Polimeno *et al. Journal of Big Data* (2025) 12:62
https://doi.org/10.1186/s40537-025-01118-5

Journal of Big Data

RESEARCH Open Access

Maximizing data quality while ensuring data protection in service-based data pipelines

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

*Correspondence:
claudio.ardagna@unimi.it

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

Abstract

The growing capacity to handle vast amounts of data, combined with a shift in service delivery models, has improved scalability and efficiency in data analytics, particularly in multi-tenant environments. Data are treated as digital products and processed through orchestrated service-based data pipelines. However, advancements in data analytics do not find a counterpart in data governance techniques, leaving a gap in the effective management of data throughout the pipeline lifecycle. This gap highlights the need for innovative service-based data pipeline management solutions that prioritize balancing data quality and data protection. The framework proposed in this paper optimizes service selection and composition within service-based data pipelines to maximize data quality while ensuring compliance with data protection requirements, expressed as access control policies. Given the NP-hard nature of the problem, a sliding-window heuristic is defined and evaluated against the exhaustive approach and a baseline modeling the state of the art. Our results demonstrate a significant reduction in computational overhead, while maintaining high data quality.

Keywords: Access control, Big data, Data protection, Data quality, Privacy, Service-based data pipelines

Introduction

The wide success and adoption of cloud-edge infrastructures and their intrinsic multitenancy radically have changed how distributed systems are developed, deployed, and executed, redefining IT scalability, flexibility, and efficiency. Multitenancy in fact enables multiple users to share resources, such as computing power, storage, and services, optimizing their utilization and reducing operational costs.

The increasing ability to collect and manage huge volumes of data, coupled with a paradigm shift in service delivery models, has also significantly enhanced scalability and efficiency in data analytics. Data are treated as digital products, which are managed and analyzed by multiple services orchestrated in pipelines. This shift is fostering the emergence of new platforms and environments, such as data marketplaces and data spaces, where data in critical domains (e.g., law enforcement, healthcare, transportation) can be pooled and shared to maximize data quality and trustworthiness, and distributed

data management systems supporting data storing, versioning, and sharing for complex analytics processes.[1]

The flip side of a scenario where service-based data pipelines orchestrate services selected at run time and are delivered in the cloud-edge continuum is the increased complexity in data governance. Data are shared and analyzed by multiple services owned by different providers introducing unique security challenges. On one side, the pipeline owner and data providers have different security requirements, access policies, and data sensitivity that vary according to the specific orchestrated services; on the other side, orchestrated services (data consumers) have different profiles that impact on the amount of data they can access and analyze.

Adequate measures such as encryption, access control mechanisms, and data anonymization techniques have been implemented to protect data against unauthorized access and ensure compliance with regulatory requirements such as GDPR [1] or HIPAA [2]. However, data quality is also crucial and must be guaranteed, 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 [3, 4]. On the other hand, *data security and privacy* focused on the protection of confidential information and adherence to rigorous privacy regulations [5–8]. Although extensively studied, these investigations often prioritize enhancing the quality, security, and privacy of source data rather than ensuring data quality, security, and privacy throughout the entire processing pipeline, or the integrity of outcomes derived from data.

A valid solution requires a holistic approach that integrates technological solutions, organizational policies, and ongoing monitoring and adaptation to emerging threats and regulatory changes across the entire pipeline lifecycle. The implementation of robust access control mechanisms or privacy techniques, ensuring that only authorized users can access specific datasets (or a portion thereof) is just a mandatory but initial step. Additional requirements are emerging. First, data protection requirements should be defined at each stage of the pipeline, potentially integrating techniques like data masking and anonymization (e.g., k-anonymity, l-diversity, differential privacy) to safeguard sensitive information, thereby preserving data privacy while enabling high-quality 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. Third, data protection and data quality requirements should drive the process that builds a pipeline with maximum data quality while addressing data protection requirements.

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

[1] https://joinup.ec.europa.eu/collection/elise-europeanlocation-interoperability-solutions-e-government/glossary/term/data-marketplace, https://internationaldataspaces.org/, https://digitalstrategy.ec.europa.eu/en/library/staff-working-documentdata-spaces.

