6 RELATED WORK

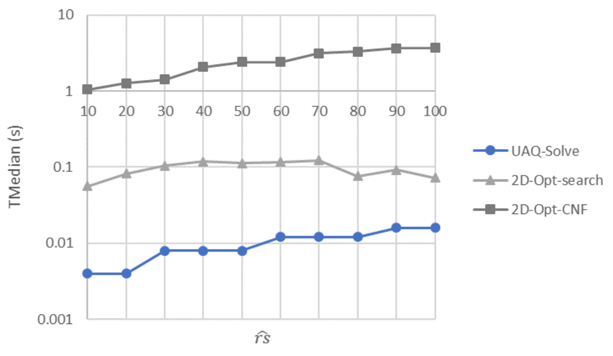
As UAQ is a central problem in RBAC systems, significant effort [9, 11, 12, 17] has been put to develop techniques for tackling it efficiently and to understand its underlying complexity. To our knowledge, [6] is the first paper that discusses UAQ where authors show that the complexity of finding minimal set of roles to be activated in a session that covers the permissions requested by the user is NP-complete. While they analyze UAQ in the presence of



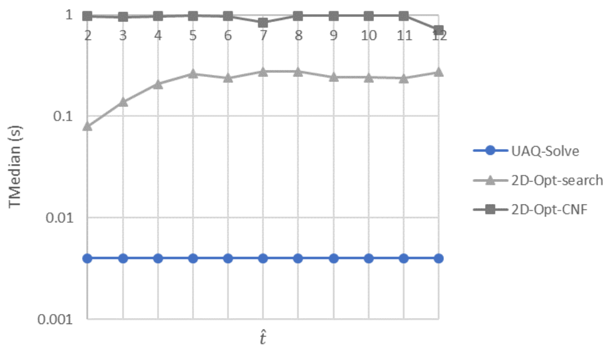
(a) roles Benchmark



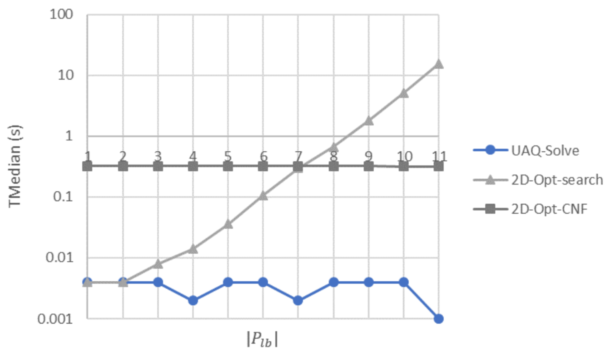
(b) d Benchmark



(c) rolesPerConstr Benchmark



(d) t Benchmark

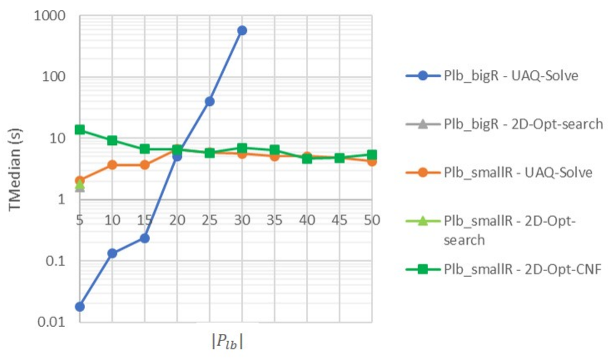


(e) plb Benchmark

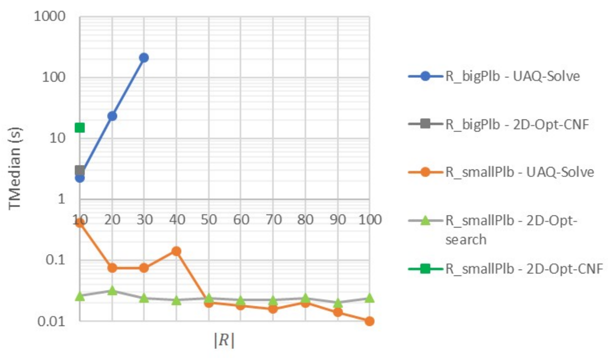
Figure 1: Performance of 2D-Opt-Search, 2D-Opt-CNF and UAQ-Solve on the benchmarks of Table 2

complex role hierarchies, they do not consider the constraint types (e.g. mutual exclusion of roles) available in RBAC. A natural extension of this work is presented in [20] where authors present two algorithms for solving UAQ problem instances. The first algorithm is a greedy search algorithm that looks for a set of roles covering the requested permissions while trying to minimize the additional permissions these roles provide. It is very efficient, but incomplete since it does not explore the space of all possible solutions. The second algorithm, aimed at providing completeness, is based on a simple generate-and-test strategy. It enumerates all subsets of roles assigned to the user until one is found that provides the needed permissions and satisfies all constraints. The problem with this algorithm is the first step may render the algorithm inefficient since it may need to generate $2^{|R|}$ solutions in the worst-case.

