TBA

Marco Anisetti *Senior Member, IEEE,*, Claudio A. Ardagna *Senior Member, IEEE,* Chiara Braghin, Ernesto Damiani *Senior Member, IEEE,*, Antongiacomo Polimeno

Abstract—The conflict between the need of protecting and sharing data is hampering the spread of big data applications. Proper security and privacy assurance is required to protect data owners, while proper data access and sharing are fundamental to implement smart big data solutions. In this context, access control systems assume a central role for balancing the need of data protection and sharing. However, given the software and technological complexity of big data ecosystems, existing solutions are not suitable because they are neither general nor scalable, and do not support a dynamic and collaborative environment. In this paper, we propose an access control system that enforces access to data in a distributed, multi-party big data environment. It is based on data annotations and secure data transformations performed at ingestion time. We show the feasibility of our approach with a case study in a smart city domain using an Apache-based big data engine.

Index Terms—Access Control, Big Data, Data Transformation, Data Ingestion

1 Introduction

TBW

2 Motivations

Data quality is a largely studied research topic. The database management research community mainly focused on increasing the quality of the source data rather than guaranteeing data quality along the whole processing pipeline or the quality of outcomes built on data. In [36] a survey on big data quality is proposed mentioning the well known categories of big data quality grouped by intrinsic, contextual representational and accessibility categories. It also presents an holistic quality management model where the importance of data quality during processing is just mentioned in terms of requirements for the pre-processing job (e.g., data enhancement due to cleaning jobs). In this paper we depart from this idea on data quality at pre processing time only measuring it at each step of the big data pipeline.

2.1 Background

2.2 Data Protection

Research on data governance and protection focuses on the definition of new approaches and techniques aimed to protect the security and privacy of big data (e.g., CIA triad), as well as managing their life cycle with security and privacy in mind. Often, the research community is targeting specific security and privacy problems, resulting in a proliferation of solutions and tools, which are difficult to integrate in a coherent framework. Many solutions have been developed to protect the users' identity (e.g., anonymity [43], pseudonimity [28], k-anonymity [33]), to guarantee data confidentiality and integrity (e.g., encryption [37], differential privacy [18], access control [34], [39]), and to govern data sharing and analysis (e.g., data lineage [44], ETL/ELT ingestion [41]).

This project proposal focuses on access control, an approach adopted to protect access to data from unauthorized users from the very beginning of ICT systems. Coming to big data, most of the current solution leverage on Attribute-Based Access Control (ABAC) [20] to manage policy rules adaptable at run time using attributes. In general, these systems must be instantiated on the specific scenario of interest and are ineffective when complexity increase due to the computational overhead and introduction of delays in policy enforcement. There are also database-centric approaches that focus on specific databases such as noSQL databases or graph databases, or specific types of analytical pipelines such [12], [15], [21]. However, these solutions are widely based on query rewriting mechanisms leading to high complexity and low efficiency. Finally, some solutions are scenario-specific (federate cloud, edge microservices or IoT) and lack the generality needed to adapt to multiple contexts [25], [29]. The closest approach to this project proposal is the work of Hu et al. [19], introducing a generalized access control model for big data processing frameworks, which can be extended to the Hadoop environment. However, the paper discusses the issues only from a high-level architectural point of view, without discussing a tangible solution. Another relevant work is by Xue et al. [46]. They propose a solution based on the notion of purpose-aware access control [10] that, although focusing only on Apache Spark, recognizes the need of a generalized approach to deal with access control in analytics pipelines.

An effective data governance and protection approach cannot avoid its integration within state-of-the-art big data infrastructures. In fact, as organizations see practical results and significant value in the usage of big data, they also recognize the limits of current big data ecosystems with respect to data governance and data protection. Recently,

- *M. Anisetti, C.A. Ardagna, E. Damiani, are with the Dipartimento di Informatica, Università degli Studi di Milano, Milano, Italy. E. Damiani is also with.*
  E-mail: {firstname.lastname}@unimi.it

both industry and academic communities started to investigate the issue, both from a data governance perspective [5], [6] or recognizing the need of new security requirements [13]. Although Attribute Based Access Control (ABAC) [20] is currently adopted in big data projects as a common underlying model given its ability to support highly flexible and dynamic forms of data protection to business-critical data, the dynamic and decentralized nature of big data asks for new solutions. Actual solutions are neither general nor scalable, since they are either platform-dependent or coarse-grained [13]. Platform-specific approaches are designed for single systems only (e.g., MongoDB, Hadoop) and leverage on native access control features of the platform [7], [31]. Some recent proposals, like Federated Access Control Reference Model (FACRM) [8] or [16], [17], are specifically tailored to the Apache Hadoop stack. On the other hand, platform-independent approaches 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.

Finally, the success of big data and the increasing central role of data in our everyday life increased the attention also by institutions and public administrations resulting in new regulations and guidelines. The possibility of using data indiscriminately by analyzing and sharing them has led to the emergence of specific and stringent regulations such as the General Data Protection Regulation (GDPR) [4]. GDPR is the first and best known of a series of regulations that aim to achieve this goal. GDPR enforced in 2018, replaced the inadequate Data Protection Directive 95/46/EC. GDPR was intended to raise awareness of companies and consumers on security, privacy and the right to be forgotten. The enormous growth of machine learning technologies and their pervasive diffusion, has led to the idea that artificial intelligence was also a field to be regulated. On 8 April 2019, the High-Level Expert Group on AI presented Ethics Guidelines for Trustworthy Artificial Intelligence, which proposes the development of reliable AI, that is: lawful, robust and ethical [1]. Finally, the problem of guaranteeing data sovereignty clearly emerged and is leading EU commission activities [27]. In this context, project Gaia-X, presented in October 2019 by the German and French Ministries of Economic Affairs, gathered more than 5,000 participants in one year. The project aims to satisfy a widespread need of European countries to regain possession of the sovereignty of their data, facilitate the creation of interconnected data spaces that comply with compliance criteria, open-source software, providing certification tools [3]. Another example of work on the data sovereignty topic is the development of the European Data Governance Act that is currently underway and aims to regulate and encourage data sharing within the European Union [2].

2.3 Data Processing

The need of analyzing and extracting value from data emerged from the very beginning of ICT, and is at the basis of a vast research area that includes database technologies [26], data mining [11], [22], big data analytics [40] and machine learning [30]. This project proposal focuses on machine learning in the context of modern big data infrastructures. Research on machine learning focused on many aspects of data learning and analysis, and is horizontal to multiple domains and research areas. Machine learning has evolved from simple mining based on logistic regression and naive Bayes to Deep Learning, via Decision tree and linear regression. Deep learning is a subset of Neural Network techniques, which owe their success to the ever increasing computational power and massive amount of data available. Deep Learning copes well with the distribution of computations reaching excellent performances thanks to the work done in the parallelization of machine learning processing [14], [42], [45].