1. How does a data protection solution affect data quality in the pipeline? How can we minimize this impact thus maximizing the overall data quality?
2. Should data protection be implemented at each pipeline step rather than filtering all data at the outset?
3. In a scenario where service-based data pipelines are built by selecting the best services among various candidate services, how might these choices be driven by quality requirements?

Based on the aforementioned considerations, we propose a data governance framework for service-based data pipelines. The primary objective of our framework is to support the selection of data processing services within the pipeline, with a central focus on the selection of those services that maximize data quality, while upholding security and privacy requirements.[2] 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 in terms of data completeness by retaining the maximum amount of information when applying data protection transformations.

The primary contributions of the paper can be summarized as follows: 1. we define a data governance framework that implements an algorithm for the selection of data processing services enriched with metadata that describe both data protection and functional requirements; 2. we propose a parametric heuristic tailored to address the computational complexity of the NP-hard service selection problem that maximizes the quality of data, while addressing data protection and functional requirements; 3. we evaluate the performance and quality of the algorithm through experiments conducted using a real, open dataset from the domain of law enforcement. Performance and quality are compared against a baseline modeling current approaches in literature.

The remainder of the paper is structured as follows: Sect. System model and reference scenario presents our system model and reference scenario. Section Pipeline template introduces the pipeline template and describes data protection and functional annotations. Section Pipeline instance describes the process of building a pipeline instance from a pipeline template according to our service selection algorithm. Section Maximizing the pipeline instance quality introduces the quality metrics used in service selection and the heuristic solving the service selection problem. Section Experiments presents our experimental results. Section Related work discusses the state of the art and Sect. Conclusions and future work draws our concluding remarks and future work.

[2] We note that the assembly of the selected services in an executable pipeline is out of the scope of this paper. However, our approach is agnostic to the specific executable environment.

System model and reference scenario

We present our system model (Sect. System model) and reference scenario (Sect. Reference scenario).

System model

We consider a service-based environment where a service-based data pipeline (service pipeline in the following) is designed to analyze data. Our service pipeline is enriched with metadata specifying data protection requirements and functional specifications, and models the data flow among component services, without posing any restrictions on the control flow. It is composed of the following parties:

- *Service*, 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 a service 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 service 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 Service 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.

Modeling the pipeline as a directed acyclic graph ensures a sound representation of its data flow. This standard approach to representing workflows, including service pipelines, closely mirrors real-world systems.

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. To simplify the explanation and maintain clarity, we model alternative execution paths as distinct service pipelines, rather than embedding alternative structures within a single pipeline. This representation is equivalent to having alternative structures within a single pipeline, as each distinct pipeline corresponds to one possible execution path. By separating these paths into individual pipelines, we avoid the additional complexity of modeling alternatives within the same structure, while fully capturing all execution possibilities. 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, while applicable to a generic scenario, operates under the assumption that *preserving a greater quantity of data is correlated with enhanced data quality*, a principle that reflects many real-world scenarios [9, 10]. 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 to be compiled in an executable pipeline.

Reference scenario

Our approach targets application domains involving sensitive data, such as Personally Identifiable Information (PII), that must be securely shared and protected across diverse and complex analytical processes involving multiple stakeholders. It is applicable across industrial use cases based on cloud-edge infrastructures, where data from third-party (IoT) devices are injected and shared via the cloud, as well as in data ecosystems across sectors such as healthcare, finance, law enforcement, and justice.

Our reference scenario draws on commonly used dataspaces, such as dataspace on public administration, focusing specifically on the law enforcement domain. Using open data, we selected a scenario that includes real sensitive records of individuals detained in Connecticut Department of Correction facilities while awaiting trial.[3] Various stakeholders may use these data for different objectives: public health agencies to monitor inmate health trends, judicial bodies to track case processing efficiency, advocacy groups to identify disparities in detention, policymakers to analyze the impacts on the criminal justice system, social services to prepare post-release support, researchers to study the broader social effects of pre-trial detention, and correctional departments to compare admission trends across facilities.

