Balancing Data Quality and Protection in Distributed Big Data Pipelines

Antongiacomo Polimeno[1], Chiara Braghin[1], Marco Anisetti[1], Claudio A. Ardagna[1*]

[1]Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy.

*Corresponding author(s). E-mail(s): claudio.ardagna@unimi.it;
Contributing authors: antongiacomo.polimeno@unimi.it;
chiara.braghin@unimi.it; marco.anisetti@unimi.it;

Keywords: Access Control, Big Data, Data Transformation, Data Ingestion

Today, the increasing ability of collecting and managing huge volume of data, coupled with a paradigm shift in service delivery models, has significantly enhanced scalability and efficiency in data analytics, particularly in multi-tenant environments. Data are today treated as digital products, which are managed and analyzed by multiple services orchestrated in data pipelines. This scenario calls for innovative solutions to data pipeline management that primarily seek to balance data quality and data protection. Departing from the state of the art that traditionally optimizes data protection and data quality as independent factors, we propose a framework that enhances service selection and composition in distributed data pipelines to the aim of maximizing data quality, while providing a minimum level of data protection. Our approach first

retrieves a set of candidate services compatible with data protection requirements in the form of access control policies; it then selects the subset of compatible services, to be integrated within the data pipeline, which maximizes the overall data quality. Being our approach NP-hard, a sliding-window heuristic is defined and experimentally evaluated in terms of performance and quality with respect to the exhaustive approach. Our results demonstrate a significant reduction in computational overhead, while maintaining high data quality.

1 Introduction

The wide success and adoption of cloud infrastructures and their intrinsic multitenancy represent a paradigm shift in the big data scenario, redefining scalability and efficiency in data analytics. Multitenancy enables multiple users to share resources, such as computing power and storage, optimizing their utilization and reducing operational costs. Leveraging cloud infrastructure further enhances flexibility and scalability. The flip side of multitenancy is the increased complexity in data governance: the shared model introduces unique security challenges, as tenants may have different security requirements, access levels, and data sensitivity. Adequate measures such as encryption, access control mechanisms, and data anonymization techniques must be implemented to safeguard data against unauthorized access and ensure compliance with regulatory requirements such as GDPR or HIPAA. As a consequence, achieving a balance between data protection and data quality is crucial, as the removal or alteration of personally identifiable information from datasets to safeguard individuals' privacy can compromise the accuracy of analytics results.

So far, all research endeavors have been mainly concentrated on exploring these two issues separately: on one hand, *data quality*, encompassing accuracy, reliability, and suitability, has been investigated to understand the implications in analytical contexts. Although extensively studied, these investigations often prioritize enhancing

the quality of source data rather than ensuring data quality throughout the entire processing pipeline, or the integrity of outcomes derived from data. On the other hand, *data security and privacy* focused on the protection of confidential information and adherence to rigorous privacy regulations.

A valid solution however requires a holistic approach that integrates technological solutions, organizational policies, and ongoing monitoring and adaptation to emerging threats and regulatory changes. The implementation of robust access control mechanisms, ensuring that only authorized users can access specific datasets or analytical tools is just a mandatory but initial step. Additional requirements are emerging. First, data protection requirements should be identified at each stage of the data lifecycle, potentially integrating techniques like data masking and anonymization to safeguard sensitive information, thereby preserving data privacy while enabling data sharing and analysis. Second, data lineage should be prioritized, fostering a comprehensive understanding and optimization of data flows and transformations within complex analytical ecosystems.

When evaluating a solution meeting these criteria, the following questions naturally arise:

1. How does a robust data protection policy affect analytics?
2. When considering a (big data) pipeline, should data protection be implemented at each pipeline step rather than filtering all data at the outset?
3. In a scenario where a user has the option to choose among various candidate services, how might these choices affect the analytics?

Based on the aforementioned considerations, we propose a data governance framework for modern data-driven pipelines, designed to mitigate privacy and security risks. The primary objective of this framework is to support the selection and assembly of data processing services within the pipeline, with a central focus on the selection of those services that optimize data quality, while upholding privacy and security

requirements. To this aim, each element of the pipeline is *annotated* with *i)* data protection requirements expressing transformation on data and *ii)* functional specifications on services expressing data manipulations carried out during each service execution. Though applicable to a generic scenario, our data governance approach starts from the assumption that maintaining a larger volume of data leads to higher data quality; as a consequence, its service selection algorithm focuses on maximizing data quality by retaining the maximum amount of information when applying data protection transformations.

The primary contributions of the paper can be summarized as follows: 1. Defining a data governance framework supporting selection and assembly of data processing services enriched with metadata that describe both data protection and functional requirements; 2. Proposing a parametric heuristic tailored to address the computational complexity of the NP-hard service selection problem; 3. Evaluating the performance and quality of the algorithm through experiments conducted using the dataset from the running example.

The remainder of the paper is structured as follows: Section 2, presents our system model, illustrating a reference scenario where data is owned by multiple organizations. Section 3 introduces the pipeline template and describe data protection and functional annotations. Section 4 describes the process of building a pipeline instance from a pipeline template according to service selection. Section 5 introduces the quality metrics used in service selection and the heuristic solving the service selection problem. Section 6 presents our experimental results. Section 7 discusses the state of the art and Section 8 draws our concluding remarks.

2 System Model and Reference Scenario

We present our system model (Section 2.1) and our reference scenario (Section 2.2).

2.1 System Model

We consider a service-based environment where a service pipeline is designed to analyze data. Our system model is derived by a generic big-data framework and enriched with metadata specifying data protection requirements and functional specifications. It is composed of the following parties:

- *Service*, a software distributed by a service provider that performs a specific task;
- *Service Pipeline*, a sequence of connected services that collect, prepare, process, and analyze data in a structured and automated manner;
- *Data Governance Policy*, a structured set of privacy guidelines, rules, and procedures regulating data access, sharing, and protection;
- *User*, executing an analytics pipeline on the data. We assume the user is authorized to perform this operation, either as the data owner or as a data processor with the owner's consent.
- *Dataset*, the data target of the analytics pipeline. We assume all data are ready for analysis, that is, they underwent a preparatory phase addressing issues such as missing values, outliers, and formatting discrepancies.

A service pipeline is a graph formally defined as follows.

Definition 1 (Service Pipeline)**.** A Pipeline is as a direct acyclic graph G(V,E), where V is a set of vertices and E is a set of edges connecting two vertices v_i,$v_k \in V$. The graph has a root ($\bullet$) vertex $v_r \in V$, a vertex $v_i \in V_S$ for each service s_i, an additional vertex $v_f \in V$ for each parallel ($\oplus$) structure modeling the contemporary execution (*fork*) of services.

We note that $V = \{v_r, v_f\} \cup V_S$, with vertices v_f modeling branching for parallel structures, and root v_r possibly representing the orchestrator. In addition, for simplicity but no lack of generality, alternative structures modeling the alternative execution

of services are specified as alternative service pipelines, that is, there is no alternative structure in a single service pipeline.

We refer to the service pipeline annotated with both functional and non-functional requirements, as the **pipeline template**. It acts as a skeleton, specifying both the structure of the pipeline, that is, the chosen sequence of desired services, and the functional and non-functional requirements for each component service. We note that, in our multi-tenant cloud-based ecosystem, each element within the pipeline may have a catalog of candidate services. A pipeline template is then instantiated in a **pipeline instance** by selecting the most suitable candidates from the pipeline template.

This process involves retrieving a set of compatible services for each vertex in the template, ensuring that each service meets the functional requirements and aligns with the policies specified in the template. Since we also consider security policies that may necessitate security and privacy-aware data transformations, compatible services are ranked based on their capacity to fulfill the policy while preserving the maximum amount of information (*data quality* in this paper). Indeed, our data governance approach, though applicable in a generic scenario, operates under the assumption that *preserving a larger quantity of data correlates with enhanced data quality*, a principle that represents many real-world scenarios. However, we acknowledge that this assumption may not universally apply and remain open to exploring alternative solutions in future endeavors. The best service is then selected to instantiate the corresponding component service in the template. Upon selecting the most suitable service for each component service in the pipeline template, the pipeline instance is completed and ready for execution.

Table 1: Dataset sample

DOWNLOAD DATE	ID	FNAME	LNAME	LAD	RACE	GENDER	AGE	BOND	OFFENSE	...
05/15/2020	ZZHCZBZZ	ROBERT	PIERCE	08/16/2018	BLACK	M	27	150000	CRIMINAL POSS ...	...
05/15/2020	ZZHZZRLR	KYLE	LESTER	03/28/2019	HISPANIC	M	41	30100	VIOLATION OF P...	...
05/15/2020	ZZSRJBEE	JASON	HAMMOND	04/03/2020	HISPANIC	M	21	150000	CRIMINAL ATTEM...	...
05/15/2020	ZZHBJLRZ	ERIC	TOWNSEND	01/15/2020	WHITE	M	36	50500	CRIM VIOL OF P...	...
05/15/2020	ZZSRRCHH	MICHAEL	WHITE	12/26/2018	HISPANIC	M	29	100000	CRIMINAL ATTEM...	...
05/15/2020	ZZEJCZWW	JOHN	HARPER	01/03/2020	WHITE	M	54	100000	CRIM VIOL OF P...	...
05/15/2020	ZZHJBJBR	KENNETH	JUAREZ	03/19/2020	HISPANIC	M	35	100000	CRIM VIOL ST C...	...
05/15/2020	ZZESESZW	MICHAEL	SANTOS	12/03/2018	WHITE	M	55	50000	ASSAULT 2ND, V...	...
05/15/2020	ZZRCSHCZ	CHRISTOPHER	JONES	05/13/2020	BLACK	M	43	10000	INTERFERING WIT...	...

2.2 Reference Scenario

Our reference scenario considers a service pipeline analyzing a dataset of individuals detained in the Department of Correction facilities in the state of Connecticut while awaiting trial[1].

