A Privacy-Preserving Approach for Composite Web Service Selection

Amine Belabed*, Esma Aïmeur, Mohammed Amine Chikh***,Fethallah Hadjila ***

* UABT University - Tlemcen, LRI, Algeria.

** Computer Science Department, University of Montreal, Canada.

*** UABT University - Tlemcen, EBM, Algeria.

E-mail: {belabed.amine,f_hadjila, mea_chikh}@mail.univ-tlemcen.dz, aimeur@iro.umontreal.ca

Received 7 May 2016; received in revised form 21 March 2017; accepted 8 May 2017

Abstract. Web services technologies are considered as the most promising technologies for the integration of applications and heterogeneous data sources. In spite of the progress achieved in this area, the issue of users' privacy protection remains a major concern for both academic and industrial communities. This paper presents a model for preserving privacy in the context of web services and demonstrates its feasibility in a web service selection scenario. In addition, we introduce three privacy based selection approaches. The first one is based on a best first search like procedure, while the two others are based on two well-known declarative AI models, namely propositional satisfiability (SAT) and answer set programming (ASP). Finally, an experimental evaluation on different types of datasets, demonstrates the effectiveness of our proposed approaches.

Keywords. Privacy, Web Services, Selection, Composition.

1 Introduction

Nowadays, Web Services play a central role in the functioning of the Internet. Thanks to their flexibility and modularity, they are used as the main technology of communication and data exchange in recent software models like cloud computing. One of the most important advantages behind using such a technology lies in its ability to compose existing web services. By doing so, a new value-added functionality is created with less effort and resources.

Due to the increasing number of services over the Internet, Web Service selection becomes a crucial task. In a nutshell, the selection task consists in choosing amongst a set of candidate services, those that best meet the users' needs. Usually, Web Services selection is based on non-functional criteria covering various categories such as Quality of Service (QoS), security or privacy. Finding a service or composition of services that best fulfill non functional requirements is known to be an NP-hard problem [1]. For this reason, the design of effective systems remains an essential research issue.

Although the majority of Web Service selection approaches are based on QoS as the main criterion of selection [1, 2, 3, 4, 5, 6], the growing concerns about data privacy has resulted in numerous studies addressing privacy issues in the area of service selection [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Despite these efforts, the concept of privacy remains a challenging problem in this field. In fact, it entails many other questions such as : i) How to represent privacy policies in order to allow users and service providers to specify their privacy needs, ii) How to efficiently handle the specified policies to find a composition that satisfies all privacy constraints, iii) If such a composition does not exist, how to ensure an acceptable level of privacy protection. Our work addresses the privacy issue in the context of Web Service selection. Our objective is to protect both users and service providers from privacy violation. The infraction can be caused by any misuse or unauthorized disclosure of exchanged data. To ensure this objective, we propose a privacy selection framework, which aims to find a composite Web Service that preserves the privacy requirements of both users and service providers. This framework takes as input a composition data exchange plan, a set of user privacy requirements as well as the privacy policies of each candidate service. The output is a concrete composition that best fulfills all input privacy constraints:
The contribution of this paper is three-fold.

- Firstly, we propose two privacy models: a privacy composition model and a privacy policy model. The privacy composition model is used to represent the data exchange between the component services. In this model, only data that is qualified as private is taken into account. On the other hand, the privacy policy model is used to express users' and services' privacy requirements and policies. In this model, privacy definition is based on four dimensions : purpose, visibility, granularity, and retention time. These four dimensions are considered as the most complete predicates for defining privacy needs [17]. Moreover, the utilization of these dimensions makes our model compatible with the w3c privacy framework standard P3P [18, 19].

- The second contribution consists in defining a fuzzy integral based data disclosure function. This function measures the amount of the risk of privacy threat caused by service providers while using private data. In our privacy model, this function is used to rank compositions that fulfill privacy requirements, from which we select the composition with minimal privacy risk.

- Based on the proposed privacy models, our third contribution offers three selection algorithms. The main objective is to demonstrate the feasibility and the compatibility of our privacy models with different types of algorithms. The first algorithm is called Constrained Best First Search (CBFS) algorithm. This approach adapts an existing best first search algorithm (BFS) to manage privacy constraints. The other algorithms are based on two of the most popular declarative models: Boolean satisfiability (SAT) and Answer Set Programming (ASP). In these last models, the specification of the problem is encoded as a propositional formula or ASP program, then a general search program called solver is used to search for solutions. The effectiveness of these algorithms is tested and compared on different types of datasets.

The remainder of this paper is structured in the following manner. Section 2 introduces some preliminary concepts and necessary definitions after which Section 3 provides a detailed description of our proposed selection privacy framework. This description includes details about the Composition Model as well as the privacy policy model. The privacy preserving Web Services selection problem formulation is presented in Section 4. Section 5 is devoted to the proposed privacy risk function along with the algorithms' details. Evaluation and experimental results are provided in Section 6 and some related work are presented in section 7. Finally, section 8 concludes and provides some perspectives.

2 Technical Background and Preliminary Definitions

This section introduces some preliminary concepts which are used throughout this paper. It defines Web Services composition and selection, as well as some basic concepts related to the Weighted Partial MaxSAT (WP-MaxSAT) and the Answer Set Programming (ASP).

2.1 Web Services Composition and Selection

A Web service is defined as a modular and self-described application invocable with standard web technologies (HTTP, SOAP, etc.)[20, 21]. In a typical web service architecture, a service provider publishes service descriptions as WSDL (Web Service Description Language) files on registries like UDDI (Universal Description Discovery and Integration) [20]. Consumers can discover services from the registry and finally invoke the discovered services. For certain types of applications, it is necessary to combine a set of Web Services (aggregate or composite Web services) to meet more complex requirements. This combination (composition or work-flow) leads to an abstract composite service that specifies a set of atomic tasks as well as data flow and control between them. This work-flow can be formalized through the use of specific languages like BPEL (Business Process Execution Language) [22]. Considering the abstract composition, the selection phase consists in searching for a set of concrete services that best fulfill non-functional requirements. In general, non-functional requirements are Quality of Service (QoS) (e.g. latency, availability, cost, etc.), security or, like in our work: privacy constraints. Throughout the rest of this paper, the two terms Web Service and service are used interchangeably.

2.2 SAT and Weighted Partial MaxSAT

A propositional (or Boolean) formula Φ is in conjunctive normal form (CNF) if it is a conjunction ($\wedge$) of clauses, where a clause is a disjunction ($\vee$) of literals. A literal can be a propositional variable p or its negation $\neg p$. A CNF formula can be seen as a set of clauses and each clause as a set of literals. We denote by $Var(\Phi)$ the set of propositional variables occurring in Φ. A truth assignment σ to a set $V \subseteq Var(\Phi)$ of propositional variables is a map $\sigma : V \rightarrow \{0, 1\}$. An assignment σ satisfies a CNF formula Φ ($\sigma(\Phi) = 1$) if it satisfies all its clauses, in this case, σ is called a model of Φ. The propositional satisfiability problem (SAT) [23] consists in deciding if a given CNF formula admits a model or not. Today, This canonical NP-Complete problem has gained a considerable audience with the advent of a new generation of solvers able to solve large CNF instances encoding real-world problems. In addition to the traditional applications of SAT to hardware and software formal verification, this impressive progress led to increasing use of SAT technology to solve new real-world applications such as planning, bioinformatics, data mining and cryptography. In the majority of these applications, we are mainly interested in the decision problem and some of its optimization variants such us Maximum satisfiability (MaxSAT), Partial MaxSAT (P-MaxSAT) or Weighted Partial MaxSAT (WP-MaxSAT). MaxSAT is defined as the problem of finding a truth assignment satisfying a maximum number of clauses. In

P-MaxSAT, given a CNF formula $\Phi_h \wedge \Phi_s$, the problem is to find a truth assignment satisfying the hard part (Φ_h), while maximizing the number of satisfied soft clauses (Φ_s). In the Weighted Partial MaxSAT problem [23], the CNF formula contains also two sets of clauses, a set of hard clauses (Φ_h) that must be satisfied and a set of weighted soft clauses (Φ_{ws}). A solution to a weighted partial MaxSAT problem consists in finding an optimal assignment that satisfies all the hard clauses (Φ_h) and maximizes the sum of the weights of the satisfied soft clauses (Φ_{ws}).

2.3 Answer Set Programming (ASP)

Answer Set Programming (ASP) [24, 25], has become one of the most popular declarative approaches to solving difficult search problems (NP-hard). ASP can be seen as a branch of knowledge representation, logic programming and (non-monotonic) reasoning. Syntactically, ASP looks like logic programming (e.g Prolog) while the semantics are based on stable models, also called answer sets (see [26]). An ASP program Π is represented as a finite set of rules. A normal rule r_i is of the form:

$$h_1 \vee \cdots \vee h_k \leftarrow a_1, ..., a_m, \neg a_{m+1}, ..., \neg a_n. \;\; (k \geqslant 0, n \geqslant m \geqslant 0)$$

where h_i, a_j are atoms (literals) in first order language and $\neg$ represents a negation-as-failure symbol. In addition, the set $H(r_i) = \{h_1, \ldots, h_k\}$ is called the head of the rule r_i, and $B(r_i) = \{a_1, a_2, ..., a_m, \neg a_{m+1}, ..., \neg a_n\}$ is the body of the rule r_i. If $B(r_i) = \emptyset$ the rule represents a fact; while if $H(r_i) = \emptyset$ the rule represents a constraint. A solution for an ASP program corresponds to its underlying answer set. Intuitively, finding the answer set is equivalent to finding the set of only the literals that are justified by a rule in the program Π. The utilization of ASP entails the specification of the problem as an ASP program, thereafter we leverage an ASP solver to handle the aforementioned specification.

3 A Privacy Based Selection Framework

3.1 Framework Architecture

In this work, we propose a privacy selection framework (Figure 1). The objective is to find a composition that satisfies privacy constraints of both users and service providers. The main component of this system is the privacy manager, which generates an executable composite Web Service (concrete composition) that best fulfills all privacy constraints of both users and service providers. To generate the concrete composition, the privacy manager uses a selection algorithm which takes as input: i) an abstract composition that specifies the Data-flow and the Control-flow of the component services (abstract tasks); ii) a set of concrete services that implement the previous abstract tasks, in addition to the specification of their privacy constraints (stored in services registry); iii) a user's request that specifies his/her privacy requirements.
The selection algorithm implemented in the privacy manager is based on two models: the composition model and the privacy model. The composition model mainly focuses on the Data-flow that links the component services, while the privacy model describes how to define the privacy rules used for expressing privacy requirements and constraints. Details about these two models are given in the next two sections.