To streamline the use case, we focused on a subset of this real-world scenario, envisioning three Department of Correction (DOC) partners—Connecticut, New York, and New Hampshire—sharing data according to their privacy policies. In this scenario, a user from the Connecticut DOC seeks to compare admission trends in Connecticut's facilities with those in New York and New Hampshire to evaluate, for instance, possible

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

discrimination and unfair treatment of individuals awaiting trial. Additionally, the policy requires that all service execution remains within the Connecticut DOC environment, mandating data protection measures if data transmission extends beyond Connecticut's borders.

Our reference scenario aligns with the latest regulations on data governance (e.g., the European AI Data Governance Act[4]) and artificial intelligence (e.g., the EU AI Act[5]). In particular, the EU AI Act identifies law enforcement and administration of justice as high-risk domains where proper data governance, risk management, and quality management systems must be employed in AI training and operation following the requirements on data quality and protection set in this paper.

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.

The user's objective aligns with the predefined service pipeline in Fig. 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 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.

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 (Sect. Pipeline template definition). We then present both functional annotations (Sect. Functional annotations) and data protection annotations (Sect. Data protection annotation), providing an example of a pipeline template in the context of the reference scenario in Sect. Reference scenario.

Pipeline template definition

Given the service pipeline in Definition 1, we use annotations to express data protection requirements and functional requirements on the services to be integrated into 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 to be enforced on data, in the form of policies $p \in P$, with

[4] https://digital-strategy.ec.europa.eu/en/policies/data-governance-act).

[5] https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai.

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...	...

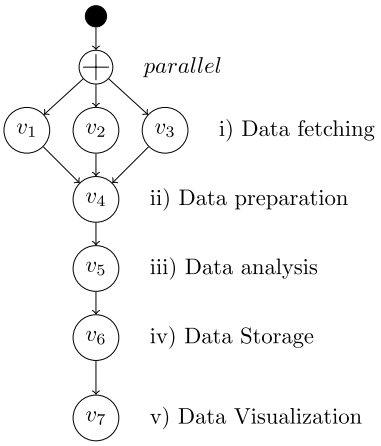


Fig. 1 Service pipeline in the reference scenario

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, λ, γ) is a direct acyclic graph extended with two annotation functions: 1. *Data Protection Annotation* λ that assigns a label λ (v_i) to each vertex $v_i \in V_S$. Label λ (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 γ (v_i) to each vertex $v_i \in V_S$. Label γ (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 λ (v_i) are combined using logical OR, meaning that the access decision is positive if at least one policy p_j evaluates to *true*.

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 3 (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.

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 an 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. 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 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 with 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 Pipeline instance describes the selection process and the pipeline instance generation.

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 γ (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, and expected outputs. It also 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.

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 Sect. Reference scenario. Figure 2c presents an example of a pipeline template consisting of five stages, each one annotated with a policy in Table 2a and corresponding data transformations in Table 2b.

The first stage in Fig. 2c 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 in Fig. 2a with an empty transformation t_0^p in Fig. 2b. Functional requirements F_1, F_2, F_3 prescribe a URI as input and the corresponding dataset as output.

The second stage in Fig. 2c 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 in Fig. 2a, which apply different data transformations depending on the relation between the dataset and the service owner.

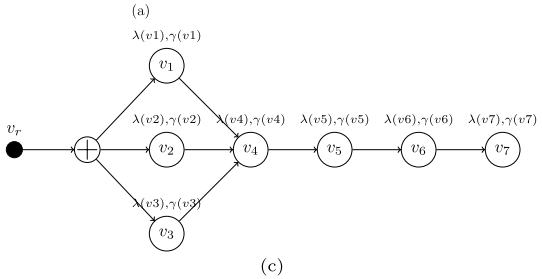
If the service owner is also the dataset owner (i.e., $(service\_owner = dataset\_owner)$), the dataset is not anonymized (t_0^p). If the service owner is a partner of the dataset owner (i.e., $(service\_owner = partner(dataset\_owner))$), the dataset is anonymized at *level1* (t_1^p). If the service 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.