Table 1 presents a sample of the adopted dataset. Each row represents an inmate; each column includes the following attributes: date of download, a unique identifier, last entry date, race, gender, age of the individual, the bound value, offense, entry facility, and detainer. To serve the objectives of our study, we extended this dataset by introducing randomly generated first and last names.

In this context, the user, a member of the Connecticut Department of Correction (DOC), is interested to compare the admission trends in Connecticut prisons with the ones in New York and New Hampshire. We assume that the three DOCs are partners and share data according to their privacy policies. Moreover, the policy specifies that the entire service execution must occur within the Connecticut Department of Correction. In case data transmission extends beyond Connecticut's borders, data protection measures must be implemented.

The user's objective aligns with the predefined service pipeline in Figure 1 that orchestrates the following sequence of operations: (i) *Data fetching*, including the download of the dataset from other states; (ii) *Data preparation*, including data merging, cleaning, and anonymization; (iii) *Data analysis*, including statistical measures

[1] https://data.ct.gov/Public-Safety/Accused-Pre-Trial-Inmates-in-Correctional-Faciliti/b674-jy6w

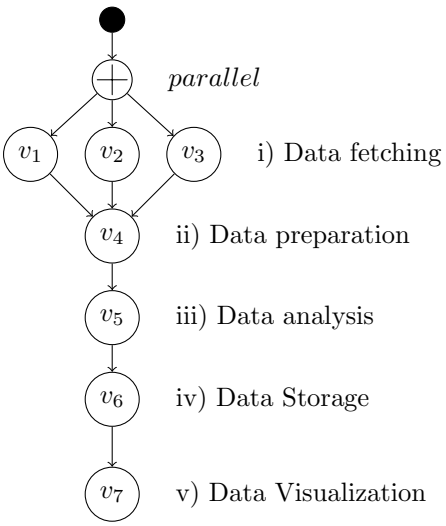


Fig. 1: Service pipeline in the reference scenario

like average, median, and clustering-based statistics; (iv) *Data storage*, including the storage of the results; (v) *Data visualization*, including the visualization of the results.

3 Pipeline Template

Our approach integrates data protection and data management into the service pipeline using annotations. To this aim, we extend the service pipeline in Definition 1 with: *i)* data protection annotations that express transformations on data, ensuring compliance with data protection requirements, *ii)* functional annotations that express data manipulations carried out during service execution. These annotations enable the implementation of an advanced data lineage, tracking the entire data lifecycle by monitoring changes that result from functional service execution and data protection requirements.

In the following, we first introduce the annotated service pipeline, called pipeline template (Section 3.1). We then present both functional annotations (Section 3.3) and data protection annotations (Section 3.2), providing an example of a pipeline template in the context of the reference scenario.

3.1 Pipeline Template Definition

Given the service pipeline in Definition 1, we use annotations to express data protection requirements to be enforced on data and functional requirements on the services to be integrated in the pipeline. Each service vertex in the service pipeline is labeled with two mapping functions forming a pipeline template: *i*) an annotation function $\lambda : V_S \rightarrow P$ that associates a set of data protection requirements, in the form of policies $p \in P$, with each vertex $v_i \in V_S$; *ii*) an annotation function $\gamma : V_S \rightarrow F$ that associates a functional service description $F_i \in F$ with each vertex $v_i \in V_S$.

The template is formally defined as follows.

Definition 2 (Pipeline Template)**.** Given a service pipeline G(V,E), a pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ is a direct acyclic graph extended with two annotation functions:

1. *Data Protection Annotation* λ that assigns a label $\lambda(v_i)$ to each vertex $v_i \in V_S$. Label $\lambda(v_i)$ corresponds to a set P_i of policies p_j to be satisfied by service s_i represented by v_i;
2. *Functional Annotation* γ that assigns a label $\gamma(v_i)$ to each vertex $v_i \in V_S$. Label $\gamma(v_i)$ corresponds to the functional description F_i of service s_i represented by v_i.

We note that, at this stage, the template is not yet linked to any service. We also note that policies $p_j \in P_i$ in $\lambda(v_i)$ are combined using logical OR, meaning that the access decision is positive if at least one policy p_j evaluates to *true*. The pipeline template of the service pipeline of Fig. 1 is depicted in Fig. 2.

3.2 Data Protection Annotation

Data Protection Annotation λ expresses data protection requirements in the form of access control policies. We consider an attribute-based access control model that offers flexible fine-grained authorization and adapts its standard key components to

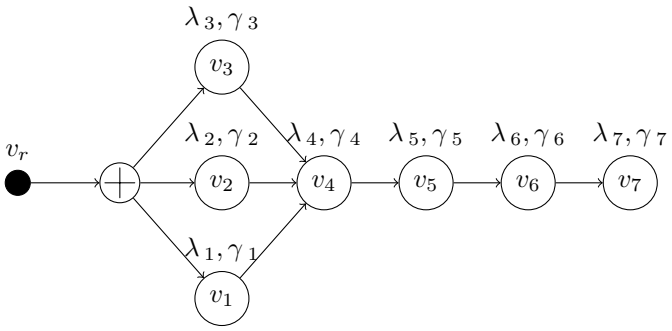


Fig. 2: Pipeline Template

address the unique characteristics of a big data environment. Access requirements are expressed in the form of policy conditions that are defined as follows.

Definition 3 (Policy Condition)**.** A *Policy Condition pc* is a Boolean expression of the form (*attr_name* op *attr_value*), with op∈{<,>,=,≠,≤,≥}, *attr_name* an attribute label, and *attr_value* the corresponding attribute value.

Built on policy conditions, an access control policy is then defined as follows.

Definition 4 (Policy)**.** A *policy* $p \in P$ is 5-uple $<subj, obj, act, env, T^P>$ that specifies who (*subject*) can access what (*object*) with action (*action*), in a specific context (*environment*) and under specific obligations (*data transformation*).

More in detail, *subject subj* specifies a service s_i issuing an access request to perform an action on an object. It is a set $\{pc_i\}$ of *Policy Conditions* as defined in Definition 3. For instance, (classifier="SVM") specifies a service providing a SVM classifier. We note that *subj* can also specify conditions on the service owner (*e.g.*, owner_location="EU") and the service user (*e.g.*, service_user_role="DOC Director").

Object obj defines the data governed by the access policy. In this case, it is a set $\{pc_i\}$ of *Policy Conditions* on the object's attributes. For instance, {(type="dataset"), (region="CT")} refers to an object of type dataset and whose region is Connecticut.

Action act specifies the operations that can be performed within a big data environment, from traditional atomic operations on databases (e.g., CRUD operations) to coarser operations, such as an Apache Spark Direct Acyclic Graph (DAG), Hadoop MapReduce, an analytics function call, and an analytics pipeline.

Environment env defines a set of conditions on contextual attributes, such as time of the day, location, IP address, risk level, weather condition, holiday/workday, and emergency. It is a set $\{pc_i\}$ of *Policy Conditions* as defined in Definition 3. For instance, (time="night") refers to a policy that is applicable only at night.

Data Transformation T^P defines a set of security and privacy-aware transformations on *obj* that must be enforced before any access to data is given. Transformations focus on data protection, as well as on compliance to regulations and standards, in addition to simple format conversions. For instance, let us define three transformations that can be applied to the dataset in Table 1, each performing different levels of anonymization: i) level *l0* (t_0^p): no anonymization; ii) level *l1* (t_1^p): partial anonymization with only first and last name being anonymized; iii) level *l2* (t_2^p): full anonymization with first name, last name, identifier and age being anonymized.

Access control policies $p_j \in P_i$ annotating a vertex v_i in a pipeline template $G^{\lambda,\gamma}$ specify the data protection requirements that a candidate service must fulfill to be selected in the pipeline instance. Section 4 describes the selection process and the pipeline instance generation.

3.3 Functional Annotations

A proper data management approach must track functional data manipulations across the entire pipeline execution, defining the functional requirements of each service operating on data. To this aim, each vertex $v_i \in V_S$ is annotated with a label $\gamma(v_i)$, corresponding to the functional description F_i of the service s_i represented by v_i. F_i describes the functional requirements, such as API, inputs, expected outputs. It also

Table 2: Anonymization policies (a) and data transformations (b)

Vertex	Policy	$\langle$subject, *object*, *action*, *environment*, *transformation*$\rangle$
v_1,v_2,v_3	p_0	$\langle$ANY, *dataset*, *READ*, *ANY*, $t_0^p\rangle$
v_4,v_5	p_1	$\langle\{(service\_owner = dataset\_owner)\}, dataset, READ, ANY, t_0^p\rangle$
v_4,v_5	p_2	$\langle\{(service\_owner = partner(dataset\_owner))\}, dataset, READ, ANY, t_1^p\rangle$
v_6	p_3	$\langle\{(service\_region = dataset\_origin)\}, dataset, WRITE, ANY, t_0^p\rangle$
v_6	p_4	$\langle\{(service\_region = \{\text{“}NY\text{”}, \text{“}NH\text{”}\})\}, dataset, WRITE, ANY, t_1^p\rangle$
v_7	p_5	$\langle$ANY, *dataset*, *READ*, $\langle$environment=“risky”$\rangle$, $t_3^p\rangle$
v_7	p_6	$\langle$ANY, *dataset*, *READ*, $\langle$environment=“not_risky”$\rangle$, $t_4^p\rangle$

(a)

t_i^p	Level	Data Transformation
t_0^p	l_0	$anon(\varnothing)$
t_1^p	l_1	$anon(fname, lname)$
t_2^p	l_2	$anon(fname, lname, id, age)$
t_3^p	r_0	$aggregation(cluster = \infty)$
t_4^p	r_1	$aggregation(cluster = 10)$

(b)

specifies a set T^F of data transformation functions t_i^f, which can be triggered during the execution of the corresponding service s_i.

Function $t_i^f \in T^F$ can be: *i*) an empty function t_ϵ^f that applies no transformation or processing on the data; *ii*) an additive function t_a^f that expands the amount of data received, for example, by integrating data from other sources; *iii*) a transformation function t_t^f that transforms some records in the dataset without altering the domain; *iv*) a transformation function t_d^f (out of the scope of this work) that changes the domain of the data by applying, for instance, the PCA.