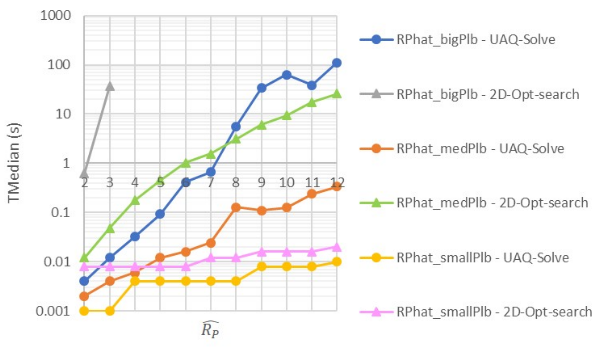
The more generic form for the UAQ problem where there are lower (P_{lb}) and upper (P_{ub}) bound permissions has been first proposed in [18]. The authors proposed two solutions for this problem. One is a variant of backtracking based search algorithm used in



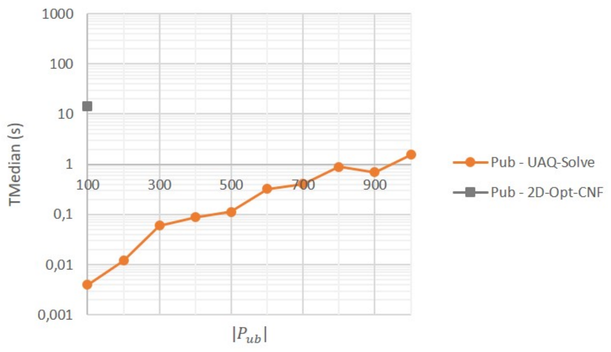
(a) Plb Benchmark



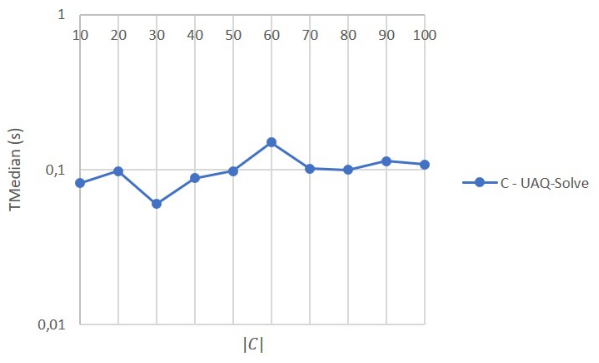
(b) R Benchmarks



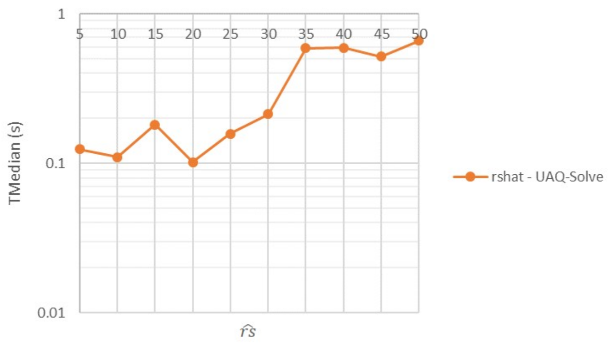
(c) RPhat Benchmark



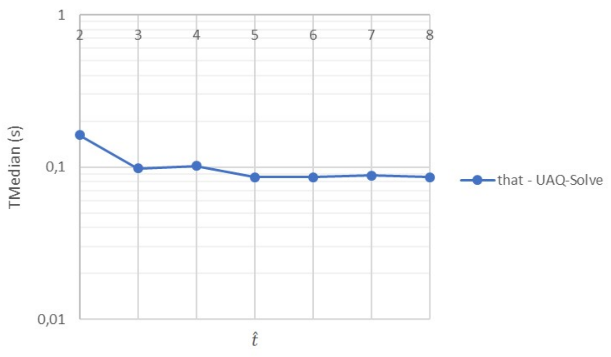
(d) Pub Benchmark



(e) C Benchmark