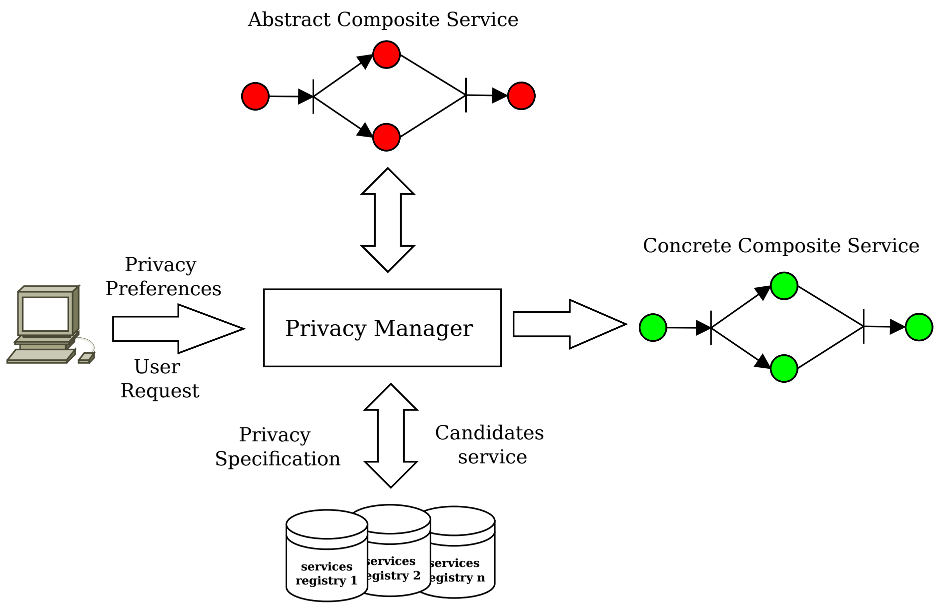


Figure 1: Privacy selection Framework

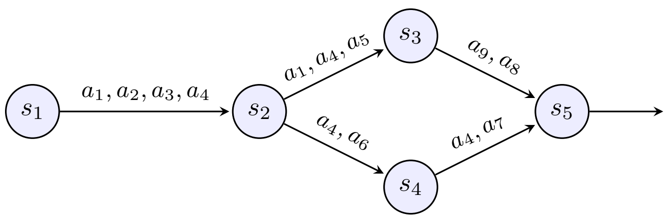


Figure 2: Example of a composition Dataflow graph

3.2 Composition Model

In our work, a composite Web Service (composition of services) C is represented by a couple $C =< D, W >$, where D represents the Data-flow model of the composition and W represents the Work-flow model. In a privacy context we are more interested in the Data-flow model which represents the data exchange plan between the elementary services of the composition. The Data-flow model is represented by a valued directed graph $D = (S, A)$. In this graph, the set S of vertices represents elementary services, while the set A of arcs represents data dependencies between services. Figure 2 represents a simple Data-flow graph. The figure shows that each service in the composition needs a set of attributes to provide the expected service. For example, the service s_3 needs the data attributes a_1,a_4 and a_5 as inputs to provide the service functionality. Note that the service s_1, which is without input attributes, represents a user agent. To ensure privacy protection, each service of the composition must comply with the privacy constraints required by the services that provide the needed attributes.

We observe in this example (see Figure 2) that the attributes a_1 and a_4 (arc (s_2,s_3)) are provided by the service s_2 whereas they are produced by the service s_1. In other words, the service s_1 is the *owner* of the attributes a_1 and a_4. Therefore, for these two attributes, the service s_3 must comply with the privacy requirements of s_1 and not those of s_2.

According to the Data-flow graph (composition graph) and the concept of *attributes owner*, we can construct a new graph called *privacy graph* which is defined as follows:

Definition 1 (Privacy graph). A privacy graph of a composition graph $D = (S, A)$ is a directed graph $P_G = (S, A_p)$, where the set of arcs A_p represents the privacy constraints related to the data attributes exchanged between services.

Figure 3 represents the privacy graph issued from the composition graph of figure 2. For example, in this figure, the arc between s_1 and s_2 means that the privacy policies of service s_2 related to the attributes a_1, a_2, a_3 and a_4 must comply with privacy requirements of service s_1.

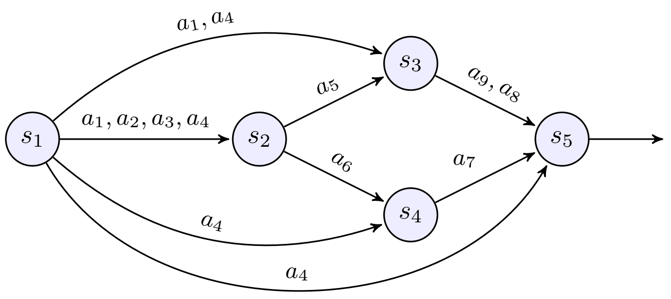


Figure 3: Privacy graph corresponding to Dataflow graph of figure 2

A privacy graph is said to be valid if all privacy constraints between the nodes (services) in this graph are satisfied.

Definition 2 (Service Precedence Set). Given a privacy graph $P_G = (S, A_p)$, the precedence set PRD^{s_x} of a service $s_x \in V$ is defined as follows:

$$PRD^{s_x} = \{s_y \in S \,|\, \exists\, (s_y, s_x) \in A_p\}$$

Example: in the privacy graph of figure 3, the precedence set of the service s_5 is: $PRD^{s_5} = \{s_1, s_3, s_4\}$.

Definition 3 (Set of Dependencies). Given a privacy graph $P_G = (S, A_p)$, the set of dependencies DEP^{s_x,s_y} between two services $s_x, s_y \in S$ is the value of the arc (s_x, s_y).

Example: in the privacy graph of figure 3, we have:
$DEP^{s_1,s_2} = \{a_1, a_2, a_3, a_4\}$, $DEP^{s_2,s_5} = \emptyset$.
The concepts of *service precedence set* and *set of dependences* are used in the *privacy model* specification, which is introduced in Section **??**.

3.2.1 Illustrative Example

Figure 4 represents a simplified scenario derived from an on-line shopping web service. This service involves three elementary services and a user agent. The elementary services

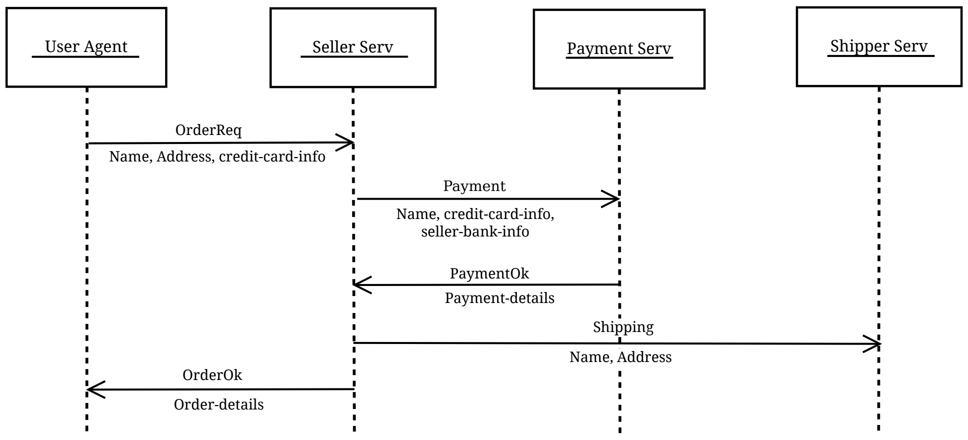


Figure 4: The scenario of on-line shopping successful order.

are: Seller, Payment and Shipper service. This scenario describes a successful on-line shopping transaction. The user agent starts the transaction by sending to the seller service an order message (*OrderReq*) along with the necessary data attributes (*Name, Address, credit-card-info*). Then, the Seller sends to the payment service a payment message (*Payment*) as well as the data attributes: *seller-banking-info*, the *Name* and the *credit-card-info* of the user. In the case of a successful payment, the payment service informs the seller by a *PaymentOk* message together with information details about the payment. After that, the seller invokes the shipper service to deliver the ordered goods. In this example, the data attributes, *Name, Address, credit-card-info* (of user) and *seller-banking-info* (of seller) are considered as private. Figure 5 shows the Data-flow graph (Figure 5 (a)) and its corresponding privacy graph (Figure 5 (b)). Note that in these graphs, only private data attributes are represented, and all other data (e.g. *Payment-details, Order-details*) are omitted. The privacy graph specifies the privacy constraints that the three services must verify to accomplish a successful transaction. For example, the privacy policies the seller service must be in accordance with the privacy requirements of the user for the data attributes, *name, address* and *credit-card-info*.

From this privacy graph we can define the *Services' precedence set* (PRD^s) and the *Sets of dependencies* ($PRED^{s_1,s_2}$) as follows:

$PRD^{User} = \emptyset$, $PRD^{Seller} = \{User\}$, $PRD^{Payment} = \{User, Seller\}$,
$PRD^{Shipper} = \{User\}$.
$DEP^{User,Seller} = \{name, address, \textit{credit-card-inf}\}$, $DEP^{Seller,Payment} = \{\textit{seller-bank-inf}\}$,
$DEP^{User,Payment} = \{address, \textit{credit-card-inf}\}$, $DEP^{User,Shipper} = \{name, address\}$.

3.3 Privacy Policy Model

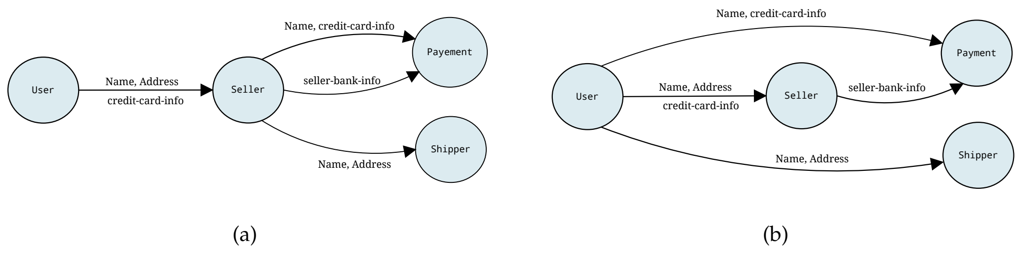


Figure 5: Dataflow graph (a) and the corresponding privacy graph (b) of composition senario of figure 4.

3.3.1 Privacy Predicates

To express privacy policies and requirements of users and service providers that correspond to each data attribute (name, date of birth, SSN, etc.), we need a set of privacy predicates (or privacy criteria) that represent as much as possible all privacy facets. In our work, we have used the same set of predicates defined in [19, 27]. These predicates represent privacy as a four-dimensional point, where each dimension represents a different privacy facet. These predicates are : *purpose*, *visibility*, *granularity* and *retention time*. They are defined as follows:

- **Purpose (Pr)**: defines how data can be used once it has been collected;
- **Visibility (V)** :defines who is allowed to see the provided data;
- **Granularity (G)** : defines how much precision of data is provided ;
- **Retention time (T)** : defines how long data is kept by the data collector.

We note that, in general, these privacy predicates are represented as lattices or hierarchy structures [19]. In our work we normalized the predicates: *visibility*, *granularity* and *retention time* to be represented as real numbers in the interval $[0..1]$. The objective of this normalization is to make privacy predicates' comparisons and calculation more easier. To explain the normalization process, we use the simple example of Figure 6. Figure 6 (a) represents the granularity hierarchy related to the marital status attribute, while Figure 6 (b) represents a simple hierarchy of visibility domain. As Figure 6 shows, the normalization consists in assigning a value between 0 and 1 to each level of the domain hierarchy. This assignment is done in such a way that the most information revealing level takes the closest value to 1. The normalized value is obtained by using the following formula:

$$N_p = \frac{D_{max} - d_p}{D_{max}} \tag{1}$$

Where D_{max} is the maximum depth of the predicate's domain hierarchy and d_p is the predicate level.
In the example of Figure 6 (a), if a divorced user does not want to reveal the exact value of his/her marital status but just that he/she is once married, the normalized value of the marital status granularity will be 0.5 ($D_{max} = 2$ and $d_p = 1$). Similarly, and according to

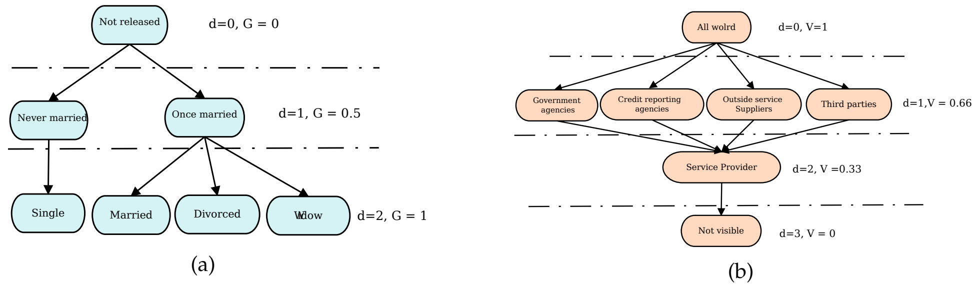


Figure 6: Example of (a)granularity and (b) visibility domain normalisation

the visibility hierarchy of Figure 6 (b), if a user wants to make a given attribute visible only to the service provider, the visibility value related to that attribute will be 0.33($D_{max} = 3$ and $d_p = 2$).
The normalization of the retention time predicate is easier as we just divide the value of the attribute retention time by the maximum value. We note that, to prevent users from handling numerical values, they can express how much information may be revealed through a GUI. After that, the mapping to the interval [0,1] will be done by the selection framework.