At the same time, substantial research and development effort has been put on the infrastructures supporting data analysis, which underwent a gradual evolution starting from a monolithic model towards a model based on micro and nano services [23]. The spread of the cloud computing and containerized systems allowed these systems to be more resilient, easier to develop, and adaptive, while increasing their complexity.[1] With the spread of big data, new tools and infrastructures have been developed for data ingestion, storage, and analysis. The main challenge to be addressed was the inability for a single system to handle such huge amount of data; this resulted in the spread of solutions based on a distributed file system [9], a system that permits to share data across multiple machines (made in general of commodity hardware), making possible for the user to read and write data without perceiving this distribution. Over the years these systems have seen more and more refinements and expansions, such as, resource managers to manage the distribution of resources in the network of vertexes (YARN) [24], SQL-Like systems that support access following standard RDBMS (HIVE, Presto, Trino) [35], [38]. Finally, other software components (for example Apache Spark [32]) have been created to meet the ever increasing needs for data access and analysis requested by machine learning technologies.

3 System Model and Service Pipeline

Big data is highly dependent on cloud-edge computing, which makes extensive use of multitenancy. Multitenancy permits sharing one instance of infrastructures, platforms or applications by multiple tenants to optimize costs. This leads to common scenarios where a service provider offers subscription-based analytics capabilities in the cloud, or a single data lake is accessed by multiple customers. Big data pipelines then mix data and services which belong to various organizations, posing a serious risk of potential privacy and security violations.

We propose a data governance framework tailored to contemporary data-driven pipelines, which aims to limit the privacy and security risks. The primary objective of this framework is to facilitate the assembly of data processing services, with a central focus on the selection of those services that optimize data quality, while upholding privacy and security requirements.

In the following of this section, we present our system model (Section 3.1) and our reference scenario (Section 3.2).

1. Today the Apache big data ecosystem counts more than 50 tools.

3.1 System Model

In today's data landscape, the coexistence of data quality and data privacy is critical to support high-value services and pipelines. The increase in data production, collection, and usage has led to a split in scientific research priorities. First, researchers are exploring methods to optimize the usage of valuable data. Here, ensuring data quality is vital, and requires accuracy, reliability, and soundness for analytical purposes. Second, there is a need to prioritize data privacy and security. This involves safeguarding confidential information and complying with strict privacy regulations. These two research directions are happening at the same time, though there are not many solutions that find a good balance between them.

Our approach seeks to harmonize these objectives by establishing a data governance framework that balances privacy and data quality. It implements a system model that is composed of the following parties:

Service, a software distributed by a **service provider** that performs a specific task according to access control privileges on data;

Pipeline, a sequence of connected services that collect, prepare, process, and analyze data in a structured and automated manner. We distinguish between a **pipeline template** that acts as a skeleton, specifying the structure of the pipeline and the (non-)functional requirements driving service selection and composition, and a **pipeline instance** instantiating the template with services according to the specified requirements;

Data Governance Policy, a structured set of privacy guidelines, rules, and procedures regulating data access and protection;

User that executes an analytics pipeline on the data. We assume that the data target of the analytics pipeline are ready for analysis, that is, they underwent a preparatory phase addressing issues such as missing values, outliers, and formatting discrepancies. This ensures that the data are in an optimal state for subsequent analysis.

The user first selects a pipeline template among a set of functionally-equivalent templates according to its non-functional requirements. It then instantiates the template in a pipeline instance. To this aim, for each component service in the template, it retrieves a set of candidate services that satisfy the functional requirements of the component service. Candidate services are filtered to retrieve a list of compatible services that comply with the policies specified in the template.

Compatible services are ranked based on their ability to retain the maximum amount of information (*data quality* in this paper), while maintaining a minimum level of privacy; 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. It is important to note that our data governance approach builds on the following assumption: *upholding a larger quantity of data is linked to better data quality.* While this assumption is not true in all settings, it correctly represents many real-world scenarios. We leave a solution that departs from this assumption to our future work.

3.2 Service Pipeline and Reference Scenario

We consider a service-based environment where a service pipeline is designed to analyze data. We define a service pipeline as a graph defined as follows.

Definition 3.1 (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 $v_r \in V$, a vertex $v_i \in V_S$ for each service s_i, an additional vertex $v_f \subset V$ for each parallel ($\oplus$) structure modeling the contemporary execution (*fork*) of services.

We note that $\{v_r,v_f\} \cup V_S = V$, vertices v_f model branching for parallel structures, and root v_r possibly represents the orchestrator. We also note that, 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.

Our reference scenario considers a service pipeline analyzing a dataset of individuals detained in Department of Correction facilities in the state of Connecticut while awaiting trial [**?**]. In particular, the user, a member of the Connecticut Department of Correction (DOC), seeks to compare admission trends in Connecticut prisons with DOCs in New York and New Hampshire. We assume DOCs to be partners and share data according to their privacy policies. The user's preferences align with a predefined pipeline template 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 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.

We note that the template requires the execution of the entire service within the Connecticut Department of Correction. If the data needs to be transmitted beyond the boundaries of Connecticut, data protection measures must be implemented. A visual representation of the flow is presented in Figure 1. Table 1 presents a sample of the adopted dataset.[2] 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 have extended this dataset by introducing randomly generated first and last names.

4 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 3.1 with: *i)* data protection annotations expressing transformations on data to enforce data protection requirements, *ii)* functional annotations expressing data manipulations carried out during services execution. These annotations permit to

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

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...	...

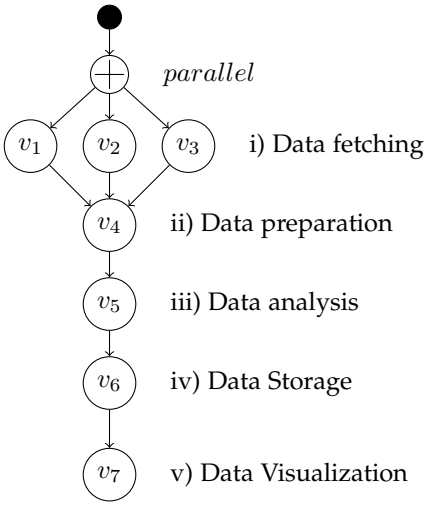


Fig. 1: Reference Scenario

implement an advanced data lineage, tracking the entire data lifecycle by monitoring changes arising from functional service execution and data protection requirements.