Vertex	Annotation λ	p = $\langle$subject, *object*, *action*, *environment*, *transformation*$\rangle$
v_1,v_2,v_3	$\lambda(v_1)$, $\lambda(v_2)$, $\lambda(v_3)$	$p_0 = \langle \text{ANY}, dataset, READ, ANY, t_0^p \rangle$
v_4,v_5	$\lambda(v_4)$, $\lambda(v_5)$	$p_1 = \langle \{(service\_owner = dataset\_owner)\}, dataset, READ, ANY, t_0^p \rangle$
v_4,v_5	$\lambda(v_4)$, $\lambda(v_5)$	$p_2 = \langle \{(service\_owner = partner(dataset\_owner))\}, dataset, READ, ANY, t_1^p \rangle$
v_6	$\lambda(v_6)$	$p_3 = \langle \{(service\_region = dataset\_origin)\}, dataset, WRITE, ANY, t_0^p \rangle$
v_6	$\lambda(v_6)$	$p_4 = \langle \{(service\_region = \{\text{“}NY\text{”}, \text{“}NH\text{”}\})\}, dataset, WRITE, ANY, t_1^p \rangle$
v_7	$\lambda(v_7)$	$p_5 = \langle \text{ANY}, dataset, READ, \langle \text{environment=“risky”} \rangle, t_3^p \rangle$
v_7	$\lambda(v_7)$	$p_6 = \langle \text{ANY}, dataset, READ, \langle \text{environment=“not\_risky”} \rangle, t_4^p \rangle$

(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)



(c)

Fig. 2 Anonymization policies (**a**) and data transformations (**b**) Pipeline Template Example (**c**)

The third stage in Fig. 2c consists of vertex v_5 for data analysis. Data protection annotation λ (v_5) refers to policies p_1 and p_2 in Fig. 2a, 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 in Fig. 2c consists of vertex v_6, managing data storage. Data protection annotation λ (v_6) refers to policies p_3 and p_4 in Fig. 2a, 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 in Fig. 2c consists of vertex v_7, responsible for data visualization. Data protection annotation λ (v_7) refers to policies p_5 and p_6 in Fig. 2a, 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 a data visualization interface (possibly in the form of a JSON file) as output.

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 sets of

INPUT
$G^{\lambda,\gamma}$: Pipeline Template
S^c: Candidate Services

OUTPUT
G': Pipeline Instance

Instantiate_Pipeline($G^{\lambda,\gamma}$, S^c)

```
1  /* Initialize the pipeline instance*/
2  G' = {};
3  /* Traverse the pipeline template using BFS*/
4  for each v in G^{λ,γ}
5      v' = Generate_Vertex(v);
6      G' = G' ∪ v';
7      S' = Filter_Services(S^c[v], v.policies);
8      selectedService = Select_A_Service(S');
9      v'.service = selectedService;
10 endfor;
11 return G';

12 Filter_Services(S^c[v], policies)
13 /* Filter candidate services based on policies*/
14 S' = {};
15 for each service s in S^c[v]:
16     if s.profile satisfies any policy:
17         S' = S' ∪ s;
18     endif;
18 endfor;
19 return S';
```

Fig. 3 Pseudocode of the pipeline instantiation process

services S_i^c, one for each vertex $v_i \in V_S$. Recall from Sect. Functional annotations 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 pseudocode of the pipeline instantiation process is presented in Fig. 3. The Pipeline Instance is generated by traversing the Pipeline Template with a breadth-first search algorithm (line 4–10), starting from the root vertex v_r. Then, for each vertex v_f in the pipeline template, the corresponding vertex v'_f is generated (line 5). Finally, for each vertex $v_i \in V_S$, a two-step approach is executed 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 (line 16). If yes, service s_j is compatible, otherwise it is discarded (line 17). The filtering algorithm finally returns a subset $S'_i \subseteq S_i^c$ of compatible services for each vertex $v_i \in V_S$ (line 19).
2. *Selection Algorithm*—The selection algorithm selects one service s'_i for each set S'_i of compatible services, which instantiates the corresponding vertex $v'_i \in V'$ (line 8–9).