For simplicity but with no loss of generality, we assume that all candidate services meet functional annotation F and that $T^F = t^f$. As a consequence, all candidate services apply the same transformation to the data during the pipeline execution.

Example 3.1 (Pipeline Template). Let us consider the reference scenario introduced in Section 2.1. Fig. 2 presents an example of pipeline template consisting of five stages, each one annotated with a policy in Table 2.

The first stage consists of three parallel vertices v_1, v_2, v_3 for data collection. Data protection annotations $\lambda(v_1)$, $\lambda(v_2)$, $\lambda(v_3)$ refer to policy p_0 with an empty transformation t_0^p. Functional requirements F_1, F_2, F_3 prescribe a URI as input and the corresponding dataset as output.

The second stage consists of vertex v_4, merging the three datasets obtained at the first stage. Data protection annotation $\lambda(v_4)$ refers to policies p_1 and p_2, which apply different data transformations depending on the relation between the dataset and the

service owner. If the service owner is also the dataset owner (i.e., ($service\_owner = dataset\_owner$)), the dataset is not anonymized (t_0^p). If the service owner is a partner of the dataset owner (i.e., ($service\_owner = partner(dataset\_owner)$)), the dataset is anonymized at *level1* (t_1^p). If the service owner has no partner relationship with the dataset owner, no policy applies. Functional requirement F_4 prescribes n datasets as input and the merged dataset as output.

The third stage consists of vertex v_5 for data analysis. Data protection annotation $\lambda(v_5)$ refers to policies p_1 and p_2, as for the second stage. Functional requirement F_5 prescribes a dataset as input and the results of the data analysis as output.

The fourth stage consists of vertex v_6, managing data storage. Data protection annotation $\lambda(v_6)$ refers to policies p_3 and p_4, which apply different data transformations depending on the relation between the dataset and the service region. If the service region is the dataset origin (condition ($service\_region$=$dataset\_origin$) in p_3), the dataset is anonymized at level l_0 (t_0^p). If the service region is in a partner region (condition ($service\_region$={"NY","NH"}) in p_4), the dataset is anonymized at level l_1 (t_1^p). Functional requirement F_7 prescribes a dataset as input and the URI of the stored data as output.

The last stage consists of vertex v_7, responsible for data visualization. Data protection annotation $\lambda(v_7)$ refers to policies p_5 and p_6, which anonymize data according to the environment where the service is executed. A *risky* environment is defined as a region outside the owner or partner facility. If the environment is risky (p_5), the data are anonymized at level r_0 (t_3^p). If the environment is not risky (p_6), the data are anonymized at level r_1 (t_4^p). Functional requirement F_8 prescribes a dataset as input and data visualization interface (possibly in the form of JSON file) as output.

4 Pipeline Instance

A Pipeline Instance $G'(V', E, \lambda)$ instantiates a Pipeline Template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ by selecting and composing services according to data protection and functional annotations in the template. It is formally defined as follows.

Definition 5 (Pipeline Instance)**.** Let $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ be a pipeline template, a pipeline Instance $G'(V', E, \lambda)$ is an isomorphic directed acyclic graph where: *i*) $v'_r{=}v_r$; *ii*) for each vertex v_f modeling a parallel structure, there exists a corresponding vertex v'_f; *iii*) for each $v_i{\in}V_S$ annotated with policy P_i (label $\lambda(v_i)$) and functional description F_i (label $\gamma(v_i)$), there exists a corresponding vertex $v'_i{\in}V'_S$ instantiated with a service s'_i, such that:

1) s'_i satisfies data protection annotation $\lambda(v_i)$ in $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$;
2) s'_i satisfies functional annotation $\gamma(v_i)$ in $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$.

Condition 1 requires that each selected service s'_i satisfies the policy requirements P_i of the corresponding vertex v_i in the Pipeline Template, whereas Condition 2 is needed to preserve the process functionality, as it simply states that each service s'_i must satisfy the functional requirements F_i of the corresponding vertex v_i in the Pipeline Template.

We then define a *pipeline instantiation* function that takes as input a Pipeline Template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ and a set S^c of candidate services, and returns as output a Pipeline Instance $G'(V', E, \lambda)$. We note that S^c is partitioned in different set of services S^c_i, one for each vertex $v_i{\in}V_S$. Recall from Section 3.3 that all candidate services meet the functional annotation in the template, meaning that Condition 2 in Definition 5 is satisfied for all candidate services.

The Pipeline Instance is generated by traversing the Pipeline Template with a breadth-first search algorithm, starting from the root vertex v_r. Then, for each vertex

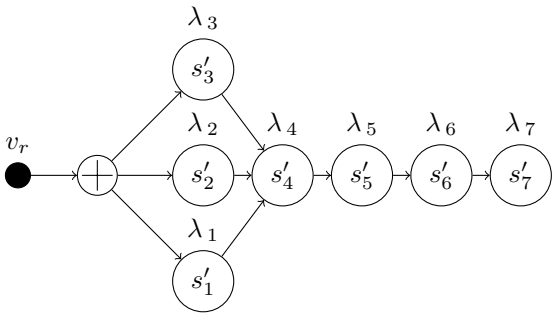


Fig. 3: Service composition instance

v_f in the pipeline template, the corresponding vertex v'_f is generated. Finally, for each vertex $v_i \in V_S$, a two-step approach is applied as follows.

1. *Filtering Algorithm* – It checks whether profile prf_j of each candidate service $s_j \in S_i^c$ satisfies at least one policy in P_i. If yes, service s_j is compatible, otherwise it is discarded. The filtering algorithm finally returns a subset $S'_i \subseteq S_i^c$ of compatible services for each vertex $v_i \in V_S$.
2. *Selection Algorithm* – The selection algorithm selects one service s'_i for each set S'_i of compatible services, which instantiates the corresponding vertex $v'_i \in V'$. There are many ways of choosing s'_i, we present our approach based on the maximization of data *quality* Q in Section 5.

When all vertices $v_i \in V$ in $G^{\lambda,\gamma}$ have been visited, the Pipeline Instance G' is generated, with a service instance s'_i for each $v'_i \in V'$. Vertex v'_i is still annotated with policies in P_i according to λ, because policies in P_i are evaluated and enforced at runtime, only when the pipeline instance is triggered and before any service is executed. In the case of policy evaluation returns *true*, data transformation $T^P \in P_i$ is applied, otherwise a default transformation that removes all data is applied.

Example 4.1 (Pipeline Instance). Let us consider a subset $\{v_5, v_6, v_7\}$ of the pipeline template $G^{\lambda,\gamma}$ in Example 3.1.

Table 3: Instance example

Vertex→Policy	Candidate	Profile	Filtering	Instance
$v_5 \rightarrow p_1$,p_2	s_{51}	service_owner = "CT"	✓	✓
	s_{52}	service_owner = "NY"	✓	✗
	s_{53}	service_owner = "CA"	✗	✗
$v_6 \rightarrow p_3$,p_4	s_{61}	service_region = "CA"	✗	✗
	s_{62}	service_region = "CT"	✓	✓
	s_{63}	service_region = "NY"	✓	✗
$v_7 \rightarrow p_5$,p_6	s_{71}	visualization_location = "CT_FACILITY"	✓	✓
	s_{72}	visualization_location = "CLOUD"	✓	✗

(a) Valid Instance example

Candidate	Ranking
s_{51}	1
s_{52}	2
s_{53}	–
s_{61}	–
s_{62}	1
s_{63}	2
s_{71}	1
s_{72}	2

(b) Best Quality Instance example

As presented in Table 3(a), each vertex is labeled with policies (column *Vertex→Policy*), and is associated with different candidate services (column *Candidate*) and corresponding profile (column *Profile*). The filtering algorithm matches each candidate service profile against the policies annotating the corresponding vertex (Table 2). It returns the set of services whose profile satisfies a policy (column *Filtering*): *i*) for vertex v_5, the filtering algorithm produces the set $S_1 = \{s_{51}, s_{52}\}$. Assuming that the dataset owner is "CT", the service profile of s_{51} matches p_1 and the one of s_{52} matches p_2. For s_{53}, there is no policy match and, thus, it is discarded; *ii*) for vertex v_6, the filtering algorithm returns the set $S'_2 = \{s_{62}, s_{63}\}$. Assuming that the dataset region is "CT", the service profile of s_{62} matches p_5 and the one of s_{63} matches p_6. For s_{61}, there is no policy match and, thus, it is discarded; *iii*) for vertex v_7, the filtering algorithm returns the set $S'_3 = \{s_{71}, s_{72}\}$. Since policy p_7 matches against any subject, the filtering algorithm keeps all services.

For each vertex v'_i, we select a matching service s'_j from S'_i and incorporate it into a valid instance. For instance, we select s_{51} for v_5; s_{62} for v_6, and s_{71} for v_7 as depicted in Table 3(a) (column *instance*). We note that to move from a valid to an optimal instance, it is mandatory to evaluate candidate services based on specific quality metrics that reflect their impact on data quality, as discussed in the following of this paper.

5 Maximizing the Pipeline Instance Quality

Our goal is to generate a pipeline instance with maximum quality, addressing data protection requirements throughout the pipeline execution. To this aim, we first discuss the quality metrics used to measure and monitor data quality, which guide the generation of the pipeline instance with maximum quality. Then, we prove that the problem of generating a pipeline instance with maximum quality is NP-hard (Section 5.2). Finally, we introduce a parametric heuristic (Section 5.3) designed to tackle the computational complexity associated with enumerating all possible combinations within a given set. The main objective of the heuristic is to approximate the optimal path for service interactions and transformations, especially within the realm of complex pipelines consisting of numerous vertices and candidate services. Our focus extends beyond identifying optimal combinations to include an understanding of the quality changes introduced during the transformation processes.