In the following, we first introduce the annotated service pipeline, called pipeline template (Section 4.1). We then present functional annotations (Section 4.3) and data protection annotations (Section 4.2). We finally provide an example of a pipeline template (Section 5.1).

4.1 Pipeline Template Definition

Given the service pipeline in Definition 3.1, we use annotations to express data protection requirements to be enforced on data and functional requirements on 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*) a labeling 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*) a labeling 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 4.1 (Pipeline Template)**.** Given a service pipeline G(V,E), a pipeline template $G^{\lambda,\gamma}$(V,E,λ,γ) is a direct acyclic graph with two labeling functions:

i λ that assigns a label $\lambda(v_i)$, corresponding to a set P_i of policies p_j to be satisfied by service s_i represented by v_i, for each vertex $v_i \in V_S$;

ii γ that assigns a label $\gamma(v_i)$, corresponding to the functional description F_i of service s_i represented by v_i, for each vertex $v_i \in V_S$.

We note that, at this stage, the template is not yet linked to any service. We also note that policies $p_j \in P_i$ annotated with $\lambda(v_i)$ are ORed, meaning that the access decision is positive if at least one policy p_j is evaluated to *true*. An example of pipeline template is depicted in Fig. 2

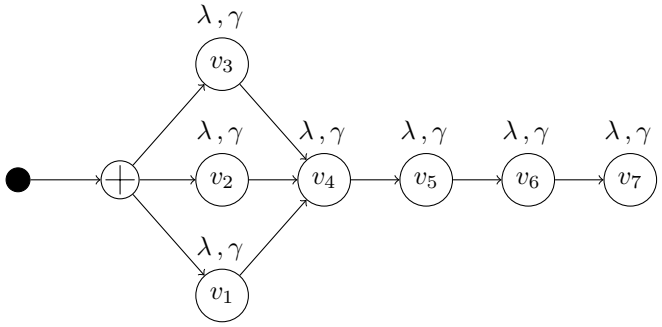


Fig. 2: Pipeline Template

4.2 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 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 4.2 (Policy Condition)**.** A *Policy Condition pc* is a Boolean expression of the form (*attr_name* op *attr_value*), with op$\in\{<,>,=,\neq,\leq,\geq\}$, *attr_name* an attribute label, and *attr_value* the corresponding attribute value.

An access control policy then specifies who (*subject*) can access what (*object*) with action (*action*), in a specific context (*environment*) and under specific obligations (*data transformation*), as formally defined below.

Definition 4.3 (Policy)**.** A *policy* $p \in P$ is 5-uple $<$*subj*, *obj*, *act*, *env*, $T^P>$, where:

Subject *subj* defines a service s_i issuing an access request to perform an action on an object. It is of the form $<$*id*, $\{pc_i\}>$, where *id* defines a class of services (e.g., classifier), and $\{pc_i\}$ is a set of *Policy Conditions* on

the subject, as defined in Definition 4.2. For instance, <*service*,{(classifier = "SVM")}> refers to a service providing a SVM classifier. We note that *subj* can also specify conditions on the service owner, such as, <*service*,{(owner_location = "EU")}> and on the service user, such as, <*service*,{(service_user_role = "DOC Director")}>.

Object *obj* defines any data whose access is governed by the policy. It is of the form <*type*, $\{pc_i\}$>, where: *type* defines the type of object, such as a file (e.g., a video, text file, image, etc.), a SQL or noSQL database, a table, a column, a row, or a cell of a table, and $\{pc_i\}$ is a set of *Policy Conditions* defined on the object's attributes. For instance, <*dataset*,{(region = CT)}> refers to a dataset whose region is Connecticut.

Action *act* defines any operations that can be performed within a big data environment, from traditional atomic operations on databases (e.g., CRUD operations varying depending on the data model) to coarser operations, such as an Apache Spark Direct Acyclic Graph (DAG), Hadoop MapReduce, an analytics function call, or 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, emergency. It is a set $\{pc_i\}$ of *Policy Conditions* as defined in Definition 4.2. For instance, <*env*,{(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*, which must be enforced before any access to data. Transformations focus on data protection, as well as compliance to regulations and standards, in addition to simple format conversions. For instance, let us define three transformations that can be applied to the Table 1: i) *level0* (t_0^p): no anonymization is carried out; ii) *level1* (t_1^p): The data has been partially anonymized with only the first name and last name being anonymized; iii) *level2* (t_2^p): The data has been fully anonymized with the first name, last name, identifier, and age being anonymized.

Access control policies $p_j \in P_i$ annotating vertex v_i in a pipeline template $G^{\lambda,\gamma}$ are used to filter out those candidate services $s \in S^c$ that do not match data protection requirements. Specifically, each policy $p_j \in P_i$ is evaluated to verify whether a candidate service $s \in S^c$ for vertex v_i is compatible with data protection requirements in P_i ($\lambda(v_i)$). Policy evaluation matches the profile *prf* of candidate service $s \in S^c$ with the policy conditions in each $p_j \in P_i$. If the credentials and declarations, defined as a set of attributes in the form (*name*, *value*), in the candidate service profile fails to meet the policy conditions, meaning that no policies p_j are evaluated to *true*, the service is discarded; otherwise it is added to the set S' of compatible service, which is used in Section 5 to generate the pipeline instance G'. No policy enforcement is done at this stage.

4.3 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 on the corresponding service s_i, such as API, inputs, expected outputs. It also specifies a set T^F of data transformation functions t_i^f, possibly triggered during execution of the connected service s_i.

Each $t_i^f \in T^F$ can be of different types as follows: *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, PCA or K-means.

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 data during execution.

4.4 Example

Let us consider our reference scenario in Section 3.1. 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 requirement F_1, F_2, F_3 prescribes a URI as input and the corresponding dataset as output.

The second stage consists of vertex v_4, merging the three datasets obtained stage 1. 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 service owners. If the service owner is also the dataset owner ($\langle service\_owner = dataset\_owner \rangle$), the dataset is not anonymized (t_0^p). We note that if the service owner has no partner relationship with the dataset owner, no policies apply. If the service owner is a partner of the dataset owner ($\langle service\_owner = partner(dataset\_owner) \rangle$), the dataset is anonymized at level l_1 (t_1^p). 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 stage 2. Functional requirement F_5 prescribes a dataset as input and the results of the data analysis as output. Data protection annotation $\lambda(v_5)$ refers to policy p_4 with data transformation t_2^p, that is, anonymization level l_2 to prevent personal identifiers from entering into the machine learning algorithm/model. Functional requirement F_6 prescribes a dataset as input, and the trained model and a set of inferences as output.