There are many ways of choosing s'_i, Sect. Maximizing the pipeline instance quality presents our approach based on the maximization of data *quality* Q.

When all vertices $v_i \in V$ in $G^{\lambda,\gamma}$ have been visited, the Pipeline Instance G' is generated (line 11), with a service instance s'_i for each $v'_i \in V'$. Vertex v'_i is 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. When 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.

As presented in Table 2a, 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 s_{52}'s one 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 2a (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.

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 (Sect. NP-hardness of the max-quality pipeline instantiation problem). Finally, we introduce a parametric heuristic (Sect. Heuristic) designed to tackle the computational complexity associated with enumerating all possible combinations within a given set. The main objective of

Table 2 Instance example

(a) Valid 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"	✓	✗

(b) Best Quality 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

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.

Quality metrics

Ensuring data quality is mandatory to implement service-based data pipelines that provide accurate results and decision-making along the whole pipeline execution. Different definition of quality exists (e.g., [3, 4]) according to different dimensions such as completeness, timeliness, and accuracy, to name but a few. Quality metrics measure the data quality preserved at each step of the pipeline according to the selected quality dimensions, 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.

It is important to note that providing a comprehensive taxonomy of all possible dimensions and metrics is beyond the scope of this paper. In our future work, we will examine the conceptual and practical aspects of classifying and defining relevant quality metrics, such as timeliness and consistency, as well as their impact on our methodology.

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 [11]:

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

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.

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 [12] and is applicable to a pair of statistical distributions only. It is defined as follows:

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

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 Sect. Maximizing the pipeline instance quality, 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.

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 $1 \leq j < |S_i^c|$.

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.

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) [13]. 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 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 Sect. Pipeline instance 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 2b, 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 to 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 $Q_{31}\, Q_{32}$.

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 Sect. Pipeline instance 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 is 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. The 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. The function **SlidingWindowHeuristic** retrieves the optimal service combination composing G', considering the candidate

services associated with each vertex in $G^{\lambda,\gamma}$ (V, E, λ, γ) 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 the function **SelectService** (defined in line 10).

The 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 13–16). If the current combination has quality metric $\mathrm{M}(G'_w)$ higher than the best quality metric $\mathrm{M}(G^*_w)$, the current combination G'_w updates the best combination G^*_w (lines 14–15).

The 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, the 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).

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** $\mathrm{M}(G'_w) < \mathrm{M}(G^*_w)$
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

Experiments

We experimentally evaluated the performance and quality of our methodology (heuristic algorithm in Sect. Heuristic), and compared it against the exhaustive approach in Sect. NP-hardness of the max-quality pipeline instantiation problem and our baseline modeling solutions in the state of the art in Sect. Testing infrastructure and experimental settings. In the following, Sect. Testing infrastructure and experimental settings presents the simulator and experimental settings used in our experiments; Sect. Performance analyses the performance of our solution in terms of execution time; Sect. Quality discusses the quality of the best pipeline instance generated by our solution according to the metrics M_J and M_{JSD} in Sect. 5.1.

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 l 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 does not affect the performance and quality of our solution. Each vertex is associated with a (set of) policy that applies a filtering transformation that removes a given percentage of data.

The simulator then starts the instantiation process. At each step i, it selects the subset $\{v_i,\cdot\cdot,v_{|w|+i-1}\}$ 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}; $M_{J_{22}}$ =0.775 at s_{22}, showing how dependencies can affect the pipeline quality and, in turn, the instantiation process.