5.1 Quality Metrics

Ensuring data quality is mandatory to implement data pipelines that provide accurate results and decision-making along the whole pipeline execution. Quality metrics measure the data quality preserved at each step of the data pipeline, and can be classified as *quantitative* or *qualitative*. Quantitative metrics monitor the amount of data lost during data transformations to model the quality difference between datasets X and Y. Qualitative metrics evaluate changes in the properties of datasets X and Y. For instance, qualitative metrics can measure the changes in the statistical distribution of the two datasets.

In this paper, we use two metrics, one quantitative and one qualitative, to compare the input dataset X and the dataset Y obtained by enforcing data protection requirements (i.e., our policy-driven transformation described in Section 4) on X at

each step of the pipeline. We note that a complete taxonomy of possible metrics is outside the scope of this paper and will be the target of our future work.

5.1.1 Quantitative metric

We propose a quantitative metric M_J based on the Jaccard coefficient that assesses the similarity between two datasets. The Jaccard coefficient is defined as follows [1]:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

where X and Y are two datasets of the same size.

The coefficient is calculated by dividing the cardinality of the intersection of two datasets by the cardinality of their union. It ranges from 0 to 1, with 0 indicating no similarity (minimum quality) and 1 indicating complete similarity (maximum quality) between the datasets. It has several advantages. Unlike other similarity measures, such as Euclidean distance, it is not affected by the magnitude of the values in the dataset. It is suitable for datasets with categorical variables or nominal data, where the values do not have a meaningful numerical interpretation.

Metric M_J extends the Jaccard coefficient with weights that model the importance of each element in the dataset as follows:

$$M_J(X,Y) = \frac{\sum_{i=1}^{n} w_i(x_i \cap y_i)}{\sum_{i=1}^{n} w_i(x_i \cup y_i)}$$

where $x_i \in$X ($y_i \in$Y, resp.) is the i-th feature of dataset X (Y, resp.), and w_i the weight modeling the importance of the i-th feature.

It is computed by dividing the cardinality of the intersection of two datasets by the cardinality of their union, weighted by the importance of each feature in the datasets. It provides a more accurate measure of similarity.

5.1.2 Qualitative Metric

We propose a qualitative metric M_{JSD} based on the Jensen-Shannon Divergence (JSD) that assesses the similarity (distance) between the probability distributions of two datasets.

JSD is a symmetrized version of the KL divergence [2] and is applicable to a pair of statistical distributions only. It is defined as follows:

$$JSD(X,Y) = \frac{1}{2}\left(KL(X||M) + KL(Y||M)\right)$$

where X and Y are two distributions of the same size, and M=0.5*(X+Y) is the average distribution. JSD incorporates both the KL divergence from X to M and from Y to M.

To make JSD applicable to datasets, where each feature in the dataset has its own statistical distribution, metric M_{JSD} applies JSD to each column of the dataset. The obtained results are then aggregated using a weighted average, thus enabling the prioritization of important features that can be lost during the policy-driven transformation in Section 5, as follows:

$$M_{JSD} = 1 - \sum_{i=1}^{n} w_i \cdot \text{JSD}(x_i, y_i)$$

where $\sum_{i=1}^{n} w_i$=1 and each JSD(x_i, y_i) accounts for the Jensen-Shannon Divergence computed for the i-th feature in datasets X and Y. It ranges from 0 to 1, with 0 indicating no similarity (minimum quality) and 1 indicating complete similarity (maximum quality) between the datasets.

M_{JSD} provides a weighted measure of similarity, which is symmetric and accounts for the contribution from both datasets and specific features. It can compare the similarity of the two datasets, providing a symmetric and normalized measure that considers the overall data distributions.

5.1.3 Pipeline Quality

Metrics M_J and M_{JSD} contribute to the calculation of the pipeline quality Q as follows.

Definition 6 (*Pipeline Quality*)**.** Given a metric $M \in \{M_J, M_{JSD}\}$ modeling data quality, the pipeline quality Q is equal to $\sum_{i=1}^{|S|} M_{ij}$, with M_{ij} the value of the quality metric computed at each vertex $v'_i \in V'_S$ of the pipeline instance G' with respect to the service instance s'_j, with $1 \leq j < |S^c_i|$.

We also use the notation Q_{ij}, with $Q_{ij}=M_{ij}$, to specify the *quality* at vertex $v'_i \in V'_S$ of G' for service s'_j.

5.2 NP-Hardness of the Max-Quality Pipeline Instantiation Problem

The process of computing a pipeline instance (Definition 5) with maximum quality Q can be formally defined as follows.

Definition 7 (Max-Quality Problem)**.** Given a pipeline template $G^{\lambda,\gamma}$ and a set S^c of candidate services, find a max-quality pipeline instance G' such that:

- G' satisfies conditions in Definition 5,
- $\nexists$ a pipeline instance G'' that satisfies conditions in Definition 5 and such that quality $Q\ (G'') > Q\ (G')$, where $Q\ (\cdot)$ is the pipeline quality in Definition 6.

The Max-Quality problem is a combinatorial selection problem and is NP-hard, as stated by Theorem 5.1. However, while the overall problem is NP-hard, the filtering step of the process, is solvable in polynomial time. It can be done by iterating over each vertex and each service, checking if the service matches the vertex policy. This process takes polynomial time complexity $O(|V_S| * |S|)$.

Theorem 5.1. The Max-Quality problem is NP-Hard.

Proof: The proof is a reduction from the multiple-choice knapsack problem (MCKP), a classified NP-hard combinatorial optimization problem, which is a generalization of the simple knapsack problem (KP) [3]. In the MCKP problem, there are t mutually disjoint classes $N_1, N_2, \ldots, N_t$ of items to pack in some knapsack of capacity C, class N_i having size n_i. Each item $j \in N_i$ has a profit p_{ij} and a weight w_{ij}; the problem is to choose one item from each class such that the profit sum is maximized without having the weight sum to exceed C.

The MCKP can be reduced to the Max-Quality Pipeline Instantiation Process in polynomial time, with $N_1, N_2, \ldots, N_t$ corresponding to the sets of compatible services $S_1^c, S_2^c, \ldots, S_u^c$, with $t{=}u$ and n_i also the size of each set S_i^c. The profit p_{ij} of item $j \in N_i$ corresponds to quality Q_{ij} computed for each candidate service $s_j \in S_i^c$, while w_{ij} is uniformly 1 (thus, C is always equal to the cardinality of V_C). It is evident that the solution to one problem is also the solution to the other (and vice versa). Since the reduction can be done in polynomial time, the Max-Quality problem is also NP-hard.

Example 5.1 (Max-Quality Pipeline Instance)**.** We extend Example 4.1 with the selection algorithm in Section 4 built on pipeline quality Q. The selection algorithm is applied to the set S'_* of compatible services and returns three service rankings, one for each vertex v_4, v_5, v_6, according to quality metric M_J measuring the amount of preserved data after anonymization. The ranking is presented in Table 3(b), according to the transformation function in the corresponding policies. We assume that the more restrictive the transformation function (i.e., it anonymizes more data), the lower is the service position in the ranking. For example, s_{11} is ranked first because it anonymizes less data than s_{12} and s_{13}, that is, $Q_{11}{>}Q_{12}$ and $Q_{11}{>}Q_{13}$. The same applies for the ranking of s_{22} and s_{23}. The ranking of s_{31} and s_{32} is affected by the environment state at the time of the ranking. For example, if the environment where the visualization is performed is a CT facility, then s_{31} is ranked first and s_{32} second because the facility is considered less risky than the cloud, and $Q_{31}{>}Q_{32}$.

5.3 Heuristic

We design and implement a heuristic algorithm built on a *sliding window* for computing the pipeline instance maximizing quality Q. At each iteration i, a window of size $|w|$ selects a subset of vertices in the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$, from vertices at depth i to vertices at depth $|w|+i-1$. Service filtering and selection in Section 4 are then executed to maximize quality Q_w in window w. The heuristic returns as output the list of services instantiating all vertices at depth i. The sliding window w is then shifted by 1 (i.e., $i=i+1$) and the filtering and selection process executed until $|w|+i-1$ is equal to length l (max depth) of $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$, that is, the sliding window reaches the end of the template. In the latter case, the heuristic instantiates all remaining vertices and returns the pipeline instance G'. This strategy ensures that only services with low information loss are selected at each step, maximizing the pipeline quality Q.

The pseudocode of the heuristic algorithm is presented in Fig. 4. Function **SlidingWindowHeuristic** implements our heuristic; it takes the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ and the window size $|w|$ as input and returns the pipeline instance G' and corresponding metric M as output. Function **SlidingWindowHeuristic** retrieves the optimal service combination composing G', considering the candidate services associated with each vertex in $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ and the constraints (policies) in *verticesList*.

It iterates all sliding windows w step 1 until the end of the pipeline template is reached (**for cycle** in line 2). Adding the service(s) selected at step i to G' by function **SelectService** (definied in line 10).

Function **SelectService** takes as input index i representing the starting depth of the window and the corresponding window size $|w|$. It initializes the best combination of services to *empty* (line 11). It iterates through all possible combinations of services in the window using the Cartesian product of the service lists (**for cycle** in lines

INPUT
$G^{\lambda,\gamma}$: Pipeline Template
$|w|$: Window Size

OUTPUT
G': Pipeline Instance
M: Quality Metric

Sliding_Window_Heuristic ($G^{\lambda,\gamma}$, $|w|$)

1 */* For each window frame choose the best combination of services*/*
2 **for** i = 0 to l - $|w|$+ 1;
3 $G' = G' \cup$ Select_Service(j, $|w|$);
4 **endfor**;

5 */* Calculate the total quality metric*/*
6 **for** j = 0 to $|V'_S|$;
7 M=M+$M(s'_j)$;
8 **endfor**;

9 **return** G', M;

10 **Select_Service (j,**$|w|$**)**

11 G^*_w= best combination (*empty*);
12 */* Select the best combination of services*/*
13 **for** $G'_w \in \bigotimes_{k=j}^{j+|w|-1} verticesList[k]$
14 **if** M(G'_w) < M(G^*_w)
15 $G^*_w = G'_w$
16 **endfor**;

17 */* If it is the last window frame, return all services */*
18 **if** isLastWindowFrame()
19 **return** G^*_w
20 **else**
21 **return** $G^*_w[0]$

Fig. 4: Pseudocode of the sliding window heuristic algorithm.