3.3.2 Privacy Rules

A privacy rule is an assertion used by users or service providers to specify their privacy policy and requirements. A privacy rule is defined by the aforementioned privacy predicates: Purpose (*Pr*), Visibility (*V*), Granularity (*G*) and Retention time (*T*). More formally, a privacy Rule R_i is a quintuplet: $R_i = \langle a_i, Pr_i, V_i, G_i, T_i \rangle$, where:

- $a_i \in Att$, is a private data attribute ex: Name , SSN , age, address , etc.
- Pr_i, V_i, G_i and T_i represent the values of the privacy dimensions: Purpose, Visibility, Granularity and Retention time.

A privacy rule $R_i = \langle a_i, Pr_i, V_i, G_i, T_i \rangle$ means that the private data a_i was collected for the purpose Pr_i , with a visibility V_i , a precision G_i and retention time T_i. For a given privacy rule $R_i = \langle a_i, Pr_i, V_i, G_i, T_i \rangle$, we assume the following syntax: $R_i[Att] = a_i, R_i[Pr] = Pr_i, R_i[V] = V_i, R_i[G] = G_i, R_i[T] = T_i$.

Definition 4 (well defined rules). A set of privacy rules P is said to be well defined if for all rules in P we can not find two rules with the same values of data attribute, purpose and visibility but different values of granularity and/or retention time. Formally:

$$\forall R_i, R_j \in P : \begin{cases} (R_i[Att] = R_j[Att]) \wedge \\ (R_i[Pr] = R_j[Pr]) \wedge \\ (R_i[V] = R_j[V]) \Rightarrow \\ (R_i[G] = R_j[G]) \wedge (R_i[T] = R_j[T]) \end{cases} \tag{2}$$

Intuitively, a privacy rule R_i is characterized by an identifier which consists of the structure (a_i, Pr_i, V_i) (this is similar to the primary key in relational data bases). Thus, any rule that contains a triplet (a_i, Pr_i, V_i) must be unique. Otherwise, we will get an inconsistent

set of privacy rules. For example, in the privacy graph of Figure 5(b), if a user defines a set of rules that contains the two following privacy rules, then that set is not well defined (inconsistent):
$R_1 = \langle name, shopping\_purpose, service\_provider, all\_released, 60\rangle$;
$R_2 = \langle name, shopping\_purpose, service\_provider, all\_released, 50\rangle$.

3.3.3 Privacy Policies and Privacy Requirements

Privacy Policy (PP): (noted as PP) is the set of *well defined privacy rules*, specifying the set of privacy practices applicable on any collected data. Formally a privacy policy PP^{s_x} of a service s_x is defined as follows:

$$PP^{s_x} = \{R_i \in P \mid R_i[Att] \in I^{s_x}\}. \text{ Where, } I^{s_x} \subseteq \{Input\ of\ s_x\} .$$

For example, the seller service in the privacy graph of Figure 5 (b), has:
$I^{seller} \subseteq$ {*name, address, credit-card-inf*}. Thus, the seller service must specify a set of privacy rules PP^{seller}, that defines the usage of the attributes belonging to I^{seller}. For example, if the seller service wants that: i) all parts of the user $address$ must be released (G), ii) the $address$ is assigned to the purposes of: $shopping$ and $statistics$, iii) the $address$ is shared with third parties, iv), and the $address$ is retained for 90 days, then, the set PP^{seller} must contain the following privacy rules:
$R_1 = \langle address, shopping\_purpose, third\_parties, all\_released, 90\rangle$;
$R_2 = \langle name, statistics\_purpose, third\_parties, all\_released, 90\rangle$.

Privacy Requirement (PR): (noted as PR) is the set of *well defined privacy rules*, specifying the set of privacy conditions that a third-party service must meet to consume provided data. Formally a Privacy requirement, PR^{s_x}, of a service s_x is defined as follows:

$$PR^{s_x} = \{R_i \in P | R_i[Att] \in O^{s_x}\}. \text{ Where, } O^{s_x} \subseteq \{Output\ of\ s_x\} .$$

In the privacy graph of Figure 5(b), the seller service has: $O^{seller} \subseteq$ {*seller-bank-info*}. If the seller service is willing to reveal all parts of the attribute *seller-bank-info*, only to the payment service and only for payment purposes, and requires that this attribute will not be retained more than one day, then, the set PR^{seller} must contain the privacy rule:
$R = \langle seller\text{-}bank\text{-}info, payment\_purpose, service\_provider, all\_released, 1\rangle$;

3.3.4 Comparable Rules

Two privacy rules are comparable, if they are associated with the same attribute and their purposes belong to the same set. Formally, the two rules R_i and R_j are comparable if:

$$(R_i[Att] = R_j[Att]) \wedge (R_i[Pr], R_j[Pr]) \in Gl \times Gl.$$

where, $Gl \subseteq Pr$, represents the set of all goals of a given composition.
For example, in the on-line shopping web service of Figure 4, the set Gl can be :
$Gl = \{shopping\_purpose, payment\_purpose, shipping\_purpose\}$. If the user defines the privacy requirement rule $R_1 = \langle name, Shopping\_purpose, 0.5, 0.33, 0.2\rangle$, and the seller service, defines the privacy policy rule $R_2 = \langle name, statistic\_purpose, 0.6, 0.2, 0.2\rangle$. The two rules R_1 and R_2 are incomparable, since, $statistic\_purpose \notin Gl$. Consequently, we can state that these two rules are not compliant without comparing the values of the other predicates.

We define the function $comp(R_i, R_j)$ that returns 1 if the pair of rules (R_i, R_j) are comparable.

$$comp(R_i, R_j) = \begin{cases} 1 & \text{if } (R_i[Att] = R_j[Att]) \wedge (R_i[Pr], R_j[Pr]) \in Gl \times Gl \\ 0 & \text{else} \end{cases} \tag{3}$$

3.3.5 Rules Compliance

A privacy rule R_i is compliant with the privacy rule R_j ($R_i \sim R_j$) if:

1. These two rules are comparable;
2. The values: visibility, granularity and retention time of R_i are greater or equal than of those of R_j.

Formally:

$$R_i \sim R_j \Leftrightarrow \begin{cases} comp(R_i, R_j) = 1 & \wedge \\ R_i[V] \geqslant R_j[V] & \wedge \\ R_i[G] \geqslant R_j[G] & \wedge \\ R_i[T] \geqslant R_j[T] \end{cases} \tag{4}$$

3.3.6 Services Compliance

A service s_x that defines a privacy requirement $PR^{s_{xy}}$ complies with a service s_y that defines a privacy policy $PP^{s_{xy}}$, if the function $Nconf(s_x, s_y)$ returns zero. This function represents the number of non-compliant privacy rules between the two services s_x and s_y. This function is defined as follows:

$$Nconf(s_x, s_y) = \begin{cases} 0 & \text{if } \forall R_i \in PR^{s_{xy}}, \exists R_j \in PP^{s_{xy}} : R_i \sim R_j \\ k & \text{else }, (k = |NC|) \end{cases} \tag{5}$$

Where, $NC = \{R_i \in PR^{s_{xy}} | \nexists R_j \in PP^{s_{xy}} : R_i \sim R_j\}$. $PR^{s_{xy}} \subseteq PR^{s_x}$, $PP^{s_{xy}} \subseteq PP^{s_y}$ and defined as : $PR^{s_{xy}} = \{R_i \in PR^{s_x} | R_i[Att] \in DEP^{s_x, s_y}\}$, $PP^{s_{xy}} = \{R_i \in PP^{s_y} | R_i[Att] \in DEP^{s_x, s_y}\}$, where, PR^{s_x} and PP^{s_y} represent respectively all privacy requirements and privacy policies defined by the services s_x and s_y, while DEP^{s_x, s_y} represents the set of dependencies between the services s_x and s_y (see definition 3).

3.3.7 Valid Service

A service s_x is said to be valid, if it is compliant with all services on which it depends. Formally:

$$vld(s_x) = \begin{cases} 1 & \text{if } \forall s_y \in PRD^{s_x} : Nconf(s_y, s_x) = 0 \\ 0 & \text{else} \end{cases} \tag{6}$$

where, PRD^{s_x} represents the precedences set of s_x (see definition 2).

3.3.8 Valid Composition

A composition $C = \{s_1, s_{2,...,} s_n\}$ is said to be valid if all its component services are valid. We define the function $vld(C)$, which returns 1 if the composition C is valid:

$$vld(C) = \begin{cases} 1 & \text{If } \forall s_x \in C : vld(s_x) = 1 \\ 0 & \text{else} \end{cases} \tag{7}$$

3.3.9 Privacy Risk Function

As the objective of our selection framework is to find a composition that preserves all privacy constraints and minimizes the risk of a privacy threat, we need to define a function that measures such a risk. This function will be used to rank compositions that fulfill privacy requirements to select the composition with the minimal privacy risk.

We notice that our model imposes strong constraints on the purpose predicate, as mentioned earlier, a rule R_i is not compliant with a rule R_j if the two purposes defined in these two rules don't belong to the same composition purposes set Gl. Therefore, the privacy risk function is expressed only by the three privacy predicates: *visibility*, *granularity* and *retention time*. To define the privacy risk induced by the overall composition, we have first to specify the privacy risk function for each service.

Service Privacy Risk To define the privacy risk function of a service s_x, we have to define a function that aggregates the values of the three privacy predicates V,G and T of each rule of the PP^{s_x} set. With the existence of such a function, the risk of a privacy threat induced by a web service s_x in a composition is expressed by:

$$risk(s_x) = \frac{1}{|PP|} \sum_{R_{i=1}}^{R_{i=|PP|}} \lambda_i \times Agr(R_i[V], R_i[G], R_i[T]) \tag{8}$$

where, Agr represents the aggregation function, and λ_i represents the sensitivity degree of the data attribute $R_i[Att] = a_i$ of the rule R_i. Note that the sensitivity degree is defined by the data attribute owner (users or service providers).
The aggregation function used in our work is defined in section **??**.

Composition Privacy Risk Given a composition of n services $C = \{s_1, s_2, ..., s_n\}$, the overall privacy risk of this composition is obtained by aggregating the risks of its component services. The aggregation depends on the Work-flow model (execution plan) of the composition. Table 1 introduces some examples of such aggregations.

Table 1: Example of composition privacy risk aggregation function.

execution plan	**aggregation function**
Sequential	$\sum_{i=1}^{n} risk(s_i)$
Parallel	$\sum_{i=1}^{n} risk(s_i)$
Loop	0
Conditional	$\sum_{i=1}^{n} p_i \times risk(s_i)$

As mentioned in Table 1, the execution plan structure of the composition (Sequential, parallel, loop, conditional) doesn't have a large influence on the calculation of the overall privacy risk. In most cases, the aggregation function is the sum of the privacy risks introduced by the component services. The exception is made for the conditional structure, where only one service runs among a set (according to the value of a condition). In this case, one can assess the privacy risk introduced by this structure as the weighted sum of services' privacy risk, where the coefficients represent the participation probability of each service in the execution. In a loop structure, the execution of the same service more than once will not introduce an additional privacy risk. As previously mentioned, this function measures the rate of potential privacy risk. Consequently, the higher the privacy risk function is, the smaller privacy protection becomes.