The fourth stage consists of vertex v_6, managing data storage. Data protection annotation $\lambda(v_6)$ refers to policies p_5 and p_6, which apply different data transformations depending on the relation between the dataset and service region. If the service region is the dataset origin ($\langle service\_region = dataset\_origin" \rangle$) ($p_5$), the dataset is

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

Vertex	Policy	⟨subject, $object$, $action$, $environment$, $transformation$⟩
v_1,v_2,v_3	p_0	⟨ANY, $dataset$, $READ$, ANY, t_0^p⟩
v_4,v_5	p_1	⟨⟨$service\_owner = dataset\_owner$⟩, $dataset$, $READ$, ANY, t_0^p⟩
v_4,v_5	p_2	⟨⟨$service\_owner = partner(dataset\_owner)$⟩, $dataset$, $READ$, ANY, t_1^p⟩
v_6	p_5	⟨⟨$service\_region = dataset\_origin$⟩, $dataset$, $WRITE$, ANY, t_0^p⟩
v_6	p_6	⟨⟨$service\_region =$ "{NY, NH}"⟩, $dataset$, $WRITE$, ANY, t_1^p⟩
v_7	p_7	⟨ANY, $dataset$, $READ$,environment = risky, t_3^p⟩
v_7	p_8	⟨ANY, $dataset$, $READ$,environment = not_risky, t_4^p⟩

(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)

anonymized at level l_1 (t_1^p). If the service region is in a partner region (⟨$service, region = NY, NH$"⟩) (p_6), the dataset is anonymized at level l_2 (t_2^p). Functional requirement F_7 prescribes a dataset as input and the URI of the stored data as output.

The sixth and last stage consists of vertex v_7, responsible for data visualization. Data protection annotation $\lambda(v_7)$ refers to policies p_7 and p_8, 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_7), the data are anonymized at level r_0 (t_3^p). If the environment is not risky (p_8), 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.

5 Pipeline Instance

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

Definition 5.1 (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 $v \in V_\otimes \cup V_\oplus$, there exists a corresponding vertex $v' \in V'_\otimes \cup V'_\oplus$; *iii*) for each $v_i{\in}V_S$ annotated with policy P_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, with a specific set of services S_i^c for each vertex $v_i{\in}V_S$, and returns as output a Pipeline Instance $G'(V', E, \lambda)$. Recall from Section 4.3 that all candidate services meet the functional annotation in the template, meaning that Condition 2 in Definition 5.1 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 $v \in V_\oplus \bigcup V_\otimes$ in the pipeline template, the corresponding vertex $v' \in V'_\oplus \bigcup V'_\otimes$ is generated. Finally, for each vertex $v_i{\in}V_S$, a two-step approach is applied as follows.

1) *Filtering Algorithm* – The filtering algorithm checks if the profile prf_j of each candidate service $s_j{\in}S_i^c$ satisfies the policies $p_k{\in}P_i$ corresponding to $\lambda(v_i)$. If prf_j satisfies at least one policy, 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 and instantiates the corresponding vertex $v'_i{\in}V'$ with it. There are many ways of choosing s'_i, we present our approach based on the minimization of quality loss in Section **??**.

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

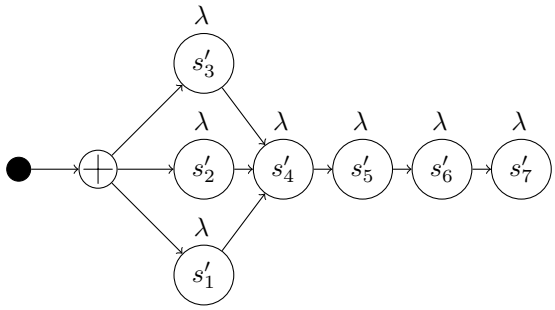


Fig. 3: Service composition instance

5.1 Example

TBD (mettere un esempio in cui non sia stato scelto l'ottimo che invece cercheremo e metteremo come esempio nella prossima sezione)

Example 5.1 (Pipeline Instance). Let us consider a subset $\{v_5, v_6, v_7\}$ of the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ in Section 5.1.

Each vertex is associated with three candidate services, each having a profile. The filtering algorithm matches each candidate service's profile with the policies annotating the corresponding vertex. It returns the set of services whose profile matches a policy. *i*) For v_6, the filtering algorithm produces the set $S'_1 = \{s_{61}, s_{62}\}$; assuming that the dataset owner is "CT", the service profile of $s61$ matches p_1 and the one of $s62$ matches p_2. For $s63$, there is no policy match and, thus, it is discarded. *ii*) For v_7, the filtering algorithm returns the set $S'_2 = \{s_{72}, s_{73}\}$; assuming that the dataset region is "CT", the service profile of $s72$ matches p_5 and the one of $s73$ matches p_6. For $s71$, there is no policy match and, thus, it is discarded. *iii*) For v_8, the filtering algorithm returns the set $S'_3 = \{s_{81}, s_{82}, s_{83}\}$. Since policy p_7 matches with any subjects, the filtering algorithm does not discard any service.

For each vertex, we could select the initial matching servic from each set S'_* and incorporate it into the instance. For instance, for v_6, we select $s61$; for v_7, $s72$ is chosen, and for v_8, $s81$ is the preferred option. The instance thus formulated is depicted in Table 3. It is imperative to acknowledge that this instance is valid it satisfies all the policies in the pipeline template. However, it does not represent the optimal instance achievable. o determine the optimal instance, it is essential to evaluate services based on specific quality metrics that reflect their impact on data quality. In the next sections, we will introduce the metrics that we use to evaluate the quality of services and the results of the experiments conducted to evaluate the performance of our approach.

6 Maximizing the Pipeline Instance Quality

Our goal is to generate a pipeline instance with maximum quality, which addresses data protection requirements with the minimum amount of information loss across the pipeline execution. To this aim, we first discuss the role of some metrics (**??**) to specify and measure data quality, and describe the ones used in the paper. Then, we prove that the problem of generating a pipeline instance with maximum quality is NP-hard (Section 6.2). Finally, we present a parametric heuristic (Section 6.3) tailored to address the computational complexity associated with enumerating all possible combinations within a given set. The primary aim of the heuristic is to approximate the optimal path for service interactions and transformations, particularly within the landscape of more complex pipelines composed of numerous vertexes and candidate services. Our focus extends beyond identifying optimal combinations, encompassing an understanding of the quality changes introduced during the transformation processes.

6.1 Quality Metrics

Ensuring data quality is mandatory to implement data pipelines that provide accurate results and decision-making along the whole pipeline execution. To this aim, quality metrics evaluate the quality loss introduced 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 as 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 provide two metrics, one quantitative and one qualitative, that compare the input dataset X and dataset Y generated by enforcing data protection requirements (i.e., our policy-driven transformation in Section [**?**]) on X at each step of the data pipeline.