13-16). If the current combination has quality metric M(G'_w) higher than the best

quality metric M(G^*_w), current combination G'_w updates the best combination G^*_w (lines 14-15).

Function **SelectService** then checks whether it is processing the last window (line 18). If yes, it returns the best combination G^*_w (line 19). Otherwise, it returns the first service in the best combination G^*_w (line 21).

Within each window, function **SlidingWindowHeuristic** finally iterates through the selected services to calculate the total quality metric M (**for cycle** in lines 6-8). This metric is updated by summing the quality metrics of the selected services. The function concludes by returning the best pipeline instance G' and the corresponding quality metric M (line 9).

6 Experiments

We experimentally evaluated the performance and quality of our methodology (heuristic algorithm in Section 5.3), and compared it against the exhaustive approach in Section 5.2. In the following, Section 6.1 presents the simulator and experimental settings used in our experiments; Section 6.2 analyses the performance of our solution in terms of execution time; Section 6.3 discusses the quality of the best pipeline instance generated by our solution according to the metrics M_J and M_{JSD} in Section 5.1.

6.1 Testing Infrastructure and Experimental Settings

Our testing infrastructure is a Swift-based simulator of a service-based ecosystem, including service execution, selection, and composition. The simulator first defines the pipeline template as a sequence of vertices, with l the length of the pipeline template, and defines the size $|w|$ of the sliding window, such that $|w| \leq l$. We recall that alternative vertices are modeled in different pipeline templates, while parallel vertices are not considered in our experiments since they only add a fixed execution time that is negligible and do not affect the performance and quality of our solution.

Each vertex is associated with a (set of) policy that applies a filtering transformation that remove a given percentage of data.

The simulator then starts the instantiation process. At each step i, it selects the subset $\{v_i,\ldots,v_{|w|+i-1}\}$ of vertices with their corresponding candidate services, and generates all possible service combinations. The simulator calculates quality Q for all combinations and instantiates v_i with service s'_i from the optimal combination with maximum Q. The window is shifted by 1 (i.e., i=i+1) and the instantiation process restarts. When the sliding window reaches the end of the pipeline template, that is, $v_{|w|+i-1}$=v_l, the simulator computes the optimal service combination and instantiates the remaining vertices with the corresponding services. Figure 5 shows an example of a simulator execution with i=2 and $|w|$=3. Subset $\{v_2,v_3,v_4\}$ is selected, all combinations generated, and corresponding quality Q calculated. Optimal service combination $\{s'_{11},s'_{22},s'_{31}\}$ is retrieved and v'_2 in the pipeline instance instantiated with s'_{11}.

The simulator defines dependencies between filtering transformations made by candidate services at consecutive vertices of the pipeline template. To this aim, it assigns a dependency rate to each service s_i modeling the amount of the filtering transformation done at s_i that overlaps the one at s_{i-1}. For example, let us consider the pairs of services (s_{11},s_{21}) and (s_{11},s_{22}) with the following configurations: *i)* service s_{11} introduces a filtering transformation that removes the 20% of the dataset, *ii)* service s_{21} introduces a filtering transformation that removes 10% of the dataset and has dependency rate equal to 1, meaning that the filtering transformation made by s_{21} completely overlaps the one made by s_{11}, *iii)* service s_{22} introduces a filtering transformation that removes 5% of the dataset and has dependency rate equal to 0.5, meaning that the filtering transformation made by s_{22} overlaps half of the one made by s_{11}. Jaccard Metric $M_{J_{21}}$=0.8 at service s_{21}; $M_{J_{22}}$=0.75 at s_{22}, showing how dependencies can the pipeline quality and, in turn, the instantiation process.

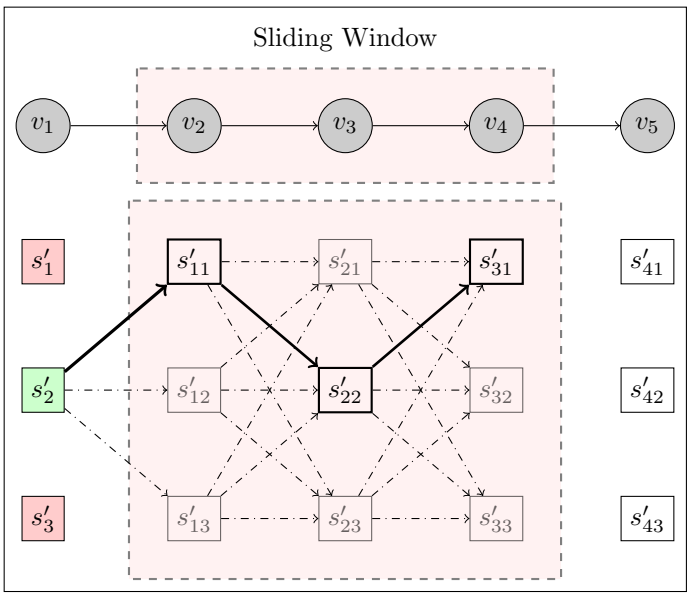
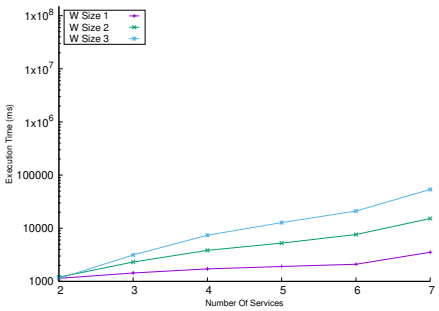


Fig. 5: Execution example of the sliding window heuristic using v=5, s=3, $|w|$=3 at i=1 step.

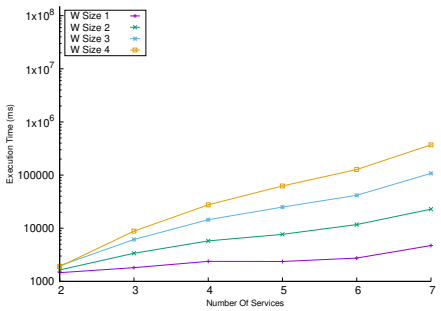
Our experiments have been run on a virtual machine equipped with a Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz CPU and 32GB RAM. Each experiment was repeated 10 times and the results averaged to improve the reliability of findings.

6.2 Performance

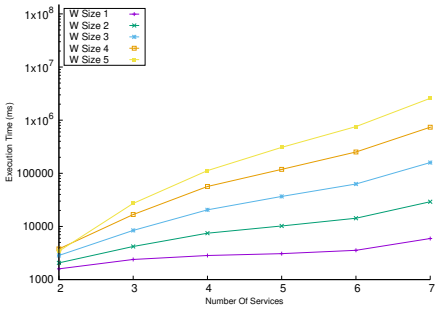
We first measured the performance (execution time) of our exhaustive and heuristic solutions by varying the number of vertices in the pipeline template from 2 to 7 and the number of services per vertex from 2 to 7. Section 6.2 presents our results for both the exhaustive and heuristic solutions. The exhaustive approach is able to provide the optimal solution for all configurations, but its execution time grows exponentially with the number of vertices and services, making it impractical for large instances. For $|w|$from 1 to 3 (step 1), we observed a substantial reduction in execution time, with the heuristic always able to produce an instance in less than $\approx 2.7 \times 10^5 ms$. The worst heuristic performance (7 vertices, 7 services, $|w|$=6) is $\approx 3.8 \times 10^7 ms$ is still one



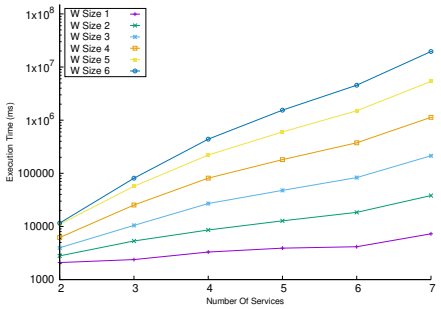
(a) 3 vertices



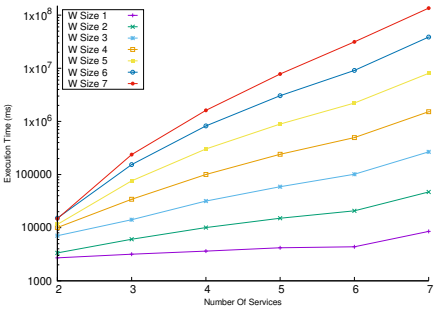
(b) 4 vertices



(c) 5 vertices



(d) 6 vertices



(e) 7 vertices

order of magnitude lower than the best exhaustive performance (7 vertices, 7 services, $|w|$=7) $\approx 1.35 \times 10^8 ms$.