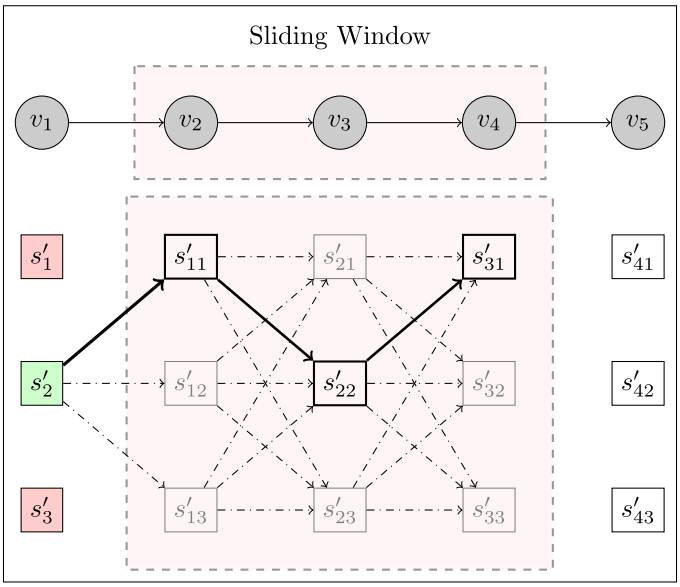


Fig. 5 Execution example of the sliding window heuristic using $l = 5, |S^c| = 3, |w| = 3$ at $i = 1$ step

Our simulator also supports the comparison of the performance and quality of our sliding-window heuristic with *i)* a baseline modeling solutions in the state of the art and *ii)* the exhaustive approach (i.e., the theoretical optimum). We modeled our baseline as a greedy approach that, for each node of the service pipeline, selects the best service that maximizes the data quality, while addressing data protection requirements in annotation λ. The reason is that, to the best of our knowledge, existing (industry) solutions and standards do not support service-based data pipelines and are therefore unable to instantiate the service pipeline according to the pipeline structure and service dependencies. We therefore defined our baseline as the sliding window heuristic configured with window size |w|=1. We implemented the exhaustive approach calculating the theoretical optimum as the sliding window heuristic configured with window size |w|=*l*, to illustrate the potential efficiency of our heuristics within realistic computational limits.

Table 3 outlines the parameters and corresponding values used in our experimental evaluation. Parameter *Window Size* |w|, varying from 1 to 7, models different configurations of our heuristic including our baseline and the exhaustive approach. Parameter *Pipeline Template Length l*, varying from 3 to 7, models the depth of the pipeline template as the number of vertices composed in a sequence. Parameter *Number of Candidate Services*, varying from 2 to 7, models the number of alternative services at each vertex of the pipeline template. Parameter *Filtering Configuration* considers two representative filtering transformations: *wide* removing a percentage of data in [0.2,1] and *average* in [0.5,0.8]. Parameter *Metric* considers two quality metrics: quantitative (M_J) and qualitative (M_{JSD}).

Our experiments have been run on a virtual machine equipped with an 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.

Performance

We first measured the performance (execution time) of our exhaustive, baseline, and heuristic solutions by varying the pipeline template length l in [3, 7] and the number of candidate services $|S^c|$ in [2, 7]. Figure 6 presents our results. The exhaustive approach can provide the optimal solution for all settings, but its execution time grows exponentially with the pipeline length and number of candidate 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 (l=7, $|S^c|$=7, $|w|$=6) is $\approx 3.8 \times 10^7 ms$, one order of magnitude lower than the corresponding exhaustive performance (l=7, $|S^c|$=7, $|w|$=7) $\approx 1.35 \times 10^8 ms$. As expected, the baseline (i.e., our heuristic with $|w|$=1) shows the best performance in all settings.

Quality

We evaluated the quality of our heuristic algorithm with different $|w|$ comparing its results with the baseline and, where possible, 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.