6.1.1 Jaccard coefficient

The Jaccard coefficient is a quantitative metric that can be used to measure the difference between the elements in two datasets. It is defined as:

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

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

The Jaccard coefficient is computed by dividing the cardinality of the intersection of two sets by the cardinality of their union. It ranges from 0 to 1, where 0 indicates no similarity and 1 indicates complete similarity between the datasets.

The Jaccard coefficient 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.

6.1.2 Jensen-Shannon Divergence

The Jensen-Shannon divergence (JSD) is a quantitative metric that can be used to measure the dissimilarity between the probability distributions of two datasets. It is a symmetrized version of the KL divergence [**?**] .

The JSD between X and Y is defined as:

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

where X and Y are two datasets 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. It provides a balanced measure of dissimilarity that is symmetric and accounts for the contribution from both datasets. JSD can compare the dissimilarity of the two datasets, providing a symmetric and normalized measure that considers the overall data distribution. It provides a more comprehensive understanding of the dissimilarity between X and Y, taking into account the characteristics of both datasets.

We note that our metrics can be applied either to the entire dataset or to specific features only. The features can be assigned with equal or varying importance, providing a weighted version of the metrics, thus enabling the prioritization of important features that might be possibly lost during the policy-driven transformation in Section [**?**]. A complete taxonomy of possible metrics is however outside the scope of this paper and will be the target of our future work.

TABLE 3: Instance example

Vertex→Policy	Candidate	Profile	Filtering	Instance	Candidate	Ranking
$v_4 \rightarrow p_1$,p_2	$s41$	service_owner = "CT"	✓	✓	$s41$	1
	$s42$	service_owner = "NY"	✓	✗	$s42$	2
	$s43$	service_owner = "CA"	✗	✗	$s43$	–
$v_5 \rightarrow p_5$,p_6	$s51$	service_region = "CA"	✗	✗	$s51$	–
	$s52$	service_region = "CT"	✓	✓	$s52$	1
	$s53$	service_region = "NY"	✓	✗	$s53$	2
$v_6 \rightarrow p_7$,p_8	$s61$	visualization_location = "CT_FACILITY"	✓	✓	$s61$	1
	$s62$	visualization_location = "CLOUD"	✓	✗	$s62$	2

6.2 NP-Hardness of the Max Quality Pipeline Instantiation Process

se lo definiamo in maniera formale come il problema di trovare un'istanza valida in accordo alla definizione di istanza tale che non ne esiste una con un loss piu' piccolo?

Definition 6.1 (Max Quality Pipeline Instantiation Process)**.** Given $dtloss_i$ the value of the quality metric computed after applying the transformation of the policy matching the service selected to instantiate vertex $v_i \in V_S$, the Max quality Pipeline Instantiation Process is the case in which the *pipeline instantiation* function returns a Pipeline Instance where the $dtloss_i$ sum is maximized.

The Max Quality Pipeline Instantiation Process is a combinatorial selection problem and is NP-hard, as stated by Theorem 6.1. However, while the overall problem is NP-hard, there is a component of the problem that is solvable in polynomial time: matching the profile of each service with the node policy. This can be done by iterating over each node and each service, checking if the service matches the node's policy. This process would take $O(|N| * |S|)$ time. This is polynomial time complexity.

Theorem 6.1. The Max Quality Pipeline Instantiation Process 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) []. 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 $S_1^c, S_1^c, \ldots, S_u^c$, $t{=}u$ and n_i the size of S_i^c. The profit p_{ij} of item $j \in N_i$ corresponds to $dtloss_{ij}$ 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).

Since the reduction can be done in polynomial time, our problem is also NP-hard. (non è sufficiente, bisogna provare che la soluzione di uno e' anche soluzione dell'altro)

Example 6.1 (Max-Quality Pipeline Instance)**.** Let us consider a subset $\{v_5, v_6, v_7\}$ of the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ in Section 5.1. Each vertex is associated with three candidate services, each having a profile. The filtering algorithm matches each candidate service's profile with the policies annotating the corresponding vertex. It returns the set of services whose profile matches a policy.

The comparison algorithm is then applied to the set of services S'_* and it returns a ranking of the services. The ranking is based on the amount of data that is anonymized by the service. The ranking is listed in Table 3 and it is based on the transformation function of the policies, assuming that a more restrictive transformation function anonymizes more data affecting negatively the position in the ranking. For example, $s11$ is ranked first because it anonymizes less data than $s12$ and $s13$. The ranking of $s22$ and $s23$ is based on the same logic. Finally, the ranking of $s31$, $s32$ is influenced by the environment state at the time of the ranking. For example, if the environment in which the visualization is performed is a CT facility, then $s31$ is ranked first and $s32$ second; thus because the facility is considered a less risky environment than the cloud.

6.3 Heuristic

HO RIVISTO IL PARAGRAFO VELOCEMENTE GIUSTO PER DARE UN'INDICAZIONE. DOBBIAMO USARE LA FORMALIZZAZIONE E MAGARI FORMALIZZARE ANCHE LO PSEUDOCODICE. We design and implement a heuristic algorithm for computing the pipeline instance maximizing data quality. Our heuristic is built on a *sliding window* and aims to minimize information loss according to quality metrics. At each step, a set of vertexes in the pipeline template $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$ is selected according to a specific window w=[i,j], where i and j are the starting and ending depth of window w. Service filtering and selection in Section 5 are then executed to minimize information loss in window w. The heuristic returns as output the list of services instantiating vertexes at depth i. A new window w=[i+1,j+1] is considered until j+1 is equal to the max depth of $G^{\lambda,\gamma}(V, E, \lambda, \gamma)$, that is the window reaches the end of the template. This strategy ensures that only services with low information loss are selected at each step, minimizing the overall information loss. Pseudo-code for the sliding window algorithm is presented in Algorithm 1.

```
1 selectedServices = empty
2 for i from 0 to length(serviceCombinations):
```

```
3       minMetricCombination = None
4       minMetric = +∞
5       M = JSD or J //JSD or Jaccard coefficient
6
7       for j from i to i + windowSize:
8           totalMetric = 0
9           for service in serviceCombinations[j]:
10              totalMetric += M(service)
11              currentMetric = totalMetric / length(
    serviceCombinations[j])
12          if currentMetric < minMetric:
13              minMetric = currentMetric
14              minMetricCombination =
    serviceCombinations[j]
15              firstService = serviceCombinations
    [j][0]
16      add firstService to instance
17  return instance
18
19
```

Listing 1: Sliding Window Heuristic with Selection of First Service from Optimal Combination

The pseudocode implemets function *SlidingWindowHeuristic*, which takes a sequence of vertexes and a window size as input and returns a set of selected vertexes as output. The function starts by initializing an empty set of selected vertexes (line 3). Then, for each node in the sequence (lines 4–12), the algorithm iterates over the vertexes in the window (lines 7–11) and selects the node with the lowest metric value (lines 9-11). The selected node is then added to the set of selected vertexes (line 12). Finally, the set of selected vertexes is returned as output (line 13).