6.3 Quality

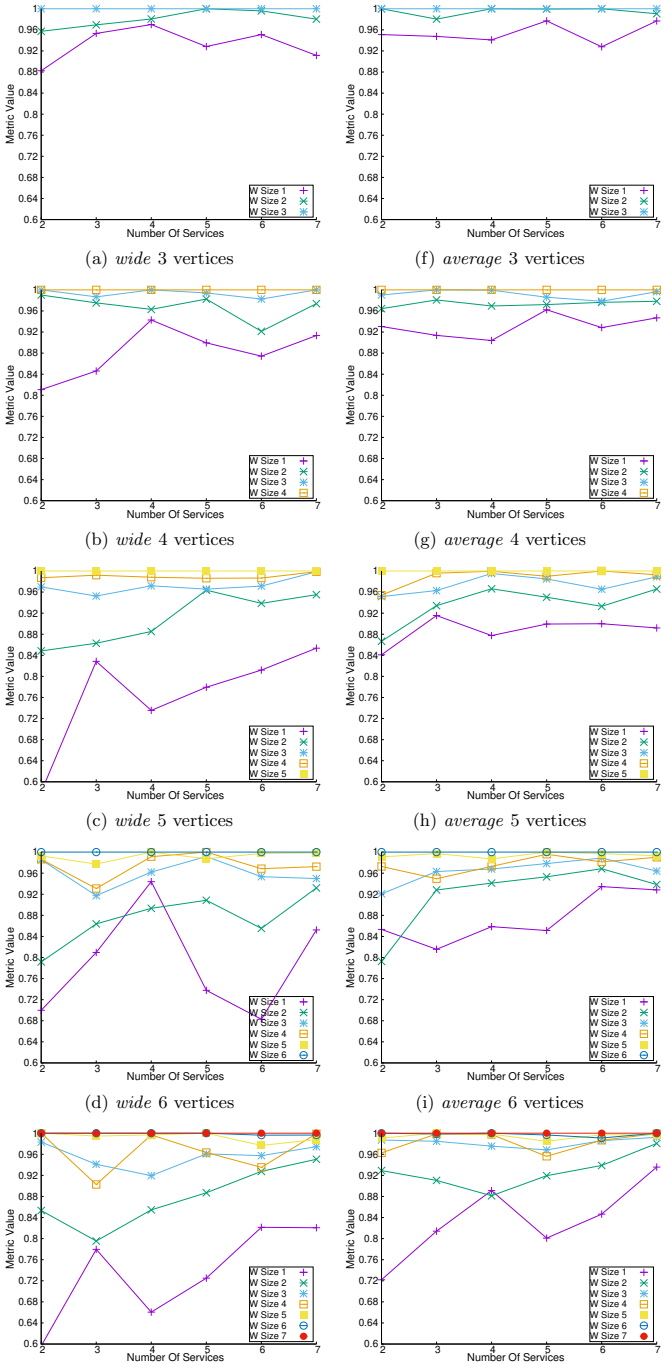
We finally evaluated the quality of our heuristic algorithm with different $|w|$ comparing, where possible, its results with the optimal solution retrieved by executing the

exhaustive approach. The quality Q of the heuristic has been normalized between 0 and 1 by dividing it by the quality Q^* retrieved by the exhaustive approach.

We run our experiments varying: *i)* the length l of the pipeline template in [3,7], that is, the depth of the pipeline template as the number of vertices composed in a sequence, *ii)* the window size $|w|$ in [1,l], and *iii)* the number of candidate services for each vertex in the pipeline template in [2, 7]. Each vertex is associated with a (set of) policy that applies a filtering transformation that either remove a percentage of data in $[0.5, 0.8]$ (*average*) or in $[0.2, 1]$ (*wide*).

Fig. 7 present our quality results using metric M_J in Section 5.1 for settings *wide* and *average*. In general, we observe that the quality of our heuristic approach increases as the window size increases, providing a quality comparable to the exhaustive approach when the window size $|w|$ approaches the length l of the pipeline template.

When considering setting *wide*, the greedy approach ($|w|$=1) provides good results on average (0.71, 0.90), while showing substantial quality oscillations in specific runs: between 0.882 and 0.970 for 3 vertices, 0.810 and 0.942 for 4 vertices, 0.580 and 0.853 for 5 vertices, 0.682 and 0.943 for 6 vertices, 0.596 and 0.821 for 7 vertices. This same trend emerges when the window size is $<l/2$, while it starts approaching the optimum when the window size is $\geq l/2$. For instance, when $|w|$=l-1, the quality varies between 0.957 and 1.0 for 3 vertices, 0.982 and 1.0 for 4 vertices, 0.986 and 0.998 for 5 vertices, 0.977 and 1.0 for 6 vertices, 0.996 and 1.0 for 7 vertices.



(a) *wide* 3 vertices (f) *average* 3 vertices

(b) *wide* 4 vertices (g) *average* 4 vertices

(c) *wide* 5 vertices (h) *average* 5 vertices

(d) *wide* 6 vertices (i) *average* 6 vertices

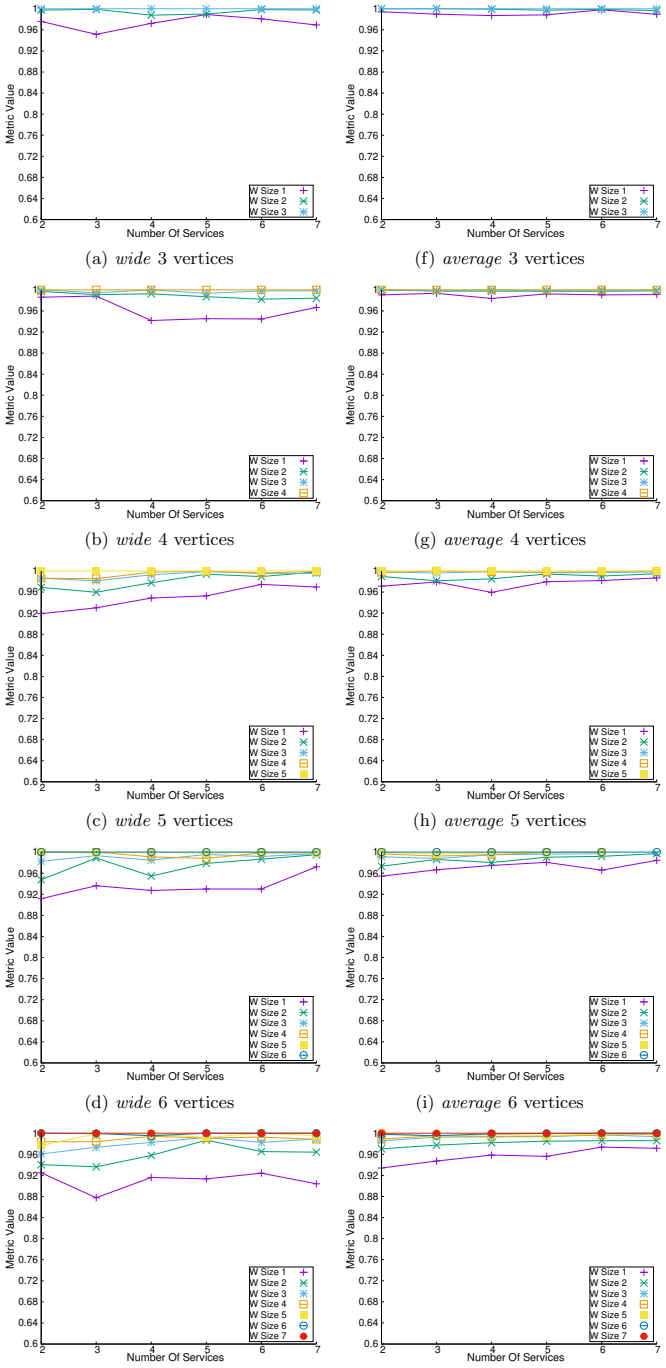
(e) *wide* 7 vertices (j) *average* 7 vertices

Fig. 7: Evaluation of Quality Using the *Qualitative* Metric in a *wide* (Figs. 7a to 7e) and *average* (Figs. 7f to 7j) Profile Configuration.

When considering setting *average*, the heuristic algorithm still provides good results, limiting the quality oscillations observed for setting *wide* and approaching the quality of the exhaustive also for lower window sizes. The greedy approach ($|w|$=1) provides good results on average (from 0.842 to 0.944), as well as in specific runs: between 0.927 and 0.978 for 3 vertices, 0.903 and 0.962 for 4 vertices, 0.840 and 0.915 for 5 vertices, 0.815 and 0.934 for 6 vertices, 0.721 and 0.935 for 7 vertices. When $|w|=l$-1, the quality varies between 0.980 and 1.0 for 3 vertices, 0.978 and 1.0 for 4 vertices, 0.954 and 1 for 5 vertices, 0.987 and 1.0 for 6 vertices, 0.990 and 1.0 for 7 vertices.

Fig. 8 present our quality results using metric M_{JSD} in Section 5.1 for settings *wide* and *average*, respectively.

When considering setting *wide*, the greedy approach ($|w|$=1) provides good results on average (0.92, 0.97), limiting oscillations observed with metric M_J; for instance, the quality varies between 0.951 and 0.989 for 3 vertices, 0.941 and 0.988 for 4 vertices, 0.919 and 0.974 for 5 vertices, 0.911 and 0.971 for 6 vertices, 0.877 and 0.924 for 7 vertices. The worst quality results are obtained with window size equal to 1, while the oscillations are negligible when the window size is ¿2. For instance, when $|w|=l$-2, the quality varies between, 0.982 and 0.996 for 4 vertices, 0.981 and 0.998 for 5 vertices, 0.988 and 1.0 for 6 vertices, 0.976 and 0.999 for 7 vertices. When $|w|=l$-1, the quality varies between 0.987 and 0.998 for 3 vertices, 0.993 and 1.0 for 4 vertices, 0.985 and 0.999 for 5 vertices, 0.997 and 1.0 for 6 vertices, 0.995 and 1.0 for 7 vertices.



(a) *wide* 3 vertices (f) *average* 3 vertices

(b) *wide* 4 vertices (g) *average* 4 vertices

(c) *wide* 5 vertices (h) *average* 5 vertices

(d) *wide* 6 vertices (i) *average* 6 vertices

(e) *wide* 7 vertices (j) *average* 7 vertices

Fig. 8: Evaluation of Quality Using the *Qualitative* Metric in a *wide* (Figs. 8a to 8e) and *average* (Figs. 8f to 8j) Profile Configuration.

When considering setting *average*, the greedy approach ($|w|$=1) provides results similar to setting *wide*. On average, quality varies from 0.920 to 0.969, limiting oscillations; for instance, the quality varies between 0.951 and 0.989 for 3 vertices, 0.942 and 0.988 for 4 vertices, 0.919 and 0.975 for 5 vertices, 0.912 and 0.972 for 6 vertices, 0.878 and 0.925 for 7 vertices. The *average* configuration provides even tighter quality oscillations than the *wide* configuration. Notably, the poorest quality outcomes are observed when the window size is set to 1. Conversely, these oscillations become negligible when the window size exceeds 1 in configurations with three and four vertices, and when it exceeds 2 in configurations involving five, six, and seven vertices. For instance, when $|w|$=3, the quality varies between 0.993 and 1 for 4 vertices, 0.981 and 0.998 for 5 vertices, 0.982 and 997 for 6 vertices, 0.960 and 0.991 for 7 vertices.