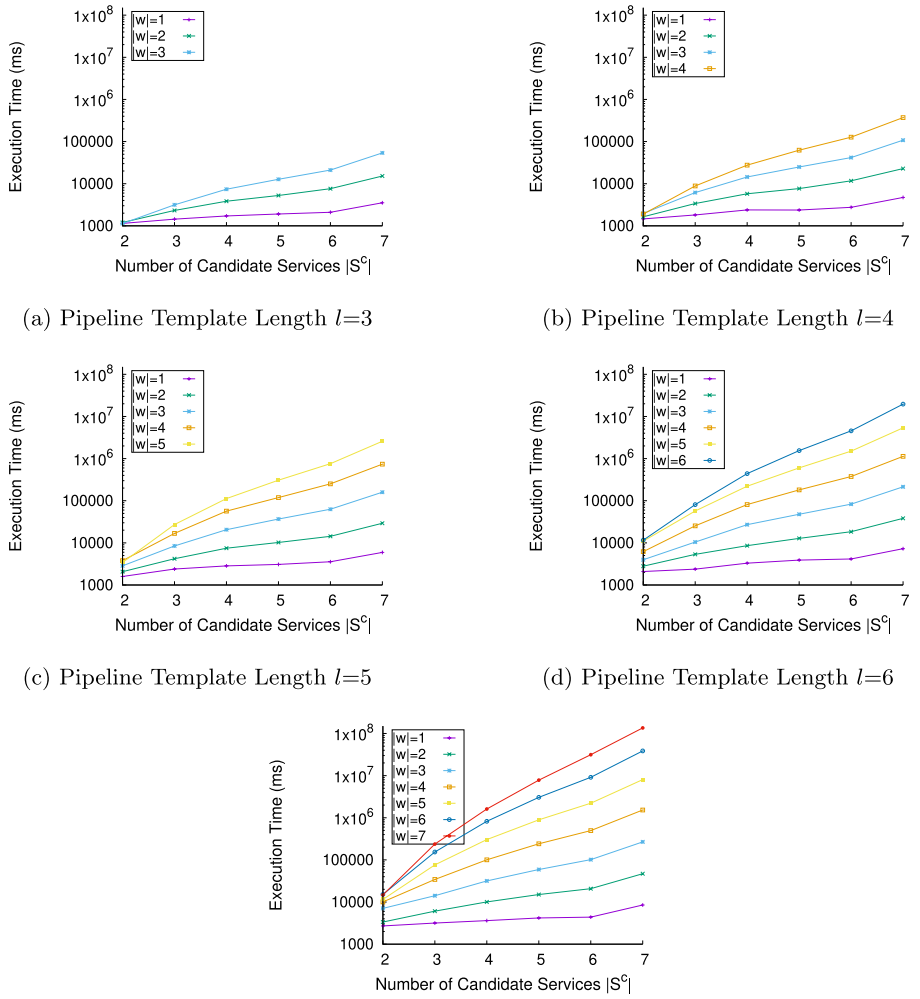
We run our experiments varying: *i)* pipeline template length l in [3, 7], *ii)* the window size $|w|$ in [1,l], and *iii)* the number of candidate services $|S^c|$ in [2, 7]. Each vertex is associated with a (set of) policy that applies filtering configuration *wide* (removing a percentage of data in [0.2, 1]) or *average* (removing a percentage of data in [0.5, 0.8]).

Figures 7 and 8 present our quality results using metric M_J in Sect. Quality metrics for configurations *wide* and *average*, respectively. 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 configuration *wide* (Fig. 7), the baseline ($|w|$=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 pipeline template length l=3, 0.810 and 0.942 for l=4, 0.580 and 0.853 for l=5, 0.682 and 0.943 for l=6, 0.596 and 0.821 for l=7. 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

Table 3 Experimental parameters

Parameter	Values
Window size |w|	1, 2, 3, 4, 5, 6, 7
Pipeline template length l	3, 4, 5, 6, 7
Number of candidate services $\|S^c\|$	2, 3, 4, 5, 6, 7
Filtering configuration	Wide, average
Metric	Quantitative (M_J), qualitative (M_{JSD})