4 Privacy Preserving Web Services Selection Problem (PP-WSP)

4.1 Problem Formulation

Given an abstract composition $C_A = \{S_1, S_2, ..., S_n\}$ represented as a privacy graph, a list of candidate services for each class S_i in C_A and a user privacy requirements U_{req} (set of privacy rules). The objective is to find a concrete composition $C_c = \{s_1, s_2, ..., s_n\}$, by binding each $S_i \in C_A$ to a concrete service $s_i \in C_c$ such that:

- The composition $\{U_{req}\} \cup C_c$ is valid.
- The privacy risk function is minimized.

According to the aforementioned privacy model, we have to find a concrete composition C_c, with:

$$C_c = \{s_1, s_2, ..., s_n\} : \begin{cases} vld(\{U_{req}\} \cup C_c) = 1. \\ minimise(risk(C_c)). \end{cases} \tag{9}$$

4.2 Complexity of Privacy Preserving Web Services Selection Problem

The general privacy preserving web services selection problem (PPWSP) with n service classes and k candidate services per class, can be formally represented as follows:

$$\begin{cases} \text{minimize } \sum_{i=1}^{n} \sum_{j=1}^{k} s_{ij} r_{ij} \\ \text{subject to } \sum_{i=1}^{n} \sum_{j=1}^{k} s_{ij} v_{ij} \leq V \\ \sum_{i=1}^{k} s_{ij} = 1 \text{ , } 1 \leq i \leq n \\ s_{ij} \in \{0, 1\} \text{ , } 1 \leq i \leq n, \ 1 \leq j \leq k \end{cases} \tag{10}$$

where: s_{ij} is the service j of the class i; r_{ij} represents the privacy risk associated with s_{ij}; v_{ij} represents the number of non-compliant rules of the service s_{ij} and V represents the number of non-compliant rules tolerated in the solution. Note that the problem statement in equation 9 is a special case of PPWSP (represented in equation 10), where $V = 0$.
The PPWSP is an NP-hard problem. The proof is done by reducing the Multiple Choice Knapsack Problem (MCKP) which is known to be NP-hard [28] to PPWSP.

The MCKP is defined as follows: Given N item groups $(G_1, \cdots, G_N)$, each of them contains k items, ie: $G_j = \{I_{1j}, \cdots, I_{kj}\}$ and a knapsack of capacity C. Each item has a weight w_{ij} and profit p_{ij}. The MCKP consists of selecting one item from each group to be placed in the knapsack so that the total profit is maximized while the total weight is less than the capacity C of the knapsack.
We can reduce MCKP to our privacy preserving web services selection problem by proceeding as follows:

- each item group G_j is mapped to a service class S_j;
- each item I_{ij} is mapped to a candidate service s_{ij} ;
- the profit p_{ij} of each item I_{ij} is mapped to $(1 - r_{ij})$, where r_{ij} represents the privacy risk of the service s_{ij};
- the weight w_{ij} of each item I_{ij} will correspond to v_{ij}, the number of non-compliant rules of the service s_{ij};
- the capacity C of the knapsack will correspond to V, the number of non-compliant rules tolerated in the solution .

After the mapping, the MCKP is represented as follows:

$$\begin{cases} \text{maximize } \sum_{i=1}^{n}\sum_{j=1}^{k} s_{ij}(1-r_{ij}) \\ \text{subject to } \sum_{i=1}^{n}\sum_{j=1}^{k} s_{ij}v_{ij} \leq V \\ \sum_{i=1}^{k} s_{ij} = 1 \text{ , } 1 \leq i \leq n \\ s_{ij} \in \{0,1\} \text{ , } 1 \leq i \leq n, \ 1 \leq j \leq k \end{cases} \tag{11}$$

The formula: maximize $\sum_{i=1}^{n}\sum_{j=1}^{k} s_{ij}(1-r_{ij})$ of equation 11 is equivalent to: minimize $\sum_{i=1}^{n}\sum_{j=1}^{k} s_{ij}r_{ij}$. If we rewrite the equation 11 using the new equivalent formula, this equation will be similar to equation 10. Therefore, every solution to MCKP is also a solution for PPWSP and vis versa . As MCKP is known to be an NP-hard problem, then PPWSP is also NP-hard. □

4.3 Representation as a Multilevel Graph

To find a solution, we represent this problem as a multilevel graph that contains a source and sink nodes. The source node represents user request, while the sink node is added to be connected with all services of the last level. In this graph, the nodes of each level represent candidates service of a given class, while the arcs represent the amount of privacy risk induced by each service. Figure 7 gives an example of this representation. Part (a) of this figure shows a simple privacy graph that represents the data interaction between the abstract services, whereas part (b) depicts the relating multilevel graph. According to this representation, the issue of retrieving a solution is equivalent to finding the shortest path from the *Request* node to the *Sink* node, in such a way all services of this path are valid. It is important to note that the multilevel graph representation is not a complete representation. In fact, finding a solution using this representation refers always to the corresponding privacy graph to verify the validity constraints.

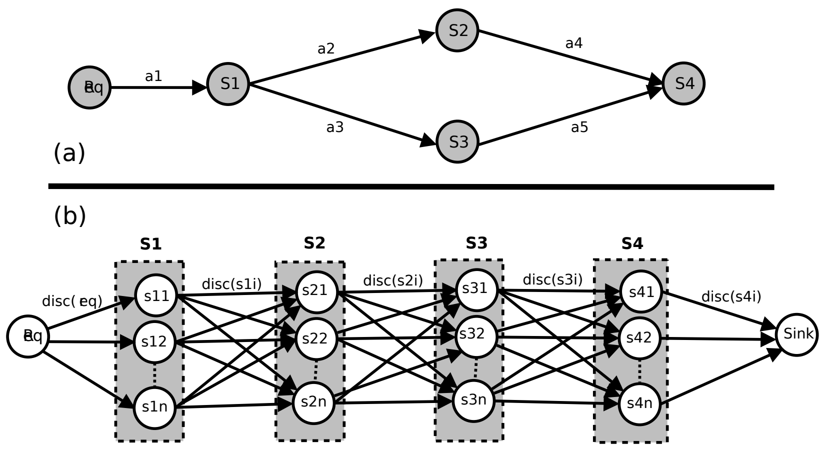


Figure 7: Multi level graph representation

5 Solution Implementation

This section presents an overview of the proposed solution for our privacy Web Service selection problem. First, we discuss the motivation behind the choice of "Choquet integral" as a mean for calculating the services' privacy risk. Then, we describe the details of the proposed algorithms.

5.1 Choquet Integral as a Privacy Risk Function

To define the privacy risk function introduced in section 5.1, we have to use an aggregate function that best represents the three privacy predicates (V,G and T). In our work we opt for a *fuzzy integral* [29] aggregation function, more precisely a *Choquet integral* [29, 30]. This choice is justified by several reasons: a Choquet integral is an aggregation operator that generalizes the traditional operators such as arithmetic mean and its variants. It also takes into account all interactions and dependencies that may exist between the different criteria unlike other classical aggregation operators [30]. To use a Choquet integral operator we have to define its associated capacity function (fuzzy measure). The following definitions introduce the concepts of capacity and Choquet integral:

Definition 5.1. Capacity: given, $N = \{1, ..., n\}$ a set of criteria. a capacity on N is a set function $\mu : 2^N \to [0, 1]$, satisfying :

- $\mu(N) = 1, \mu(\emptyset) = 0$,
- $\forall A, B \in 2^N, [A \subseteq B \Rightarrow \mu(A) \leq \mu(B)]$ (the monotony property).

Definition 5.2. The Choquet integral of a function $a : N \to R$ represented by the vector $(a_1, a_2, ..., a_n)$ w.r.t. a capacity μ on N is defined by:

$$C_\mu(a) = a_{\sigma(1)}\mu(a) + \sum_{i=2}^{n}(a_{\sigma(i)} - a_{\sigma(i-1)})\mu(\{a_{\sigma(i)}, ..., a_{\sigma(n)}\})$$

where σ is a permutation on N such that : $a_{\sigma(1)} \leqslant a_{\sigma(2)} \leqslant ... \leqslant a_{\sigma(n-1)} \leqslant a_{\sigma(n)}$. Also, $a_{\sigma(i)} := \{\sigma(i), ..., \sigma(n)\}$, for all $i \in \{1, ..., n\}$, and $a_{\sigma(i)} := \emptyset$.

5.1.1 Fuzzy based Capacity Identification

A capacity can be seen as a measure that defines the importance of each subset of criteria (in our case the privacy predicates) in an aggregation scenario. Its identification is more complex than a simple attribution of weight to different criteria. The authors in [31] have defined many methods to identify the capacity function. All these methods assume either the existence of an order between groups of criteria values, which is defined by an expert, or a small learning dataset that facilitates the identification of the weights. In our model, establishing an order based on the privacy criteria values is not evident, since the privacy criteria take continuous values in the interval [0,1]. Instead, we have used a set of fuzzy rules supposed to be given by an expert. These rules provide a fuzzy estimation of the privacy risk (high, medium, small) based on the fuzzy values of the privacy criteria. By using these rules, we implement a fuzzy system, which takes as input a set of randomly generated values of privacy criteria : V, G and T, and produces as output the associated privacy risk value, which also represents the Choquet integral function. This step results in a dataset of the form: $(V, G, T, Risk)$, which is used to calculate the capacity function. The identification of the capacity in this step is done by using the least squares approach introduced in [31]. The use of a fuzzy approach to identify the capacity function has several advantages such as being better interpretable than a simple order provided by an expert, moreover, the set of rules can be provided and revised by several experts and it is easily maintained and updated. The following rules sketch an example of the used fuzzy rules :

- `IF V IS high AND G IS high AND T IS high THEN Risk IS high;`
- `IF V IS high AND G IS high AND T IS low THEN Risk IS medium;`
- `IF G IS low AND V IS low AND (T IS low OR T IS medium) THEN Risk IS low;`

To implement the fuzzy system we utilize the Java library jFuzzyLogic [32] and for the capacity identification we use the Kappalab toolbox [31], which is a package for the GNU R statistical system [33]. The obtained capacity is described in table 2.

This table shows the weights of the privacy predicates subsets. From this fuzzy measure,

Table 2: Capacity values.

Privacy predicates	$\{\}$	$\{V\}$	$\{G\}$	$\{T\}$	$\{V, G\}$	$\{V, T\}$	$\{G, T\}$	$\{V, G, T\}$
Capacity value	0.0	0.1157	0.251	0.0066	0.5578	0.4356	0.6122	1.0

we can derive some indices (e.g. Shapley index, interaction index) that enable to interpret various parameters like, the importance and the interactions between the privacy predicates (for more details, see [30]).

5.2 Proposed Algorithms

This section describes the proposed algorithms used to solve our selection problem. First an adaptation of a Best first Search algorithm is discussed, than we detail two declarative approaches : the MaxSAT approach and the ASP approach.