Our results suggest that the proposed heuristic well approximates the results obtained by the exhaustive approach. While larger window sizes generally lead to better performance, there exists a breakpoint where the balance between window size and performance is optimized. Beyond this point, the incremental gains in metric values may not justify the additional computational burden or complexity introduced by larger windows. It is worth noting that lower window sizes are more unstable, especially with setting *wide*, meaning that the quality varies significantly among different configurations. This effect stabilizes with higher window sizes (e.g., $|w| \geq l/2$).

7 Related Work

Given the breadth of topics covered, the related work is discussed separately to provide a more detailed and organized review. In Section 7.1, we address the issue of the lack of consensus on the definition of data quality and, consequently, on data quality metrics when applying data protection transformations. In Section 7.2, we examine existing data governance solutions that tackle the problem of data protection when sharing

data among different services. In Section 7.3, we review the literature on QoS (Quality of Service) service selection.

7.1 Data quality and data protection

Data quality is a widely studied research topic across various communities and perspectives, such as the database community or when evaluating privacy preserving data mining techniques. In the context of big data, data quality primarily refers to the extent to which big data meets the requirements and expectations of its intended use, encompassing various dimensions and characteristics to ensure the data is reliable, accurate, and valuable for analysis and decision-making. Specifically, accuracy denotes the correctness of the data, ensuring it accurately represents the real-world entities and events it models.

With the increasing need to protect sensitive data, the notion of data quality has expanded to include a broader concept of accuracy, particularly in terms of the proximity of a sanitized value to the original value. This shift has emphasized the necessity of metrics to assess the quality of data resulting from anonymization processes. Various data quality metrics have been proposed in existing literature, including generalized information loss ($GenILoss$), discernability metric, minimal distortions, and average equivalence class size (C_{AVG}), which may either have broad applicability or be tailored to specific data scenarios [4–6]. However, there is currently no metric that is widely accepted by the research community. The main challenge with data quality is its relative nature: its evaluation typically depends on the context in which the data is used and often involves both objective and subjective parameters [7, 8]. A common consideration across all contexts is that accuracy is closely related to the information loss resulting from the anonymization strategy: the lower the information loss, the higher the data quality. In our scenario, we have opted for two generic metrics rooted in data

loss assessment - one quantitative and one qualitative. Nonetheless, our framework and heuristic are designed to be modular and flexible, accommodating the chosen metric.

Another critical consideration is the integration of data quality throughout the entire data lifecycle. Historically, the focus within the database management research community has predominantly been on enhancing the quality of source data, neglecting to ensure data quality across the whole processing pipeline, or the resulting outcomes. In [9], the authors propose a holistic quality management model that briefly considers data quality during processing, primarily in the context of prerequisites for preprocessing tasks (e.g., data refinement and enhancement through cleaning processes). In contrast, our approach diverges from this notion of assessing data quality solely during preprocessing, instead advocating for its evaluation at each stage of the big data pipeline. We build upon the evaluation conducted in [10] within a specific case study, wherein data protection transformations were implemented at each stage, albeit with only one candidate service.

7.2 Data governance and data protection

As organizations realize practical benefits and significant value from big data, they also acknowledge the limitations of current big data ecosystems, particularly regarding data governance and data protection, and the need for privacy-aware systems enforcing sensitive data protection. Recently, both industry and academic communities have begun to investigate the issue, recognizing the need of new security requirements [11] and the importance of addressing the conflict between the need to share and the need to protect information [12–16], from a data governance perspective [17, 18], and, more in general, to ensure compliance of the Big Data pipeline with generic non-functional requirements [19, 20].

Various proposals address data protection by implementing robust access control on big data platforms. Some approaches are platform-specific, tailored to single systems like MongoDB or Hadoop, and leverage the native access control features of these platforms [21–25]. Other approaches focus on specific databases, such as NoSQL or graph databases, or specific types of analytical pipelines [26–28]. However, these solutions often rely on query rewriting mechanisms, resulting in high complexity and low efficiency. Some solutions are designed for specific scenarios, such as federated cloud environments, edge microservices, or IoT, and lack the flexibility to adapt to multiple contexts [29, 30].

The most similar to our approach are platform-independent solutions that, like ours, adopt Attribute-Based Access Control (ABAC) [31] as a common underlying model, given its ability to support highly flexible and dynamic forms of data protection to business-critical data. They have the advantage of being more general than platform-specific solutions. However, the currently available platforms either model resources to be accessed as monolithic files (e.g., Microsoft DAC) or lack scalability. A relevant work by Hu et al. [32] introduced a generalized access control model for big data processing frameworks that can be extended to the Hadoop environment. However, their work discusses the issues only from a high-level architectural perspective and does not offer a tangible solution or address data quality issues. Another relevant work by Xue et al. [33] proposes a solution based on purpose-aware access control [34], focusing on Apache Spark. Our definition of pipeline template addresses the various challenges by allowing to express the security policies at the right level of granularity, considering individual services in the pipeline. It can also be easily mapped onto specific platforms, such as Apache-based systems, as we have demonstrated in [35].

7.3 Service Selection based on data quality

The selection and composition of services is a recent topic in the era of big data, originating from the Web service scenario but facing additional challenges due to the volume and velocity of data, as well as to the heterogeneity of services, domains, and hosting infrastructures. In the context of big data, Quality of Service (QoS) is particularly crucial for service selection: as organizations leverage vast amounts of data to enhance decision-making and operational efficiency, it is imperative to choose appropriate services for processing, analyzing, and interpreting this data. In particular, the selection process must account for both functional and non-functional requirements, including performance, scalability, reliability, and security standards. Despite its critical nature, security is often one of the least considered metrics in service selection [36]. Even when security is taken into account, it is not always evaluated in relation to data quality.

Related works include [37], where Web services are composed according to the security requirements of both service requestors and providers. However, the range of expressible requirements is limited, such as the type of encryption algorithm or authentication method (e.g., SSO), and data sanitization is not considered. Thus, the selection algorithm is just a matching rather than a ranking with respect to a security metrics.

Another relevant study [38] implements a certification-based service selection process, ranking services according to their certified non-functional properties and corresponding user requirements. In this approach, certified services are assumed to be functionally equivalent, offering the same functionality while meeting users' functional requirements.

The most related work to ours is [36], where the authors address the challenges of big service composition, particularly QoS and security issues. Similarly to what we do with our pipeline template, they define a quality model for big services by extending

the traditional QoS model of Web services to include "big data"-related characteristics, and Quality of Data (QoD) attributes, such as completeness, accuracy, and timeliness. In order to address security issues, in their model, each service is assigned an L-Severity level [39] that represents the potential severity of data leakages when consuming its data chunks. Their approach aims to select the optimal composition plan that not only maximizes QoS and QoD attributes such as timeliness (TL), completeness (CP), and consistency (CS), but it also minimizes L-Severity (LS), data sources and communication costs.

8 Conclusions

In the realm of distributed data service pipelines, managing pipelines while ensuring both data quality and data protection presents numerous challenges. This paper proposed a framework specifically designed to address this dual concern. Our data governance model employs policies and continuous monitoring to address data security and privacy challenges, while preserving data quality, in service pipeline generation. The key point of the framework is in its ability to annotate each element of the pipeline with specific data protection requirements and functional specifications, then driving service pipeline construction. This method enhances compliance with regulatory standards and improves data quality by preserving maximum information across pipeline execution. Experimental results confirmed the effectiveness of our sliding window heuristic in addressing the computationally complex NP-hard service selection problem at the basis of service pipeline construction. Making use of a realistic dataset, our experiments evaluated the framework's ability to sustain high data quality while ensuring robust data protection, which is essential for pipelines where both data utility and privacy must coexist. To fully understand the impact of dataset selection on

the retrieved quality and to ensure heuristic robustness across various scenarios, further investigation is planned for our future work. Future work will then explore deeper insights into the applicability of our heuristics across different scenarios.

9 Declarations

9.1 Ethics approval and consent to participate

Not applicable

9.2 Consent for publication

Not applicable

9.3 Availability of data and materials

All data and materials available at https://github.com/SESARLab/Big-Data-Access-Control-Extension

9.4 Competing interests

The authors declare that they have no competing interests.

9.5 Funding

Research supported, in parts, by *i)* project "BA-PHERD - Big Data Analytics Pipeline for the Identification of Heterogeneous Extracellular non-coding RNAs as Disease Biomarkers", funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.1: "Fondo Bando PRIN 2022" (CUP G53D23002910006), *ii)* project MUSA - Multilayered Urban Sustainability Action - project, funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and

creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037), *iii)* project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NextGenerationEU, *iv)* Università degli Studi di Milano under the program "Piano di Sostegno alla Ricerca". Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

9.6 Authors' contributions

Marco Anisetti (M.A.) and Claudio A. Ardagna (C.A.A.) jointly conceived the original idea and provided guidance on the research direction. C.A.A., Chiara Braghin (C.B.), and Antongiacomo Polimeno (A.P.) developed the theoretical framework. A.P. was also responsible for conducting the experiments and drafting all the manuscript, under the supervision of M.A., C.A.A., C.B. All authors discussed the results, contributed to revisions of the manuscript, and approved the final version for publication.

9.7 Acknowledgements

Not applicable. No additional support was received from any individuals not listed as authors.