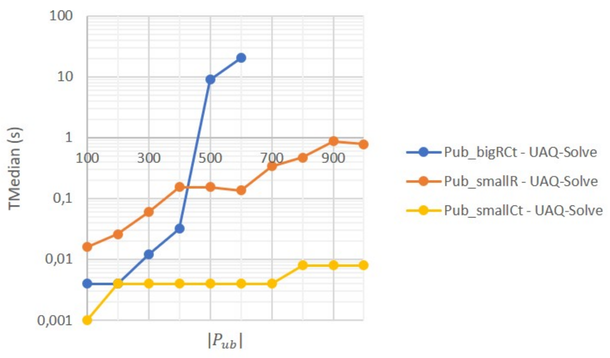


(f) rshat Benchmark

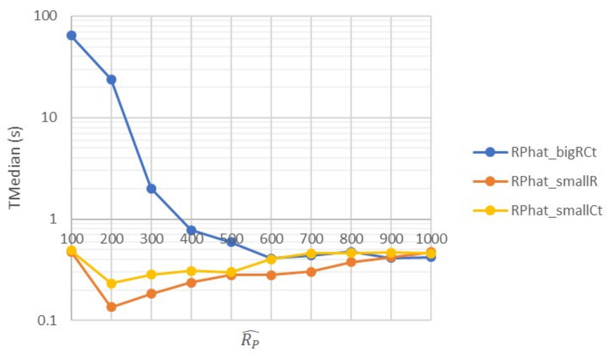


(g) that Benchmark

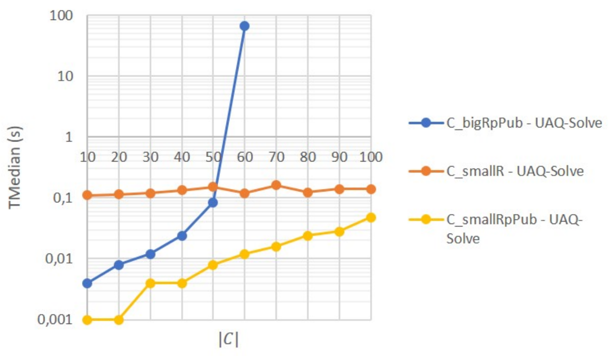
Figure 2: Performance of 2D-Opt-Search, 2D-Opt-CNF and UAQ-Solve on the benchmarks of Table 3



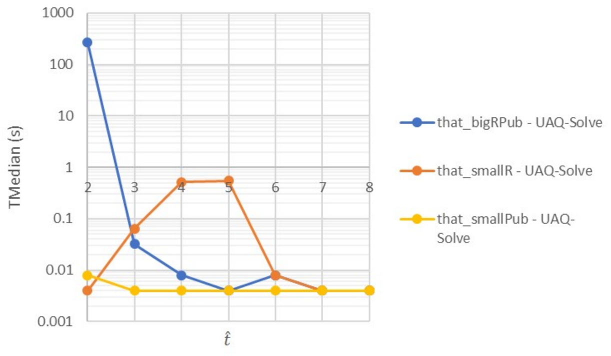
(a) Pub Benchmark



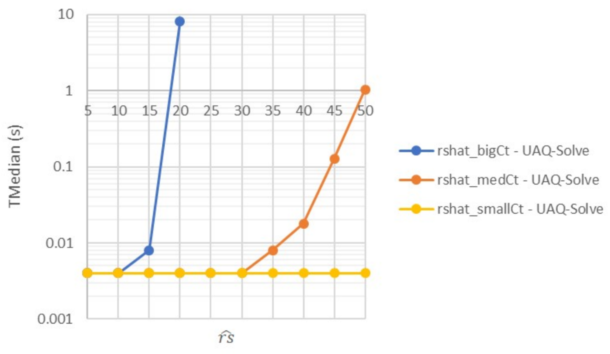
(b) RPhat Benchmark



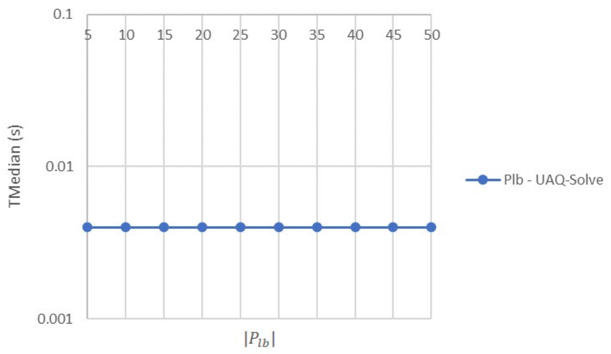
(c) C Benchmark



(d) that Benchmark



(e) rshat Benchmark



(f) Plb Benchmark

Figure 3: Performance of 2D-Opt-Search, 2D-Opt-CNF and UAQ-Solve on the benchmarks of Table 4