5.2.1 Constrained Best First Search Algorithm (CBFS)

Using the multilevel graph representation described in Section 6.2, a solution to our problem consists of finding the shortest path from the request node to the sink node, this path must involve only valid services. The most trivial approach is to adapt an efficient path-finding algorithm to deal with our validity constraints. As our objective is to get an optimal solution, non-exhaustive algorithms (Best First Search algorithms like A* [34] and its variants [35]) that strategically prune the search space through heuristics, are good candidates. In general a BFS algorithm maintains two lists of nodes, an Open list and a Closed list. The Closed list is used to stock the already explored nodes, whereas the Open list is a priority queue that orders the unexplored nodes by a cost function $f(s) = g(s) + h(s)$. The lower the cost $f(s)$ of a given node s the higher its priority. In this cost function, $g(s)$ represents the cost of getting from the initial node to the node s, and $h(s)$ represents an estimation heuristic of the cost to reach the goal node from node s. An admissible heuristic (i.e. never overestimate the real cost) provides a guarantee of optimality for such algorithms. In this work we used the same kind of BFS algorithm, with an adaptation for managing the validity constraints. To get an admissible heuristic we relax the validity constraints of our problem, so that the heuristic value of a node s_{ij} (which represents service j of class i) is equal to the shortest path from this node to the sink node without considering the validity constraints. In other words, the composition represented by the path from the node s_{ij} to the sink node can be invalid. Note that this heuristic is easy to calculate if the services of each class are sorted in ascending order of their privacy risk values. In this case the heuristic of the node s_{ij} is calculated as follows:

$$h(s_{ij}) = risk(s_{ij}) + \sum_{k=i+1}^{n} risk(s_{1k}) \tag{12}$$

Where n is the number of classes.

For example, in the representation of figure 7 (b), if we suppose that the services of all classes are sorted, the heuristic of the service s_{22} is $h(s_{22}) = risk(s_{22}) + risk(s_{13}) + risk(s_{14})$.

The listing 1 describes the main steps of our algorithm.

Algorithm 1: Constrained Best First Search Algorithm

```
Input  : $Base(S, PP, PR, Req)$,
         $S = \{S_i\}$ , $|S|$ = number of classes;
         $S_i = \{s_{ij}\}$: Set of candidate services of class $S_i$;
         $PP = \{PP_{s_{ij}}\}$ :Set of privacy policies of all services;
         $PR = \{PR_{s_{ij}}\}$ :Set of privacy requirements of all services;
         $Req = (PR_{req}, PP_{req})$: Policies and requirements of user request;
Output: Optimal composition: $OptComp$
/* calculate heuristic value for each services                        */
1  for $i \leftarrow 1$ to $|S|$ do
2      $S_i \leftarrow S[i]$
3      for $j \leftarrow 1$ to $|S_i|$ do
4          $s_{ij} \leftarrow S_i[j]$
5          $h \leftarrow calculateHeuristic(s_{ij})$
6          $setHeuristic(h, s_{ij})$
7      end
8  end
/* Search for the solution                                            */
9  $OpenList.Add(Req)$ ;                                // a priority queue
10 while $|OpenList| > 0$ do
11     $CurrentService \leftarrow OpenLis.firstElement()$
12     if isGoal($CurrentService$) then
13         return $OptComp \leftarrow path(CurrentService)$
14     end
15     $OpenList.remove(CurrentService)$
16     $NextLevelList \leftarrow generateNext(CurrentService)$
17     foreach $s_i$ in $NextLevelList$ do
18         $DistanceFromStart \leftarrow$
           $getRisk(CurrentService) + getDistanceFromStart(CurrentService)$
19         $setDistanceFromStart(DistanceFromStart, S_i)$
20         $OpenList.Add(s_i)$
21     end
22 end
23 return solution does not exist
```

This algorithm takes as input a multi level graph representation as a data structure($Base(S, PP, PR, Req)$) and returns an optimal private composition ($OptComp$) if it exists. In the algorithm cited above, we presume that the privacy risk relative to each service is already computed and that all services of each class are sorted in ascendant order of their privacy risk value. The algorithm first calculates the heuristic value for each service (lines 1-8). Then, it begins the search procedure by inserting the request node in the priority queue (line 9). Lines from 10 to 22 show the steps of the main loop of the search procedure which aims to get the node with the lowest $f(s)$ (line 11). If this node ($CurrentService$) is a goal node, then the algorithm returns the corresponding path (lines 12-14). If this is not the case, the $CurrentService$ will be removed from the queue (line 15), and the algorithm generates the corresponding next level of services (neighbors of $CurrentService$) (line 16). This generation function is what makes the difference between the proposed algorithm and the other BFS algorithms. This is because our algorithm generates only the services that are compliant with the services composing the path from the request node to the $CurrentService$ node. This means, that we have a smaller number of generated nodes, thus, the performance is improved. Thereafter, for each generated service s', the algorithm calculates the distance that separate s' and the request node, then it adds s' to the priority queue (lines 17-21). The algorithm continues until it finds a solution or the queue is empty.

5.2.2 Max-SAT based Approach

This approach consists in encoding the PPWSP problem as a partial weighted Max-SAT (PWMSAT) instance. Then, a Max-SAT solver will be called to find the optimal solution. Partial weighted Max-SAT involves two weighted CNF formulas that represent the hard and the soft clauses. The objective is to find a variable assignment that satisfies all hard clauses while maximizing the total weight of satisfiable soft clauses. To encode the privacy selection problem as a PWMSAT we proceeded as follows:

1. given an abstract composition $S = \{S_0, S_1, \cdots S_n\}$, each concrete service $s_{ij} \in S_i$ is represented by a boolean variable x_{ij}.

2. the fact of selecting only one service s_{ij} from each class S_i is represented by the special type of cardinality constraints, often called *exactly-one constraint* and represented by $\sum_{j=1}^{|S_i|} x_{ij} = 1$. To encode this constraint we used the Commander-Variable Encoding proposed by Klieber and Kwon [36]. This encoding consists in divide the variables of each abstract class into m disjoint groups: $G_1 \ldots G_m$. In each group a commander variable c_i is introduced. The commander variable is to be true if (at least) one of the variables in its group is true; otherwise it is to be false. Thus, for each class of services S_i ($|S_i| = n$), we have the following CNF formulas (as described in [36]):

   - At most one variable in a group G_k can be true:
   $$\bigwedge_{x_{ij} \in G_k} \bigwedge_{x_{ij'} \in G_k, j' < j} \neg x_{ij} \vee x_{ij'} \tag{13}$$
   - If the commander variable of a group G_k is true, then at least one of the variables in the group must be true:
   $$\neg c_k \vee \bigwedge_{x_{ij} \in G_k} x_{ij} \tag{14}$$
   - If the commander variable of a group G_k is false, then none of the variables in the group can be true:
   $$\bigwedge_{x_{ij} \in G_k} (c_k \vee \neg x_{ij}) \tag{15}$$
   - Exactly one of the commander variables is true, which is encoded by a recursive application of the commander method.

3. the validity constraint which requires that all services of a given composition must be valid (see equation 7) is encoded as:

   $\forall S_i, S_{i'} \in S$, for all services $s_{ij} \in S_i$ and services $s_{i'j'} \in S_{i'}$, if:
   $(s_{i'j'} \in PRD^{s_{ij}} \wedge Nconf(s_{i'j'}, s_{ij}) = 1)$ (see equations 6 and 7) we add the constraint

specifying that the services s_{ij} and $s_{i'j'}$ can not be in the same composition. This constraint is expressed by $(x_{ij} \Rightarrow \neg x_{i'j'} \wedge x_{i'j'} \Rightarrow \neg x_{ij})$ which is equivalent to the clause:

$$\neg x_{ij} \vee \neg x_{i'j'} \tag{16}$$

4. the hard clauses are represented by the clauses used to encode the cardinality constraint ($\sum_{j=1}^{|S_i|} x_{ij} = 1$) along with the validity constraint (formula 16).
5. the soft clauses are represented by the atomic formulas x_{ij} weighted by the value of $(1 - risk(s_{ij}))$. The maximization of the total weights $(1 - risk(s_{ij}))$ is equivalent to minimize the total value of $risk(s_{ij})$, consequently, the corresponding solution will have a minimum privacy risk.

To analyze the performance of our encoding in terms of the many variables and clauses produced, let us consider that we have k abstract classes and n services in each class.

- *Number of variables*: The fact that each service is represented by a Boolean variables, requires $n * k$ variables. In addition, the commander encoding of the cardinality constraint produces $\frac{n}{2}$ extra variables for each class (see [36]). Thus, a total of $k * \frac{n}{2}$ extra variables. Consequently, the overall number of variables required to encode an instance of the PPWSP problem is:
$$\frac{3}{2} * n * k$$
.
- *Number of clauses:* The total number of clauses produced by the proposed encoding equals to the number of the hard clauses plus the number of the soft clauses. For the hard clauses, the commander encoding of the cardinality constraint (with a group size set to 3) produces $\frac{7}{2} * n$ clauses per class(see [36]). Thus, a total of $\frac{7}{2} * n * k$ clauses. On the other hand, in the worst case, the encoding of the validity constraint produces about $n^2 * (k-1)$ binary clauses for each class (if all services are not compliant). So, a total number of $n^2 * (k-1) * k$ clauses. The encoding of the soft clauses requires $n * k$ unary clauses. Consequently, the overall number of clauses used to encode an instance of the PPWSP problem is:
$$\begin{aligned} &\frac{7}{2} * n * k + n^2 * (k-1) * k + n * k \\ &= n^2 * (k^2 - k) + \frac{9}{2} * n * k \end{aligned}$$

5.2.3 ASP-based Approach

This approach consists in encoding the PPWSP as an ASP program, and then, we use an answer set solver to find a solution. In general, an ASP encoding consists of two main parts: a *Generate* part (or a guess part) and a *Test* part (or check part)[37]. The *Generate* part defines rules or facts that generate potential stable model candidates, typically through non-deterministic constructs, whereas, the *Test* part corresponds to the definition of the problem's constraints, and also eliminates invalid candidates. Other encoding parts can also exist, like a *Define* part that defines auxiliary predicates, or an *Optimization* part in the case of an optimization problem. To encode our problem, we follow this paradigm and we proceed as follows:

1. **generate part** : this part consists of a set of predicates that generate all possible services of the problem instance. The used predicates are:

   ```
   class(1..m).
   serviceNb(0..n).
   {serviceId(I,J)} :- class(J),serviceNb(I).
   ```

   The predicates `class/1` and `servicenb/1` define respectively the number of classes and the number of services per class of the problem instance. The rule with the head `{serviceId(I,J)}` generates the set of all possible `serviceId(I,J)` (the answers set). This predicate (ie. `serviceId(I,J)`) represents the service number `J` of the class `I`.

2. **define part** : This part defines all auxiliary predicates used in the definition of the problem instance. The used predicates are specified below:

   ```
   risk(I,J,R).
   pp(I,J,Att,Pr,V,G,T).
   pr(I,J,Att,Pr,V,G,T).
   depend(I,K).
   conforme(S1,I,S2,K):-serviceId(S1,I),serviceId(S2,K),not depend(I,K).
   conforme(S1,I,S2,K):-serviceId(S1,I),serviceId(S2,K),depend(I,K),
                        pr(S1,I,A,Pr1,V1,G1,T1),pp(S2,K,A,Pr2,V2,G2,T1),
                        Pr1=Pr2,V1 >= V2 , G1 >= G2,T >= T1.
   ```

   The predicate `risk/3` is used to define the privacy risk `R` relative to the service `J` of class `I`. The two predicates `pp/7` and `pr/7` define respectively the privacy policies and the privacy requirements of the service `J` of class `I`. Thus, the predicate `pp(I,J,Att,Pr, V,G,T).` (resp. `pr(I,J,Att,Pr,V,G,T).`) defines the values of the privacy dimensions: Purpose (`Pr`),Visibility (`V`), Granularity (`G`) and Retention time (`T`) relative to the private attribute `Att`. The predicate `depend/3` specifies if a relation exist between the tow classes `I` and `K`. This predicate is subsequently used in the predicate `conforme/4` to implement the compliance rule between two services. So, tow services `S1` (of the class `I`) and `S2` (of the class `K`) are compliant (or conform) if the predicate `conforme/4` is true. This predicate is true if one of the two situations holds: a) if no relation exist between the abstract services (or classes) in which the two services belong, b) if the two relative services are compliant according to the equation (4) of the privacy model.