We note that a window of size 1 corresponds to the *greedy* approach, while a window of size N, where N represents the total number of vertexes, corresponds to the *exhaustive* method.

The utilization of heuristic in service selection can be enhanced through the incorporation of techniques derived from other algorithms, such as Ant Colony Optimization or Tabu Search. By integrating these approaches, it becomes feasible to achieve a more effective and efficient selection of services, with a specific focus on eliminating paths that have previously been deemed unfavorable.

We experimentally evaluated the performance and quality of our methodology, and corresponding heuristic implementation in Section 6.3, and compare them against the exhaustive approach in Section **??**. In the following, Section 6.4 presents the simulator and testing infrastructure adopted in our experiments, as well as the complete experimental settings; Section 6.5 analyses the performance of our solution in terms of execution time; Section 6.6 presents the quality of our heuristic algorithm in terms of the metrics in Section 6.1.

6.4 Testing Infrastructure and Experimental Settings

Our testing infrastructure is a Swift-based simulator of a service-based ecosystem, including service execution, comparison, and composition. The simulator first defines the pipeline template as a sequence of vertexes in the range $3-7$. We recall that alternative vertexes are modeled in different pipeline templates, while parallel vertexes only add a fixed execution time that is negligible and do not affect the quality of our approach. Each node is associated with a (set of) policy with transformations varying in three classes:

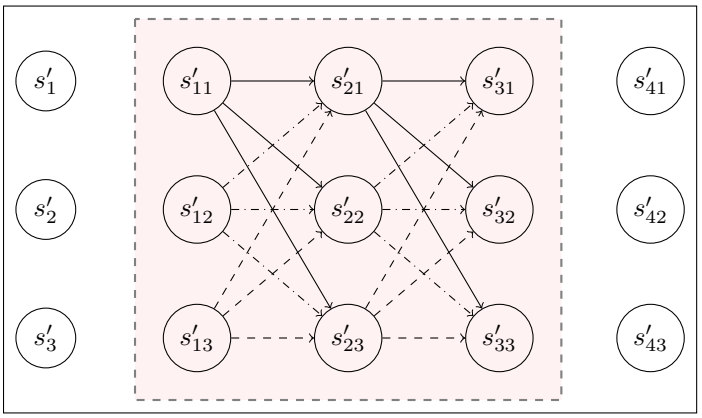


Fig. 4: Service composition instance

roman* *Confident*: Adjusts data removal to a percentage within $[0.8, 1]$. roman* *Diffident*: Sets data removal percentage to $[0.2, 0.5]$. roman* *Average*: Modifies data removal percentage within $[0.2, 1]$. set of functionally-equivalent candidate services is randomly generated.

Upon setting the sliding window size, the simulator selects a subset of vertexes along with their corresponding candidate services. It then generates all possible service combinations for the chosen vertexes. For each combination, the simulator calculates a metric, selecting the first service from the optimal combination before shifting the sliding window. When the end of the node list is reached, or when the window size equals the node count, the simulator computes the optimal service combination for the remaining vertexes.

An hash function is to simulate the natural interdependence between services. This is particularly important when the removal of data by one service may impact another. By assigning weights to the services using this function, the system aims to reflect the interconnected dynamics among the services.

The simulator is used to assess the performance and quality of our sliding window heuristic in Section 6 for the generation of the best pipeline instance (Section 5). Our experiments have been run on a workstation equipped with a 2.40GHz i5-8279U CPU with 16GB RAM and a 512GB SSD. Each experiment was repeated ten times and the results averaged to improve the reliability of the data.

6.5 Perfomance

We first calculated the execution time required by our exhaustive solution. We incrementally varied the number of vertexes and the number of services per node. The results of these evaluations are presented in **??**. As anticipated, the trend in execution times is exponential. **??** displays the execution time plots, clearly showing that as the number of vertexes increases, the execution time grows exponentially. Execution times for up to 5 vertexes and 6 services were computed directly, while the remaining data points were obtained through interpolation. Subsequently, the logical extension of this empirical inquiry involves evaluating the execution time efficiency attributable to the implementation of the sliding window heuristic.

We then evaluated our heuristics to quantify the execution time reduction achieved through the application of heuristics. In this context, the number of vertexes and services per node was incrementally increased, with the addition of a sliding window whose size was progressively enlarged in each experiment. The outcomes are depicted in **??**, and as expected, we observed a marked reduction in execution times with the implementation of the sliding window heuristic. This empirical evidence highlights the heuristic's ability to reduce computational demands, an aspect that becomes increasingly pivotal as the problem's complexity grows. The use of a logarithmic scale to illustrate the results linearizes the exponential growth associated with the exhaustive method, offering a clear visual confirmation of the heuristic's efficiency in decreasing computational time.

6.6 Quality

We finally evaluated the quality of our heuristic comparing, where possible, its results with the optimal solution retrieved by executing the exhaustive approach. The latter executes with window size equals to the number of vertexes and provides the best, among all possible, solution.

We recall that we considered three different setting, confident, diffident, average, varying the policy transformations, that is, the amount of data removal at each node. Setting confident assigns to each policy a transformation that changes the amount of data removal in the interval [x,y] (Jaccard coefficient) or decreases the probability distribution dissimilarity in the interval [x,y] (Jensen-Shannon Divergence). Setting diffident assigns to each policy a transformation that changes the amount of data removal in the interval [x,y] (Jaccard coefficient) or decreases the probability distribution dissimilarity in the interval [x,y] (Jensen-Shannon Divergence). Setting average assigns to each policy a transformation that changes the amount of data removal in the interval [x,y] (Jaccard coefficient) or decreases the probability distribution dissimilarity in the interval [x,y] (Jensen-Shannon Divergence). We finally evaluated the quality of our heuristic comparing, where possible, its results with the optimal solution retrieved by executing the exhaustive approach. The latter executes with window size equals to the number of services per node and provides the best, among all possible, solution.

The number of vertexes has been varied from 3 to 7, while the number of services per node has been set from 2 to 6. The experiments have been conducted with different service data pruning profiles.