SAT solving, and the other is based on reducing the UAQ problem to MaxSAT. In general, the second solution performs better than the first one when *obj* = max or *obj* = min. The first approach is more efficient when an exact matching between the requested permissions and the available roles is sought. However, both approaches show very poor performance (as elaborately shown in [12]) when the number of roles increases.

The formal complexity analysis of different UAQ problem classes is also a point of strong scientific interest. For instance, [5] shows the complexities of UAQ problem by reducing it to a special case of set covering problem namely, container optimization. The reduction shows that both cases (*obj* = max and *obj* = min) are polynomial time Turing equivalents of container optimization problem, and are thus NP-hard. However, the paper does not consider UAQ problems with dynamic constraints as we do in this paper. Moreover, these works did not either conduct any experimental analysis or employed extremely simple RBAC instances to evaluate their proposals. A general framework for the UAQ problem, that includes the optimization of number of roles as well as of number of permissions in proposed in [9] along with a comprehensive computational complexity analysis of various sub-cases and search-based solving techniques. However the proposed framework does not include DMER constraints which are one of the distinguishing feature of the UAQ problem considered in our work.

Mousavi et al. [13] show that there are two versions of the UAQ problem with constraints: the decision problem and the optimization problem. The authors also provide an extensive experimental analysis of the proposed techniques through a family of parametric benchmarks. At a high level, the decision version reduces the UAQ problem to SAT with an encoding similar to ours. The optimization version invokes a SAT solver in binary search while trying to minimize (or maximize) the set of roles that can be activated. They then present various algorithms (search-based or SAT reduction as discussed in Section 3.1) to tackle them efficiently. The authors also show that the complexity result shown in [5] for the case $obj = \mathsf{max}$ is different when there are constraints in the UAQ instance. More specifically, they show that there is an upper bound (NP) for the general UAQ problem and the case $obj = \mathsf{max}$ is intractable if the UAQ instances have constraints. However, as we discussed in 4 in detail, our analysis revealed that the benchmarks they used in their experiments may not adequately reflect the complexities of various UAQ problem classes. An alternative approach, proposed again by Mousavi et al. [14], to the generation of benchmarks is to reduce the UAQ problem to constraint satisfaction problem (CSP) and employ a (hard) CSP instance technique [19]. This approach is particularly useful when generating hard UAQ problem instances, as the authors do, however they are hardly representative of the problem. In fact, our evaluations of their generation method mostly resulted with instances that are very easy to solve by a PMaxSAT solver.

More recently, a weighted variant of the UAQ problem has been proposed in [10] and [11] where the importance of a permission is also taken into consideration. The authors present various algorithms along with their complexity and compare them empirically. However, they consider role assignment (as opposed to role activation) as the primary constraint enforcement mechanism and the benchmarks used in their experiments is not systematic.

The research on the enforcement of security constraints, in particular Separation of Duty (SoD), in RBAC has also contributed to the discussion of UAQ. In [7] is has been observed that RBAC suffers from an under-specification problem due to the way "sessions" are employed in the standard [15]. The standard defines a set of high level functions to model the security requirements of applications while providing a bird-eye view to the authorization. However it fails in supporting some important principles, e.g. least privilege, for which run-time support becomes necessary. In [8] it is argued that that sessions are very useful for the dynamic management of roles. That this can be done efficiently is shown [2, 17]. More specifically, [2] shows that the definition of MER constraints can be extended so to cover multiple sessions (i.e. MS-MER) and role activation history (H-MER). This way, the authorization constraints can be span multiple sessions and the the role activations occurred in the past.