References

[1] Rahman, M., Hassan, M.R., Buyya, R.: Jaccard index based availability prediction in enterprise grids. Procedia Computer Science **1**(1), 2707–2716 (2010) https://doi.org/10.1016/j.procs.2010.04.304 . ICCS 2010

[2] Fuglede, B., Topsoe, F.: Jensen-shannon divergence and hilbert space embedding. In: International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings. IEEE. https://doi.org/10.1109/isit.2004.1365067 . http://dx.doi.org/10.1109/ISIT.2004.1365067

[3] Kellerer, H., Pferschy, U., Pisinger, D.: Multidimensional Knapsack Problems, pp. 235–283. Springer, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24777-7_9

[4] Majeed, A., Lee, S.: Anonymization techniques for privacy preserving data publishing: A comprehensive survey. IEEE Access **9**, 8512–8545 (2021) https://doi.org/10.1109/ACCESS.2020.3045700

[5] Fung, B.C.M., Wang, K., Fu, A.W.-C., Yu, P.S.: Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques, 1st edn. Chapman & Hall/CRC, New York (2010). https://doi.org/10.1201/9781420091502

[6] Sharma, A., Singh, G., Rehman, S.: A review of big data challenges and preserving privacy in big data. In: Kolhe, M.L., Tiwari, S., Trivedi, M.C., Mishra, K.K. (eds.) Advances in Data and Information Sciences, pp. 57–65. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0694-9_7

[7] Wang, R.Y., Strong, D.M.: Beyond accuracy: What data quality means to data consumers. Journal of Management Information Systems **12**(4), 5–33 (1996) https://doi.org/10.1080/07421222.1996.11518099

[8] Tayi, G.K., Ballou, D.P.: Examining data quality. Commun. ACM **41**(2), 54–57 (1998) https://doi.org/10.1145/269012.269021

[9] Taleb, I., Serhani, M.A., Dssouli, R.: Big data quality: A survey. In: 2018 IEEE International Congress on Big Data (BigData Congress), pp. 166–173 (2018). https://doi.org/10.1109/BigDataCongress.2018.00029

[10] Polimeno, A., Mignone, P., Braghin, C., Anisetti, M., Ceci, M., Malerba, D., Ardagna, C.: Balancing Protection and Quality in Big Data Analytics Pipelines. Big Data (2024) https://doi.org/10.1089/big.2023.0065

[11] Colombo, P., Ferrari, E.: Access control technologies for big data management systems: literature review and future trends. Cybersecurity **2**(1), 3 (2019) https://doi.org/10.1186/s42400-018-0020-9

[12] Geetha, P., Naikodi, C., Setty, S.L.N.: Design of big data privacy framework—a balancing act. In: Jain, V., Chaudhary, G., Taplamacioglu, M.C., Agarwal, M.S. (eds.) Advances in Data Sciences, Security and Applications, pp. 253–265. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0372-6_19

[13] van den Broek, T., van Veenstra, A.F.: Governance of big data collaborations: How to balance regulatory compliance and disruptive innovation. Technological Forecasting and Social Change **129**, 330–338 (2018) https://doi.org/10.1016/j.techfore.2017.09.040

[14] Ahlbrandt, J., Brammen, D., Majeed, R., Lefering, R., Semler, S., Thun, S., Walcher, F., R ohrig, R.: Balancing the need for big data and patient data privacy–an it infrastructure for a decentralized emergency care research database. Studies in health technology and informatics **205**, 750–754 (2014) https://doi.org/10.3233/978-1-61499-432-9-750

[15] Hotz, V., Bollinger, C., Komarova, T., Manski, C., Moffitt, R., Nekipelov, D., Sojourner, A., Spencer, B.: Balancing data privacy and usability in the federal statistical system. Proceedings of the National Academy of Sciences **119** (2022) https://doi.org/10.1073/pnas.2104906119

[16] Creese, S., Hopkins, P., Pearson, S., Shen, Y.: Data protection-aware design for cloud services. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing, pp. 119–130. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_11

[17] Al-Badi, A., Tarhini, A., Khan, A.I.: Exploring Big Data Governance Frameworks. Procedia Computer Science **141**, 271–277 (2018) https://doi.org/10.1016/j.procs.2018.10.181

[18] Aissa, M.M.B., Sfaxi, L., Robbana, R.: DECIDE: A New Decisional Big Data Methodology for a Better Data Governance. In: Proc. of EMCIS, vol. 402. Dubai, EAU, pp. 63–78 (2020). https://doi.org/10.1007/978-3-030-63396-7_5 . Springer

[19] Anisetti, M., Bena, N., Berto, F., Jeon, G.: A devsecops-based assurance process for big data analytics. In: Proc. of IEEE ICWS 2022, Barcelona, Spain (2022). https://doi.org/10.1109/ICWS55610.2022.00017

[20] Ardagna, C.A., Bena, N., Hebert, C., Krotsiani, M., Kloukinas, C., Spanoudakis, G.: Big data assurance: An approach based on service- level agreements. Big Data **11** (2023) https://doi.org/10.1089/big.2021.0369

[21] Rathore, M.M., Paul, A., Ahmad, A., Anisetti, M., Jeon, G.: Hadoop-based intelligent care system (hics) analytical approach for big data in iot. ACM Transactions on Internet Technology **18**(1), 8–1824 (2017) https://doi.org/10.1145/3108936

[22] Anisetti, M., Ardagna, C., Bellandi, V., Cremonini, M., Frati, F., Damiani, E.:

Privacy-aware big data analytics as a service for public health policies in smart cities. Sustainable cities and society **39**, 68–77 (2018) https://doi.org/10.1016/j.scs.2017.12.019

[23] Awaysheh, F.M., Alazab, M., Gupta, M., Pena, T.F., Cabaleiro, J.C.: Next-generation big data federation access control: A reference model. Future Generation Computer Systems **108**, 726–741 (2020) https://doi.org/10.1016/j.future.2020.02.052

[24] Gupta, M., Patwa, F., Sandhu, R.: An Attribute-Based Access Control Model for Secure Big Data Processing in Hadoop Ecosystem. In: Proceedings of the Third ACM Workshop on Attribute-Based Access Control. ABAC'18, pp. 13–24. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3180457.3180463

[25] Gupta, M., Patwa, F., Sandhu, R.: Object-Tagged RBAC Model for the Hadoop Ecosystem. In: Livraga, G., Zhu, S. (eds.) Data and Applications Security and Privacy XXXI, pp. 63–81. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61176-1_4

[26] Chabin, J., Ciferri, C.D.A., Halfeld-Ferrari, M., Hara, C.S., Penteado, R.R.M.: Role-Based Access Control on Graph Databases. In: Bureš, T., Dondi, R., Gamper, J., Guerrini, G., Jurdziński, T., Pahl, C., Sikora, F., Wong, P.W.H. (eds.) Proc. of SOFSEM, pp. 519–534. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67731-2_38

[27] Gupta, E., Sural, S., Vaidya, J., Atluri, V.: Enabling Attribute-based Access Control in NoSQL Databases. IEEE Transactions on Emerging Topics in Computing, 1–15 (2022) https://doi.org/10.1109/TETC.2022.3193577

[28] Huang, L., Zhu, Y., Wang, X., Khurshid, F.: An Attribute-Based Fine-Grained Access Control Mechanism for HBase. In: Hartmann, S., Küng, J., Chakravarthy, S., Anderst-Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) Database and Expert Systems Applications, pp. 44–59. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-27615-7_4

[29] Preuveneers, D., Joosen, W.: Towards Multi-party Policy-based Access Control in Federations of Cloud and Edge Microservices. In: Proc. of EuroS&PW, pp. 29–38 (2019). https://doi.org/10.1109/EuroSPW.2019.00010

[30] Matos, E., Tiburski, R.T., Amaral, L.A., Hessel, F.: Providing Context-Aware Security for IoT Environments Through Context Sharing Feature. In: Proc. of IEEE TrustCom/BigDataSE, pp. 1711–1715 (2018). https://doi.org/10.1109/TrustCom/BigDataSE.2018.00257

[31] Hill, R.C., Lockhartr, H.: eXtensible Access Control Markup Language (XACML) Version 3.0. OASIS Standard (2013). https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[32] Hu, V., Grance, T., Ferraiolo, D., Kuhn, D.: An Access Control Scheme for Big Data Processing. In: Proc. of 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 1–7. IEEE, Miami, USA (2014). https://doi.org/10.4108/icst.collaboratecom.2014.257649

[33] Xue, T., Wen, Y., Luo, B., Zhang, B., Zheng, Y., Hu, e., Li, Y., Li, G., Meng, D.: GuardSpark++: Fine-Grained Purpose-Aware Access Control for Secure Data Sharing and Analysis in Spark. In: Proc. of ACM Annual Computer Security Applications Conference. ACSAC'20, pp. 582–596. ACM, Online (2020). https://doi.org/10.1145/3427228.3427640

[34] Byun, J.-W., Li, N.: Purpose based access control for privacy protection in relational database systems. The VLDB Journal **17**(4), 603–619 (2008) https://doi.org/10.1007/11408079_2

[35] Marco, A., A., A.C., Chiara, B., Ernesto, D., Antongiacomo, P., Alessandro, B.: Dynamic and Scalable Enforcement of Access Control Policies for Big Data, pp. 71–78. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3444757.3485107

[36] Sellami, M., Mezni, H., Hacid, M.S.: On the use of big data frameworks for big service composition. Journal of Network and Computer Applications **166**, 102732 (2020) https://doi.org/10.1016/j.jnca.2020.102732

[37] Carminati, B., Ferrari, E., K. Hung, P.C.: Security conscious web service composition. In: 2006 IEEE International Conference on Web Services (ICWS'06), pp. 489–496 (2006). https://doi.org/10.1109/ICWS.2006.115

[38] Anisetti, M., Ardagna, C.A., Bena, N.: Multi-dimensional certification of modern distributed systems. IEEE Transactions on Services Computing **16**(3), 1999–2012 (2023) https://doi.org/10.1109/TSC.2022.3195071

[39] Vavilis, S., Petković, M., Zannone, N.: Data leakage quantification. In: Atluri, V., Pernul, G. (eds.) Data and Applications Security and Privacy XXVIII, pp. 98–113. Springer, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43936-4_7