3. **test part** : this part consists of rules that represent the constraints of our problem. In our encoding we used the two following constraints:

   ```
   :- I= 0..m, not 1 { serviceId(I,J)} 1.
   :- serviceId(I,S1),serviceId(K,S2), not conforme(S1,I,S2,K).
   ```

   The first constraint states that a stable model can not contain more than one fact of type `serviceId(I,_)` for a given class `I`. In other words, this rule ensures that the solution must contain only one service from each class. The second constraint says that if two services are not compliant, then, they can note be in the same stable model.

4. **optimization part** : This part is a directive that instructs the ASP solver to compute the optimal stable model. In our encoding we use the following statement:

```
#minimize { R,I,J : serviceId(I,J),risk(I,J,R) }.
```

this statement minimizes the sum of the privacy risk values associated to each service of the stable models.

6 Evaluation

From a privacy preserving point of view, all the proposed algorithms provide (by design) the best composition that fulfill all users and service providers privacy constraints. Thus, our evaluation mainly affects the effectiveness and the scalability of each algorithm. To evaluate the proposed solutions we use a randomly generated dataset, this is principally due to the lack of suitable benchmark [1]. The generation process is based on several works [38, 39, 40, 41, 42]. The latters argue that the majority of Web Services networks have the characteristics of small-world [43] or/and scale-free networks[44]. Therefore, we generated tow types of datasets: small-word dataset and scale-free dataset. The privacy graphs of these datasets were generated using networkx tool [45], in which the small-word model is based on Watts-Strogatz graphs [43], while the scale-free model is based on Bollobas et al. algorithm [44]. Table 3 describes the generated datasets in terms of composition size, and the total number of privacy rules (privacy policies rules (PP), and privacy requirements rules (PR)).

Table 3: Generated datasets description.

	Small-world Dataset			**Scale-free Dataset**		
Compo size	**PP**	**PR**	**Total**	**PP**	**PR**	**Total**
3	48	36	**84**	24	24	**48**
4	72	48	**120**	48	36	**84**
5	48	41	**89**	72	36	**108**
6	72	49	**121**	120	48	**168**
7	96	64	**160**	144	72	**216**
8	72	64	**136**	120	100	**220**
9	144	107	**251**	117	93	**210**
10	144	98	**242**	144	91	**235**

All experiments are performed on Intel Core i7-4700HQ 2.40GHz CPU, in a machine with 6 Gb memory, running a 64-bit Ubuntu distribution. The constrained best first search algorithm (CBFS) is implemented in Java 8 under NetBeans environment. We used the QMaxSAT solver [46] to implement the Max-SAT approach, and the Clingo [47] solver to implement the ASP approach. Note that for the Max-SAT approach, we tried a version of MaxSatz [48] called Maxsatz2013f and CCLS_to_Akmaxsat [2] [49], the results were not so good as the ones for QmaxSAT. All used datasets and algorithms implementation can be found at the following link [3].

The first experiment evaluates the influence of the number of privacy rules(PP-PR) on the efficiency of the proposed algorithms. For this, we fixed the composition size to 2 and the

[1]The available benchmarks like `http://www.uoguelph.ca/~qmahmoud/qws/`, are only adapted for the QoS composition problem

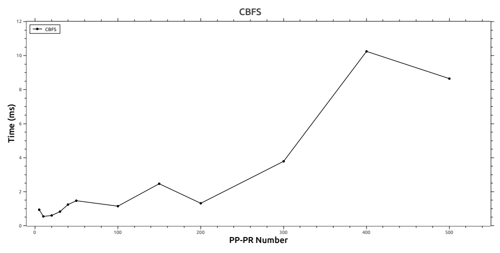
[2]`http://maxsat.ia.udl.cat/solvers/5/CCLS_to_akmaxsat_binaries.zip-201604080802`

[3]`https://www.dropbox.com/s/ag8w12tvonulib2/PrivacyDatasets.tar.gz?dl=0`.

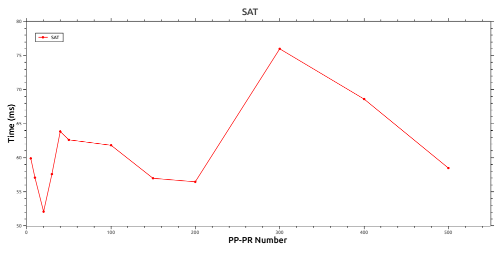
number of services in each class to 50, while we varied the number of privacy rules of each service from 5 to 500. The results of this experiment are shown in Table 4 and figure 8.

Table 4: Influence of the number of privacy rules(PP-PR) on the efficiency of the proposed algorithms

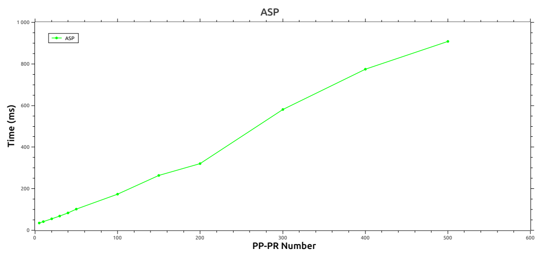
	Algorithms (ms)		
PP-PR number	**CBFS**	**SAT**	**ASP**
5	0.94	59.9	34.72
10	0.55	57.08	41.64
20	0.6	52.08	54.98
30	0.82	57.6	67.7
40	1.25	63.86	82.72
50	1.47	62.62	101.56
100	1.14	61.8	174.26
150	2.46	56.98	264.58
200	1.32	56.44	320.26
300	3.78	75.98	580.34
400	10.25	68.6	775.08
500	8.64	58.48	908.48



(a)



(b)



(c)

Figure 8: Influence of the number of privacy rules(PP-PR) on the efficiency of the proposed algorithms: (a) CBFS ,(b) SAT, (c) ASP.

The results of this experiment show that for the CBFS and the ASP algorithms, the complexity of the interactions between services (expressed by the number of privacy rules) has an apparent influence on the execution time. Mostly, the execution time increases with the increase of the number of privacy rules. Some exceptions exist for the CBFS algorithm (e.g. 150 with 200 and 400 with 500), where the execution time is smaller despite a larger number of rules. This can be explained by the fact that the execution time in the CBFS algorithm is determined by two factors (see Algorithm1): the time required to check services' compliance (which is strongly influenced by the number of privacy rules) and the time required for the manipulation of the priority queue (insertion, deletion and sorting). The execution time of the second factor is influenced by the size of the priority queue, which can increase or decrease according to the users' request independently of the number of privacy rules. In some situations there may be a larger queue with a smaller number of privacy rules, resulting in the handling time of larger queues greatly increasing the execution time.

The results are quite different for the SAT approach; in which we can't see a real dependency between the execution time and the number of privacy rules. This is principally due to our SAT encoding which is done at the services level. In this encoding, the only incompatible services are encoded as constraints (see equation 16), and not the interaction details. This makes our encoding independent of the complexity of interaction, thus, the execution time is independent too.

If we compare the effectiveness of each algorithm for this experiment (see Figure 9), we can clearly see that the CBFS and SAT approaches provide much better results than the ASP approach, with a small advantage for the CBFS algorithm over the SAT approach.

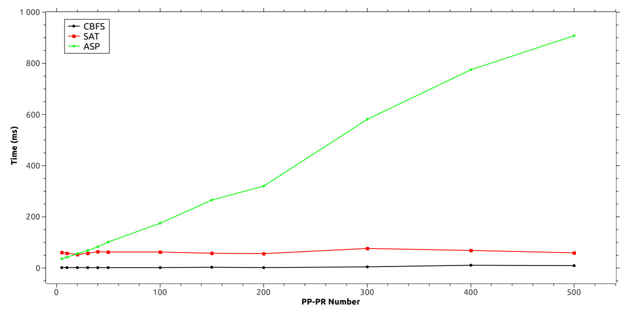
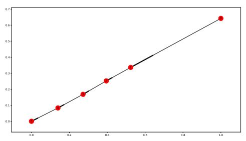


Figure 9: Comparison of algorithms effectiveness for the influence of the number of privacy rules(PP-PR) on the computation time.

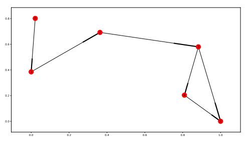
The aim of our next experiment is to investigate how the complexity of dependencies between abstract services affects the performance of the proposed algorithms. For this purpose, we started with a sequential abstract composition of five services after which we added one edge between services at a time. Consequently, we obtained 7 privacy graphs with a gradually increasing number of interactions from 5 to 11 (see Figure [4] 10). The use of a composition having only five services is based on the work of [39] who studied a collection of 6092 compositions and showed that 95.45% of them contained no more than 5 services. Note that in this experiment, an edge of a privacy graph represents one private data attribute and each abstract service has 50 candidate services.

The results of this experiment are shown in Table 5 and figure 11. The results didn't show any clear patterns between the complexity of interactions and the computation time. We can see for all algorithms that there exist for some graphs, a smaller computation time for bigger interactions and vis versa.

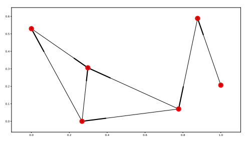
[4]Node 0 in all graphs represents a user request



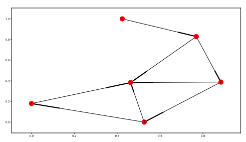
(a)



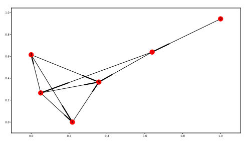
(b)



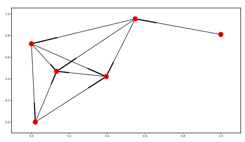
(c)



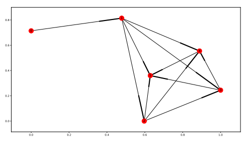
(d)



(e)

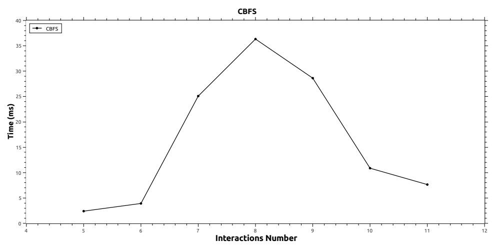


(f)

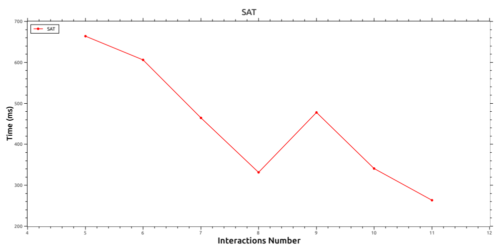


(g)

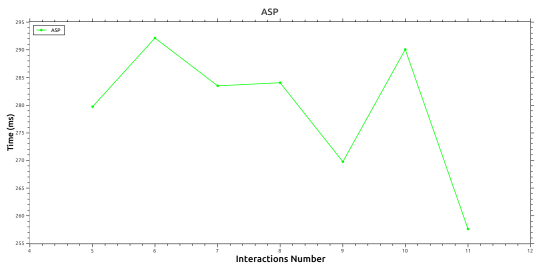
Figure 10: Privacy graphs used to test the influence of the complexity of services interaction on the efficiency of the proposed approaches



(a)



(b)



(c)

Figure 11: The influence of the complexity of service interactions on the efficiency of the proposed algorithms: (a) CBFS ,(b) SAT, (c) ASP.

Table 5: The influence of the complexity of service interactions on the efficiency of the proposed algorithms

	Algorithms		
Number of interactions	**CBFS (ms)**	**SAT (ms)**	**ASP (ms)**
5	2.41	663.89	279.71
6	3.93	605.82	292.12
7	25.08	464.57	283.47
8	36.34	331.8	284.07
9	28.65	478.01	269.77
10	10.87	340.59	290.07
11	7.67	263.14	257.63

If we compare the effectiveness of each algorithm for this experiment (Figure 12), we see clearly that the CBFS algorithm provides much better performance, while the SAT approach shows the worst one.