7 CONCLUSIONS

UAQ problem corresponds to policy evaluation procedure for role-based access control systems and plays a critical role in their functionality. It needs to be complete (i.e. finds a solution if there is one) and efficient (does not hinder the overall RBAC system performance). In this paper, we presented a systematic overview of the computational complexity, existing algorithms and available benchmarks pertinent to the UAQ problem. Our analysis of the currently available benchmarks revealed that they are inadequate to analyze the inherent complexity of different UAQ problem classes. We then proposed a carefully studied methodology to generate UAQ benchmarks starting from the known complexity results and used them to evaluate the state-of-the-art UAQ solvers. Our experimental results do not only show the effectiveness of the solvers over various UAQ problem classes but also the impact of the chosen parameter in the overall performance.

As future work we would like to apply the methodology proposed in this paper to the UAQ problem specification framework introduced in [9], using the complexity results provided in that work to guide the generation of benchmarks.

REFERENCES

[1] [n. d.]. zChaff. http://www.princeton.edu/~chaff/zchaff.html

[2] Alessandro Armando, Silvio Ranise, Fatih Turkmen, and Bruno Crispo. 2012. Efficient run-time solving of RBAC user authorization queries: pushing the envelope. In *Second ACM Conference on Data and Application Security and Privacy (CODASPY)*. 241–248.

[3] Jeremias Berg, Tuukka Korhonen, and Matti Järvisalo. 2017. Loandra: PMRES Extended with Preprocessing. In *Proc. of MaxSAT Evaluation 2017 - Solver and Benchmark Descriptions*.

[4] Liang Chen and Jason Crampton. 2009. Set covering problems in role-based access control. In *Proceedings of the 14th European conference on Research in computer security (ESORICS'09)*. 689–704.

[5] Liang Chen and Jason Crampton. 2009. Set Covering Problems in Role-Based Access Control. In *ESORICS*. 689–704.

[6] S. Du and J. B. D. Joshi. 2006. Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In *SACMAT*. 228–236.

[7] Ninghui Li, Ji-Won Byun, and Elisa Bertino. 2007. A Critique of the ANSI Standard on Role-Based Access Control. *IEEE Security & Privacy* 5, 6 (2007), 41–49.

[8] N. Li, M. V. Tripunitara, and Z. Bizri. 2007. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur.* 10 (May 2007). Issue 2.

[9] Jianfeng Lu, James B. D. Joshi, Lei Jin, and Yiding Liu. 2015. Towards complexity analysis of User Authorization Query problem in RBAC. *Computers & Security* 48 (2015), 116–130.

[10] Jianfeng Lu, Zheng Wang, Dewu Xu, Changbing Tang, and Jianmin Han. 2017. Towards an Efficient Approximate Solution for the Weighted User Authorization Query Problem. *IEICE Transactions* 100-D, 8 (2017), 1762–1769.

[11] Jianfeng Lu, Yun Xin, Zhao Zhang, Hao Peng, and Jianmin Han. 2018. Supporting user authorization queries in RBAC systems by role-permission reassignment. *Future Generation Comp. Syst.* 88 (2018), 707–717.

[12] Nima Mousavi. 2014. *Algorithmic Problems in Access Control.* Ph.D. Dissertation. University of Waterloo, Canada.

[13] Nima Mousavi and Mahesh V. Tripunitara. 2012. Mitigating the Intractability of the User Authorization Query Problem in Role-Based Access Control (RBAC). In *NSS*. 516–529.

[14] Nima Mousavi and Mahesh V. Tripunitara. 2015. Hard Instances for Verification Problems in Access Control. In *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, June 1-3, 2015.* 161–164.

[15] National Institute of Standards and Technology (NIST). 2004. Role-Based Access Control. *American National Standards Institute, Inc.* (2004).

[16] C. Sinz. 2005. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *Principles and Practice of Constraint Programming (CP)*. 827–831.

[17] Fatih Turkmen. 2012. *Exploring Dynamic Constraint Enforcement and Efficiency in Access Control.* Ph.D. Dissertation. University of Trento, Italy.

[18] G. T. Wickramaarachchi, W. H. Qardaji, and N. Li. 2009. An efficient framework for user authorization queries in RBAC systems. In *SACMAT*. 23–32.
[19] Ke Xu, Frédéric Boussemart, Fred Hemery, and Christophe Lecoutre. [n. d.]. A Simple Model to Generate Hard Satisfiable Instances. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*. 337–342.
[20] Y. Zhang and J. B. D. Joshi. 2008. UAQ: a framework for user authorization query processing in RBAC extended with hybrid hierarchy and constraints. In *SACMAT*. 83–92.