Fig. 5 presents our results In the figure each chart represents a configuration with a specific number of vertexes, ranging from 3 to 7. On the x-axis of each chart, the number of services is plotted, which ranges from 2 to 6. The y-axis represents the metric value, which varies across the charts. Each chart shows different window sizes, labeled as W Size 1, W Size 2, and so on, corresponding to various metric values.

The initial chart in Fig. 5a focusing on a 3-node configuration, reveals that metric values commence at a high of 0.12 and decline to a low of 0.06 as the number of services increases. The chart illustrates that smaller window sizes correspond to higher metric values, with window size one exhibiting the highest values and window size three the lowest. Notably, the metric value for a window size of two is comparable to, or better than, that for a window size of three.

The second chart, detailing a 4-node configuration, displays a similar pattern where metric values start at 0.16 and fall to 0.07 with an increasing number of services. In this configuration, the gap between the metric values for a window size of one is more marked, whereas the disparity between the values for window sizes of two and three is less distinct.

In the third chart, which examines a 5-node configuration, metric values initiate at a peak of 0.17 and decrease to 0.09 as services rise. Here, metric values for window sizes one and two are closely matched, whereas those for window sizes three to five are lower, with the gap between the metric values for window sizes one and five reaching up to 0.02.

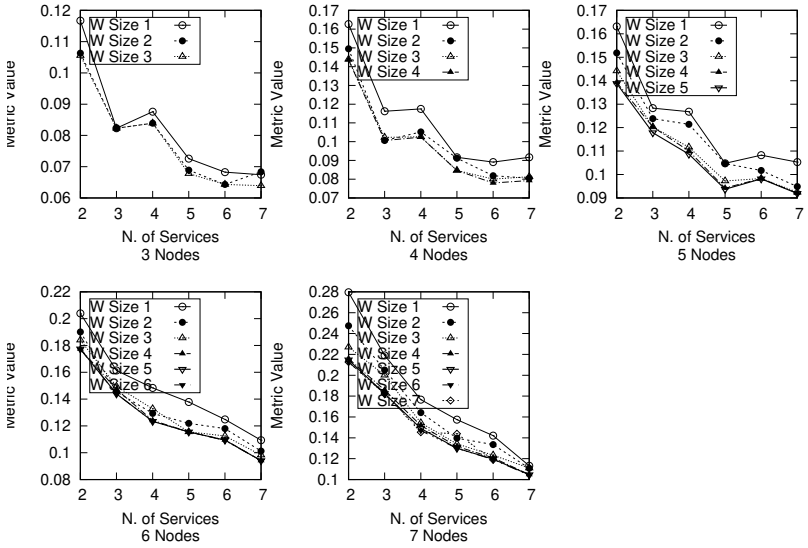
The fourth chart, exploring a 6-node configuration, shows metric values starting at 0.20 and diminishing to 0.09 with more services. The variance in metric values for a window size of one is again more noticeable, while values for window sizes three to six are lower, with some overlaps, as seen with window sizes four to six.

Lastly, the fifth chart, presenting a 7-node configuration, indicates that metric values start at 0.28 and reduce to 0.10 as the number of services escalates. The difference in metric values for a window size of one is pronounced, while values for window sizes three to six are lower, with overlapping occurrences similar to those in the 6-node setup. Metric values for window sizes two and three fluctuate between 0.25 and 0.12, while those for window sizes four and five oscillate between 0.22 and 0.1.

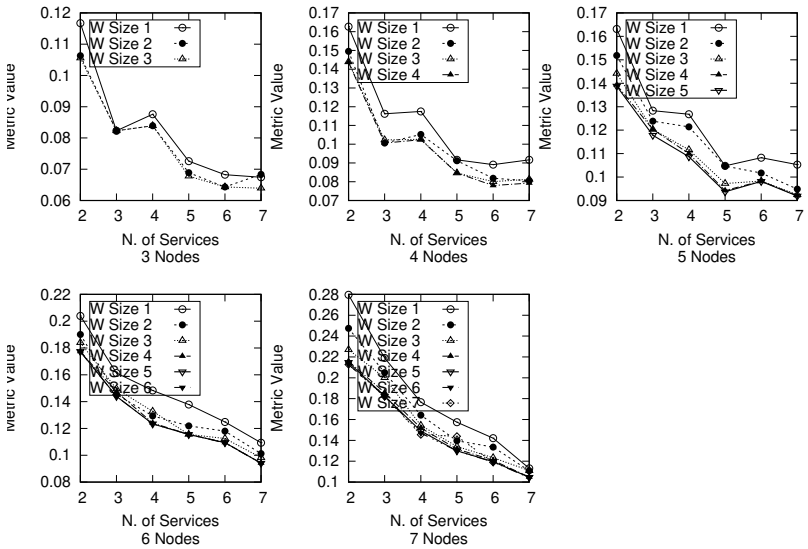
It's worth noting As the number of vertexes increases in each subsequent chart, the relationship between the window size and metric value is depicted, showing how metric values tend to decrease (better data preservation) as the window size increases across different node configurations. This suggests that the heuristic performs better when it has a broader perspective of the data it is analyzing. The trend is consistent across various numbers of vertexes, from three to seven, indicating that the heuristic's enhanced performance with larger window sizes is not confined to a specific setup but rather a general characteristic of its behavior. Finally, the data suggest that while larger window sizes generally lead to better performance, there might exist a point 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 resources or the complexity introduced by larger windows.

7 Related Work

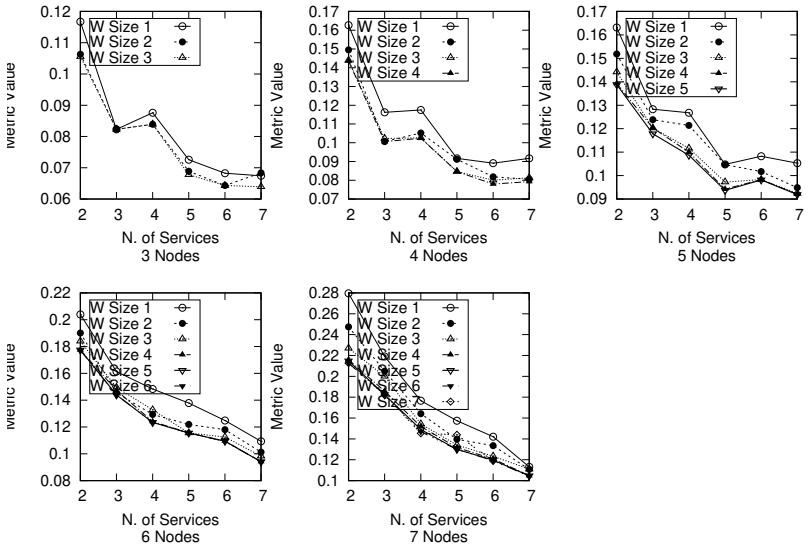
8 Conclusions



(a) Services operating on the 0.2 to 0.8 data pruning range.