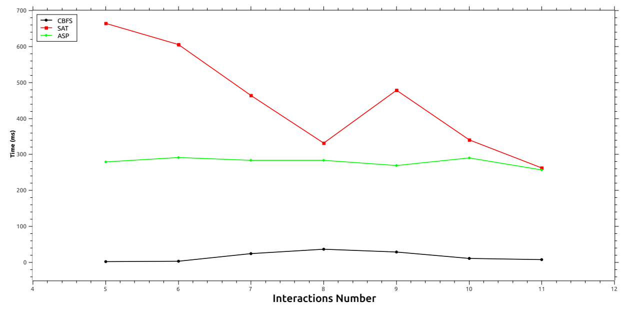
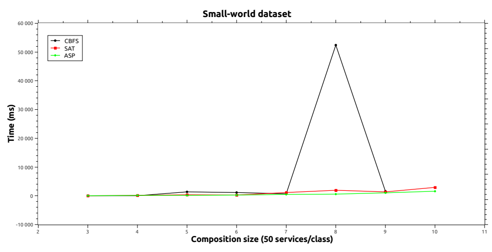


Figure 12: Comparison of algorithms effectiveness for the influence of the complexity of service interactions on computation time.

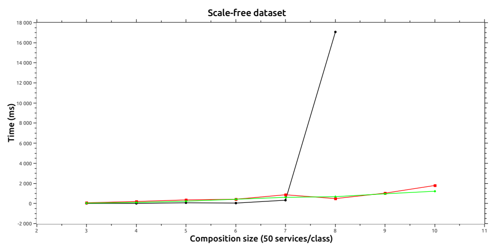
The next experiment evaluates the influence of the composition size on the efficiency of the proposed approaches. For this aim, we varied the composition size from 3 to 10, while we fixed the number of candidate services of each class to 50. The results of this experiment are shown in Table 6 and figure 13.

Table 6: The influence of the composition size on the efficiency of the proposed algorithms.

	Small-world Dataset			Scale-free Dataset		
Compo size	**CBFS**	**SAT**	**ASP**	**CBFS**	**SAT**	**ASP**
3	5.58	42,46	63,38	5,99	42,46	63,38
4	8.83	114,87	124,42	23,36	114,87	124,42
5	1 400.84	367,11	190,21	77,37	367,11	190,21
6	1 120.19	287,68	291,01	32,55	287,68	291,01
7	685.97	1161,54	498,01	334,5	1161,54	498,01
8	52 404	1948,77	642,58	170885	1948,77	642,58
9	1 571.62	1315,63	1013,18	timeout	1315,63	1013,18
10	timeout	2980,28	1560,82	timeout	2980,28	1560,82



(a)



(b)

Figure 13: The influence of the composition size on the efficiency of the proposed algorithms: (a) Small-world dataset, (b) Scale-free dataset.

The first observation drawn from these results, is that the computation time increases with the composition size, and is valid for almost all datasets and algorithms with some exceptions for the CBFS algorithm. The increase is more significant for the CBFS algorithm, so that we got a timeout response for the composition 10 in the small-world data set and the compositions 9 and 10 in the scale-free dataset (see table 6). For the CBFS algorithm, we can justify this increase by the fact that an increase in the composition size involves an increase in the depth of the final state (the sink node in our problem representation (see Figure 7)) of the search tree. Therefore, this induces a greater computation time. However, as regards the declarative approaches (MAx-SAT,ASP), an increase in the composition size, implies an increase in the number of variables and clauses of the generated Boolean formulas, thus, more computation time.

Abnormally high peak is observed for the CBFS algorithm at the compositions size 8 of the small-world dataset. That makes the computation time of the composition 9 much smaller of that of the composition 8. This leads us to state that the fact in which an increase in the composition size implies an increase in the computation time cannot be taken as a general rule. In fact, the used heuristic and the specificity of each problem instance can have a dramatic impact on the computation time. In our results, to find a solution, the CBFS algorithm handles a priority queue of an average size of 400 in the composition 9, while the average size of the priority queue in the composition 8 is about 4360 (10 times bigger), which may explain the observed execution time peak.

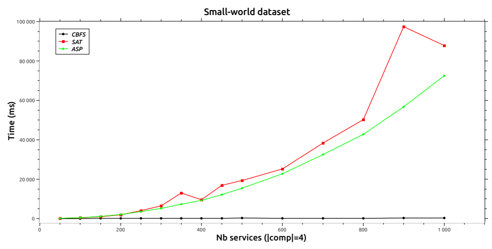
We also notice that the effectiveness of the CBFS algorithm is better for small compositions (compositions 3 and 4 in the small-world dataset, and compositions from 3 to 7 in the scale-free dataset). However, as the size becomes larger, declarative approaches become better and the difference becomes very important.

Our last experiment evaluates the influence of the number of candidate services per class on the computation time of the proposed approaches. To this end, we fixed the composition size to 4, and varied the number of services per class from 50 to 1000. The results of this experiment are shown in table 7 and figure 14.

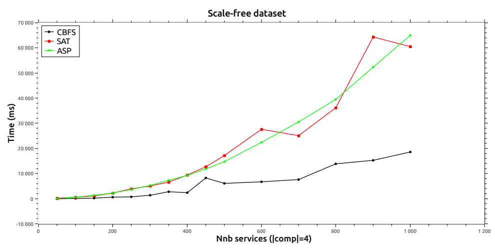
The results of this experiment show clearly that the computation time increases as the number of services per class increases. Some exceptions are made by the SAT approach whereby we find a smaller computation time with bigger candidate services. This behavior is explained by the fact that SAT approaches take advantage of many heuristics to speed up the solution search. Thus, the size increase of generated Boolean formulas induced by the increase in the number of candidate services is not the only factor that influences the computation time. The effectiveness of the applied heuristics on a particular problem in-

Table 7: The influence of the number of candidate services per class on the computation time of the proposed algorithms.

	Small-world Dataset			Scale-free Dataset		
Compo size	**CBFS**	**SAT**	**ASP**	**CBFS**	**SAT**	**ASP**
50	5,58	114,87	124,42	5,99	196,5	109,17
100	20,9	445,99	480,35	86,58	560,61	547,02
150	40,42	995,9	1100,95	243,33	1067,9	1297,69
200	52,79	1934,32	2064,22	542,03	2286,13	2257,48
250	134,05	4020,09	3486,84	757,59	3946,97	3590,85
300	190,07	6388,18	5091,78	1308,87	5067,82	5261,65
350	109,29	12889,07	7346,31	2809,9	6556,83	7293,55
400	143,5	9494,64	9187,11	2420,81	9412,54	9272,53
450	169,88	16936,5	12216,89	8242,6	12762,48	11840,33
500	247,34	19259,89	15346,48	6042,98	17155,77	14772,47
600	123,48	25139,14	22656,14	6669,8	27619,09	22382,34
700	182,4	38368,14	32421,25	7654,56	25054,57	30514,04
800	148,51	50235,33	42687,79	13854,46	36131,42	39473,17
900	318,75	97438,17	56722,6	15212,05	64424,25	52344,86
1000	298,75	87783,4	72578,13	18637,97	60564,86	65067,82



(a)



(b)

Figure 14: The influence of the number of candidate services per class on the computation time of the proposed algorithms: (a) Small-world dataset, (b) Scale-free dataset.

stance can have an apparent impact on the computation time. This experiment confirms our previous conclusion that the CBFS algorithm gives better performance on small compositions. This result is more apparent in the small-world dataset and is justified by the fact that in the small composition, the depth of the final state in the multi level graph also become small, and therefore quickly reachable. It is also important to notice that the effectiveness of the declarative approaches are very close for both datasets.

In summary, and according to the previous experiments, it is difficult to adopt a particular approach to be used in our selection framework. While the CBFS algorithm gives better performance in smaller compositions, it is less scalable in larger compositions. In this case, the declarative approaches show better performance. The results of the experiments denote that the effectivenesses of the declarative approaches are very close. This remains true despite the little advantage of ASP over the SAT approach. We also notice that the ASP solver has shown a very stable behavior which is a very interesting feature, especially if the selection system have to provide some response time guarantee. Finally, it is important to note that these results concern only QmaxSat and Clingo solver and can not be generalized to all Max-SAT and ASP solvers. Our objective was to demonstrate the adequacy of such approaches on the proposed privacy model as well as in the privacy aware service selection area. Accordingly, we can benefit from all advantages of declarative approaches like flexibility and extensibility. these properties allow us, for example, to easily add or remove constraints without remodeling or rewriting all selection algorithms. Furthermore, these approaches offer the possibility to benefit from the continuous progress of SAT and ASP solvers to even improve the performances of the selection framework.

7 Related Work

A number of approaches have been devoted to Web Service privacy. The vast majority of them are based on checking the compliance between privacy policies of service providers and privacy requirements of users. In this section, we will briefly mention the most relevant approaches to our work. A more complete state-of-the-art analysis is given in [7, 8].
In [9], the authors proposed an approach for enhancing DaaS Web Services (data as a service) privacy. In this context, they proposed a formal model that enables users and service providers to define a set of privacy rules to express their privacy policies and requirements. They propose an algorithm called PCM (Privacy Compatibility Matching) to check the privacy compatibility between the policies and the requirements in a DaaS composition. Unlike our work, this approach describes a negotiation mechanism that establishes a dynamic reconciliation between the services in case of incompatibility. However, the authors did not use any heuristics in their algorithms, which may pose problems in large dimensionality.
In [10], the authors proposed a goal based approach to protect users' and service providers' privacy in Web Service compositions. The principle is based on a model that permits privacy policies and preferences of users and service providers to be implemented across multiple dimensions. Unlike our work, the granularity is not used as a privacy metric, but rather, the authors have used the sensitivity of data attributes conjointly with the purpose, visibility and retention time. A service composition algorithm is proposed to check the conformity between users' privacy requirements and the privacy policies of the service providers, then, a privacy based selection is made to choose the most privacy preserving composition. The selection is based on a ranking function which aggregates the used privacy dimensions. The proposed algorithm has the advantage of handling both functional and non-functional(privacy) aspects in the same time, while most approaches process them in separate steps. Nevertheless, the authors in this work did not provide a real implementation to prove their privacy model.
In [11] the authors proposed a privacy-aware replaceability protocol for Web Services. In order to define this protocol, the authors proposed a rule based model which is an extension of the P3P standard. The model is used to express the privacy policies and preferences of users and service providers. The integration of this model in the business protocols has permitted a replaceability analysis in both functional and non-functional (privacy) aspects. Contrary to our work, the proposed model does not support the privacy protection in service compositions context.
In [12] the authors proposed a framework that provides a privacy-conscious composite service to consumers. In this framework, the service provider, interacts with the users and provides the principles (model) that guide the private data utilization. In the case of conflicts between the consumer privacy preferences and the received model, the consumer may relax his/her privacy policies so that the service can be used or rather, generates obligations that represent the privacy violation and forwards them to the composite service to be enforced. The main difference between this approach and our work is that this proposal

focuses only on users' privacy, and does not deal with the service providers privacy like we do in our work. Another difference is that the proposed model uses only the sensitivity of data attribute as a privacy predicate. However, this work takes into account the level of the trust of service providers, which is not done by our work.
In [13] the authors presented a negotiation approach for finding a compromise between users' privacy preferences and service providers requirements. The approach is based on a Galois lattice structure. The lattice is generated using an evaluation matrix that links the items (name, address, e-mail, etc.) and the different types of sub-services (log-in, registration, payment, etc.). The generated lattices are subsequently used to develop a set of rules that define the preferences and the requirements of users and service providers. The negotiation process is made by a variation of a parameter "θ" over a range of 1 to 5 which has the effect of changing the corresponding lattice and therefore the privacy rules. The major drawback of this approach lies in the definition of the evaluation matrix which remains a difficult task for a simple user. In addition, this approach does not support privacy protection of service compositions.
In [14], the authors deal with privacy concerns in Digital government Web Services. Here, the authors proposed three privacy protection levels: user privacy, service privacy and data privacy. More specifically, the entities that request access to the data or to the operations of other entities, must provide credentials for performing this function. If the requesting access is allowed, then, a data filter is used to control the access, after which, a mobile privacy preserving agent is used to deliver the requested data, ensuring that the requester does not violate the local entity's privacy requirements. Note that, in this work, the authors focused only on privacy of simple Web Services and did not consider service compositions privacy.
In [15] the authors proposed a Web Services selection framework whose objective is to protects users' and service provider's' privacy needs. This work addresses privacy concerns associated with the disclosure of user's personal data during Web Services location at the phase of the providers provisioning rules verification. The privacy of provisioning rules of service providers is also covered. The main component of the proposed framework is called the Private Negotiator. The latter is based on a Matching Protocol that allows two parties to jointly calculate the intersection of their inputs, without leaking any additional information. This work differs from our work in that the used privacy protection mechanism is based on a cryptographic techniques and not on privacy rules matching.
In [16], the authors proposed a privacy-preserving approach for Web Service compositions. In this approach the authors proposed a component called ElitePicker application (EPApp). This component implements a security protocol based on encryption techniques. The encryption mechanism allows the verification of the users'/ servicer's' requirements in a private way. Contrary to our work that deals with the privacy of Web Service orchestration model, this work deals with the privacy of service choreography model.

8 Conclusion

In this work we have proposed a privacy model for preserving privacy in Web Services. The model allows a privacy preservation of both users and service providers. The proposed model is implemented in a Web Service selection problem whose objective is to find a composition that best fulfills all the privacy requirements of both users and service providers. Several selection algorithms are designed with respect to the proposed model. In addition, all proposed algorithms return compositions that satisfy all privacy constraints and minimize the risk of a privacy threat. If any service violates one of the privacy constraints, that service will be discarded from the solution set. As future work, we plan to explore other generic problem solving approaches like the Satisfiability modulo theories (SMT). Another promising direction for future work, consists of handling the cases where the set of solutions is empty (ie. the privacy constraints are violated). One solution to deal with this situation is by modifying the way in which our algorithms search for solutions. For instance, instead of searching a valid composition (satisfies all privacy constraints) that minimizes the privacy risk, we can opt for a multi-objective optimization approach. Thus, we minimize at the same time the privacy risk driven by the compositions as well as the number of violated constraints. Note that this modification can be done without any adaptation of the proposed privacy model. Another solution consists of extending the privacy model to support a negotiation process. The negotiation can be initiated by the selection framework if the solutions set is empty. In such a case, the selection framework interacts with the providers of services in order to relax their privacy constraints. After that, the search is restarted by giving the priority to the services that accepted the constraints relaxation. Obviously, the negotiation process provides more efficient privacy protection, but at the same time introduces more time and resource consumption.

References

[1] Mohammad Alrifai, Thomas Risse, and Wolfgang Nejdl. A hybrid approach for efficient web service composition with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 6(2):7:1–7:31, 2012.

[2] Quanwang Wu, Qingsheng Zhu, and Peng Li. A caching mechanism for qos-aware service composition. *Journal of Web Engineering*, 11(2):119–130, 2012.

[3] Angus FM Huang, Ci-Wei Lan, and Stephen JH Yang. An optimal qos-based web service selection scheme. *Information Sciences*, 179(19):3309–3322, 2009.

[4] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6, 2007.

[5] YC Liu and YB Liu. A sort of web service selection strategy based on the fusion of qos and service reliability. *International Journal of Computer Science Issues*, 10(1):414–420, 2013.

[6] Amal Halfaoui, Hadjila Fethallah, and Fedoua Didi. Qos-aware web services selection based on fuzzy dominance. In *Computer Science and Its Applications - 5th IFIP TC 5 International Conference, CIIA 2015, Saida, Algeria, May 20-21, 2015, Proceedings*, pages 291–300, 2015.

[7] Changbo Ke, Zhiqiu Huang, and Mei Tang. Supporting negotiation mechanism privacy authority method in cloud computing. *Knowledge-Based Systems*, 51:48–59, 2013.

[8] Changbo Ke, Ruchuan Wang, Fu Xiao, and Zhiqiu Huang. Requirement-oriented privacy protection analysis architecture in cloud computing. *Journal of Communications*, 10(1):55–63, 2015.

[9] Salah-Eddine Tbahriti, Chirine Ghedira, Brahim Medjahed, and Michael Mrissa. Privacy-enhanced web service composition. *IEEE Transactions on Services Computing*, 7(2):210–222, 2014.

[10] Elisa Costante, Federica Paci, and Nicola Zannone. Privacy-aware web service composition and ranking. In *20th IEEE International Conference on Web Services (ICWS)*, pages 131–138. IEEE, 2013.

[11] Nawal Guermouche, Salima Benbernou, Emmanuel Coquery, and Mohand-Said Hacid. Privacy-aware web service protocol replaceability. In *IEEE International Conference on Web Services, ICWS.*, pages 1048–1055. IEEE, 2007.

[12] Wei Xu, VN Venkatakrishnan, R Sekar, and IV Ramakrishnan. A framework for building privacy-conscious composite web services. In *International Conference on Web Services, ICWS'06.*, pages 655–662. IEEE, 2006.

[13] Ohbyung Kwon, Yonnim Lee, and Debashis Sarangib. A galois lattice approach to a context-aware privacy negotiation service. *Expert Systems with Applications*, 38(10):12619–12629, 2011.

[14] Abdelmounaam Rezgui, Mourad Ouzzani, Athman Bouguettaya, and Brahim Medjahed. Preserving privacy in web services. In *Proceedings of the 4th international workshop on Web information and data management*, pages 56–62. ACM, 2002.

[15] Anna Cinzia Squicciarini, Barbara Carminati, and Sushama Karumanchi. Privacy aware service selection of composite web services invited paper. In *9th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom)*, pages 260–268. IEEE, 2013.

[16] Barbara Carminati, Elena Ferrari, and Ngoc Hong Tran. A privacy-preserving framework for constrained choreographed service composition. In *IEEE International Conference on Web Services (ICWS)*, pages 297–304. IEEE, 2015.

[17] Ken Barker, Mina Askari, Mishtu Banerjee, Kambiz Ghazinour, Brenan Mackas, Maryam Majedi, Sampson Pun, and Adepele Williams. A data privacy taxonomy. In *Dataspace: The Final Frontier*, pages 42–54. Springer, 2009.

[18] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The platform for privacy preferences 1.0 (p3p1. 0) specification. *W3C recommendation*, 16, 2002.

[19] Kambiz Ghazinour and Ken Barker. Capturing p3p semantics using an enforceable lattice-based structure. In *Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society*, pages 4:1–4:6. ACM, 2011.

[20] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the web services web: an introduction to soap, wsdl, and uddi. *IEEE Internet computing*, 6(2):86–93, 2002.

[21] Quan Z Sheng, Xiaoqiang Qiao, Athanasios V Vasilakos, Claudia Szabo, Scott Bourne, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218–238, 2014.

[22] Diane Jordan, John Evdemon, Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, et al. Web services business process execution language version 2.0. *OASIS standard*, 11(120):5, 2007.

[23] Chu Min Li and Felip Manyà. Maxsat. In *Handbook of satisfiability*, chapter 19, pages 613–631. ios press, 2009.

[24] Tomi Janhunen and Ilkka Nimelä. The answer set programming paradigm. *AI Magazine*, 37(3):13–24, 2016.

[25] Benjamin Kaufmann, Nicola Leone, Simona Perri, and Torsten Schaub. Grounding and solving in answer set programming. *AI Magazine*, 37(3):25–32, 2016.

[26] Vladimir Lifschitz. Answer sets and the language of answer set programming. *AI Magazine*, 37(3):7–12, 2016.

[27] Kambiz Ghazinour, Amir H Razavi, and Ken Barker. A model for privacy compromisation value. *Procedia Computer Science*, 37:143–152, 2014.

[28] David Pisinger. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research*, 83(2):394–410, 1995.

[29] Michel Grabisch. The application of fuzzy integrals in multicriteria decision making. *European journal of operational research*, 89(3):445–456, 1996.

[30] Jean-Luc Marichal. An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria. *IEEE Transactions on Fuzzy Systems*, 8(6):800–807, 2000.

[31] Michel Grabisch, Ivan Kojadinovic, and Patrick Meyer. A review of methods for capacity identification in choquet integral based multi-attribute utility theory: Applications of the kappalab

r package. *European journal of operational research*, 186(2):766–785, 2008.

[32] Pablo Cingolani and Jesus Alcala-Fdez. jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In *FUZZ-IEEE*, pages 1–8. Citeseer, 2012.

[33] R Core Team. R: A language and environment for statistical computing. r foundation for statistical computing, vienna, austria, 2012, 2014.

[34] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[35] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan Sturtevant, Robert C Holte, and Jonathan Schaeffer. Enhanced partial expansion a*. *Journal of Artificial Intelligence Research*, 50(1):141–187, 2014.

[36] Will Klieber and Gihwon Kwon. Efficient cnf encoding for selecting 1 from n objects. In *Proc. International Workshop on Constraints in Formal Verification*, 2007.

[37] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138(1):39–54, 2002.

[38] Zhiyong Feng, Bo Lan, Zhen Zhang, and Shizhan Chen. A study of semantic web services network. *The Computer Journal*, 58(6):1293–1305, 2015.

[39] Keman Huang, Yushun Fan, and Wei Tan. An empirical study of programmable web: A network analysis on a service-mashup system. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 552–559. IEEE, 2012.

[40] Chantal Cherifi, Vincent Labatut, and Jean-François Santucci. Benefits of semantics on web service composition from a complex network perspective. In *International Conference on Networked Digital Technologies*, pages 80–90. Springer, 2010.

[41] Hyunyoung Kil, Seog-Chan Oh, Ergin Elmacioglu, Wonhong Nam, and Dongwon Lee. Graph theoretic topological analysis of web service networks. *World Wide Web*, 12(3):321–343, 2009.

[42] Seog-Chan Oh, Dongwon Lee, and Soundar RT Kumara. Effective web service composition in diverse and large-scale service networks. *IEEE Transactions on Services Computing*, 1(1):15–32, 2008.

[43] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[44] Béla Bollobás, Christian Borgs, Jennifer Chayes, and Oliver Riordan. Directed scale-free graphs. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 132–139. Society for Industrial and Applied Mathematics, 2003.

[45] Aric Hagberg, Dan Schult, Pieter Swart, D Conway, L Séguin-Charbonneau, C Ellison, B Edwards, and J Torrents. Networkx. high productivity software for complex networks. *Webová strá nka https://networkx. lanl. gov/wiki*, 2013.

[46] Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. Qmaxsat: A partial max-sat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 8:95–100, 2012.

[47] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo= asp+ control: Preliminary report. *arXiv preprint arXiv:1405.3694*, 2014.

[48] Chu Min Li, Felip Manyà, Nouredine Mohamedou, and Jordi Planes. Exploiting cycle structures in max-sat. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 467–480. Springer, 2009.

[49] Chuan Luo, Shaowei Cai, Wei Wu, Zhong Jie, and Kaile Su. Ccls: an efficient local search algorithm for weighted maximum satisfiability. *IEEE Transactions on Computers*, 64(7):1830–1843, 2015.