(b) Services operating on the 0.2 to 0.5 data pruning range.



(c) Services operating on the 0.2 to 1 data pruning range.

Fig. 5: Three figures side by side

REFERENCES

[1] Ethics guidelines for trustworthy ai. URL https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

[2] European data governance act. URL https://digital-strategy.ec.europa.eu/en/policies/data-governance-act

[3] Gaiax. URL https://www.data-infrastructure.eu/GAIAX/Navigation/EN/Home/home.html

[4] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). URL https://eur-lex.europa.eu/eli/reg/2016/679/oj

[5] 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 (2020)

[6] Al-Badi, A., Tarhini, A., Khan, A.I.: Exploring Big Data Governance Frameworks. Procedia Computer Science **141**, 271–277 (2018)

[7] 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)

[8] 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)

[9] Blomer, J.: A survey on distributed file system technology. In: Journal of Physics: Conference Series, vol. 608, p. 012039. IOP Publishing (2015)

[10] Byun, J.W., Li, N.: Purpose based access control for privacy protection in relational database systems. The VLDB Journal **17**(4), 603–619 (2008)

[11] Castano, S., Ferrara, A., Montanelli, S.: Exploratory analysis of textual data streams. Future Generation Computer Systems **68**, 391–406 (2017)

[12] Chabin, J., Ciferri, C.D.A., Halfeld-Ferrari, M., Hara, C.S., Penteado, R.R.M.: Role-Based Access Control on Graph Databases. In: T. Bureš, R. Dondi, J. Gamper, G. Guerrini, T. Jurdziński, C. Pahl, F. Sikora, P.W. Wong (eds.) Proc. of SOFSEM, pp. 519–534. Springer International Publishing, Cham (2021)

[13] Colombo, P., Ferrari, E.: Access control technologies for big data management systems: literature review and future trends. Cybersecurity **2**(1), 3 (2019)

[14] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org

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

[16] Gupta, M., Patwa, F., Sandhu, R.: Object-Tagged RBAC Model for the Hadoop Ecosystem. In: G. Livraga, S. Zhu (eds.) Data and Applications Security and Privacy XXXI, pp. 63–81. Springer International Publishing, Cham (2017)

[17] 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)

[18] Hassan, M.U., Rehmani, M.H., Chen, J.: Differential privacy techniques for cyber physical systems: a survey. IEEE Communications Surveys & Tutorials **22**(1), 746–789 (2019)

[19] 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. Miami, USA (2014)

[20] Hu, V.C., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K.: Guide to attribute based access control (abac) definition and considerations. NIST special publication **800**(162), 1–54 (2014)

[21] Huang, L., Zhu, Y., Wang, X., Khurshid, F.: An Attribute-Based Fine-Grained Access Control Mechanism for HBase. In: S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A.M. Tjoa, I. Khalil (eds.) Database and Expert Systems Applications, pp. 44–59. Springer International Publishing (2019)

[22] Jain, N., Srivastava, V.: Data mining techniques: a survey paper. IJRET: International Journal of Research in Engineering and Technology **2**(11), 2319–1163 (2013)

[23] Kratzke, N.: A brief history of cloud application architectures. Applied Sciences **8**(8), 1368 (2018)

[24] Kulkarni, A.P., Khandewal, M.: Survey on hadoop and introduction to yarn. (2014)

[25] de 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). DOI 10.1109/TrustCom/BigDataSE.2018.00257

[26] Palanisamy, S., SuvithaVani, P.: A survey on rdbms and nosql databases mysql vs mongodb. In: 2020 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–7. IEEE (2020)

[27] Pedreira, V., Barros, D., Pinto, P.: A review of attacks, vulnerabilities, and defenses in industry 4.0 with new challenges on data sovereignty ahead. Sensors **21**(15), 5189 (2021)

[28] Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudonymity—a proposal for terminology. In: Designing privacy enhancing technologies, pp. 1–9. Springer (2001)

[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). DOI 10.1109/EuroSPW.2019.00010

[30] Qiu, J., Wu, Q., Ding, G., Xu, Y., Feng, S.: A survey of machine learning for big data processing. EURASIP Journal on Advances in Signal Processing **2016**(1), 1–16 (2016)

[31] 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), 8:1–8:24 (2017)

[32] Salloum, S., Dautov, R., Chen, X., Peng, P.X., Huang, J.Z.: Big data analytics on apache spark. International Journal of Data Science and Analytics **1**(3), 145–164 (2016)

[33] Samarati, P.: Protecting respondents identities in microdata release. IEEE transactions on Knowledge and Data Engineering **13**(6), 1010–1027 (2001)

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

[35] Sethi, R., Traverso, M., Sundstrom, D., Phillips, D., Xie, W., Sun, Y., Yegitbasi, N., Jin, H., Hwang, E., Shingte, N., et al.: Presto: Sql on everything. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 1802–1813. IEEE (2019)

[36] 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). DOI 10.1109/BigDataCongress.2018.00029

[37] Thambiraja, E., Ramesh, G., Umarani, D.R.: A survey on various most common encryption techniques. International journal of advanced research in computer science and software engineering **2**(7) (2012)

[38] Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment **2**(2), 1626–1629 (2009)

[39] Tolone, W., Ahn, G.J., Pai, T., Hong, S.P.: Access control in collaborative systems. ACM Computing Surveys (CSUR) **37**(1), 29–41 (2005)

[40] Tsai, C.W., Lai, C.F., Chao, H.C., Vasilakos, A.V.: Big data analytics: a survey. Journal of Big data **2**(1), 1–32 (2015)

[41] Vassiliadis, P.: A survey of extract–transform–load technology. International Journal of Data Warehousing and Mining (IJDWM) **5**(3), 1–27 (2009)

[42] Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S.: A survey on distributed machine learning. Acm computing surveys (csur) **53**(2), 1–33 (2020)

[43] Wallace, K.A.: Anonymity. Ethics and Information technology **1**(1), 21–31 (1999)

[44] Woodruff, A., Stonebraker, M.: Supporting fine-grained data lineage in a database visualization environment. In: Proceedings 13th International Conference on Data Engineering, pp. 91–102. IEEE (1997)

[45] Wu, N., Xie, Y.: A survey of machine learning for computer architecture and systems. ACM Computing Surveys (CSUR) **55**(3), 1–39 (2022)

[46] 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. Online (2020)