(a) Pipeline Template Length l=3

(b) Pipeline Template Length l=4

(c) Pipeline Template Length l=5

(d) Pipeline Template Length l=6

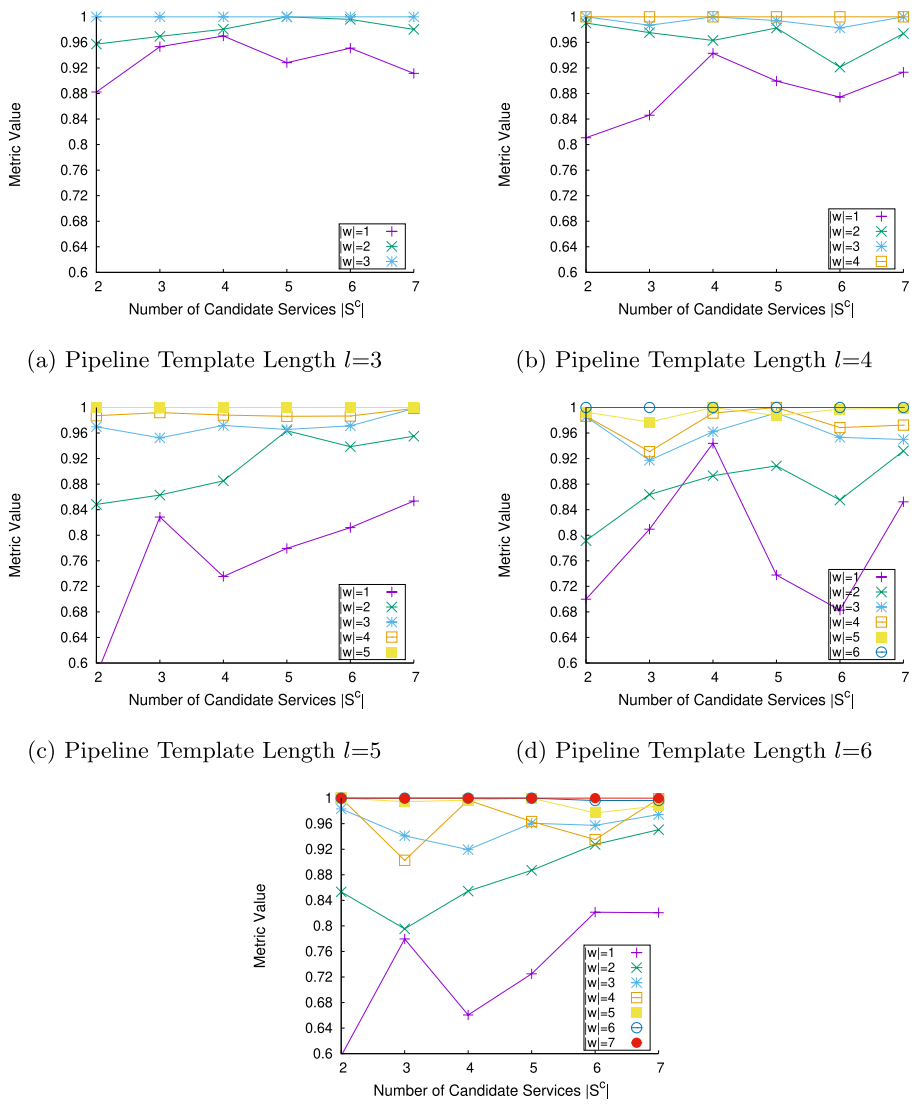
(e) Pipeline Template Length l=7

Fig. 6 Evaluation of performance using the *Qualitative* Metric in Configuration average. Each figure shows the execution time (in milliseconds) as a function of the number of candidate services $|S^C|$, for different values of the window size |w|, varying pipeline template length in: (**a**) l=3, (**b**) l=4, (**c**) l=5, (**d**) l=6, (**e**) l=7.

for l=3, 0.982 and 1.0 for l=4, 0.986 and 0.998 for l=5, 0.977 and 1.0 for l=6, 0.996 and 1.0 for l=7.

When considering configuration *average* (Fig. 8), the heuristic algorithm still provides good results, limiting the quality oscillations observed for configuration *wide* and approaching the quality of the exhaustive also for lower window sizes. The baseline ($|w|$ =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 l=3, 0.903 and 0.962 for l=4, 0.840 and 0.915 for l=5, 0.815 and 0.934 for l=6, 0.721 and 0.935 for l=7. When $|w|$=l-1, the quality varies between 0.980 and 1.0 for l=3, 0.978 and 1.0 for l=4, 0.954 and 1 for l=5, 0.987 and 1.0 for l=6, 0.990 and 1.0 for l=7.

Figures 9 and 10 present our quality results using metric M_{JSD} in Sect. Quality metrics for configurations *wide* and *average*, respectively.



(a) Pipeline Template Length l=3

(b) Pipeline Template Length l=4

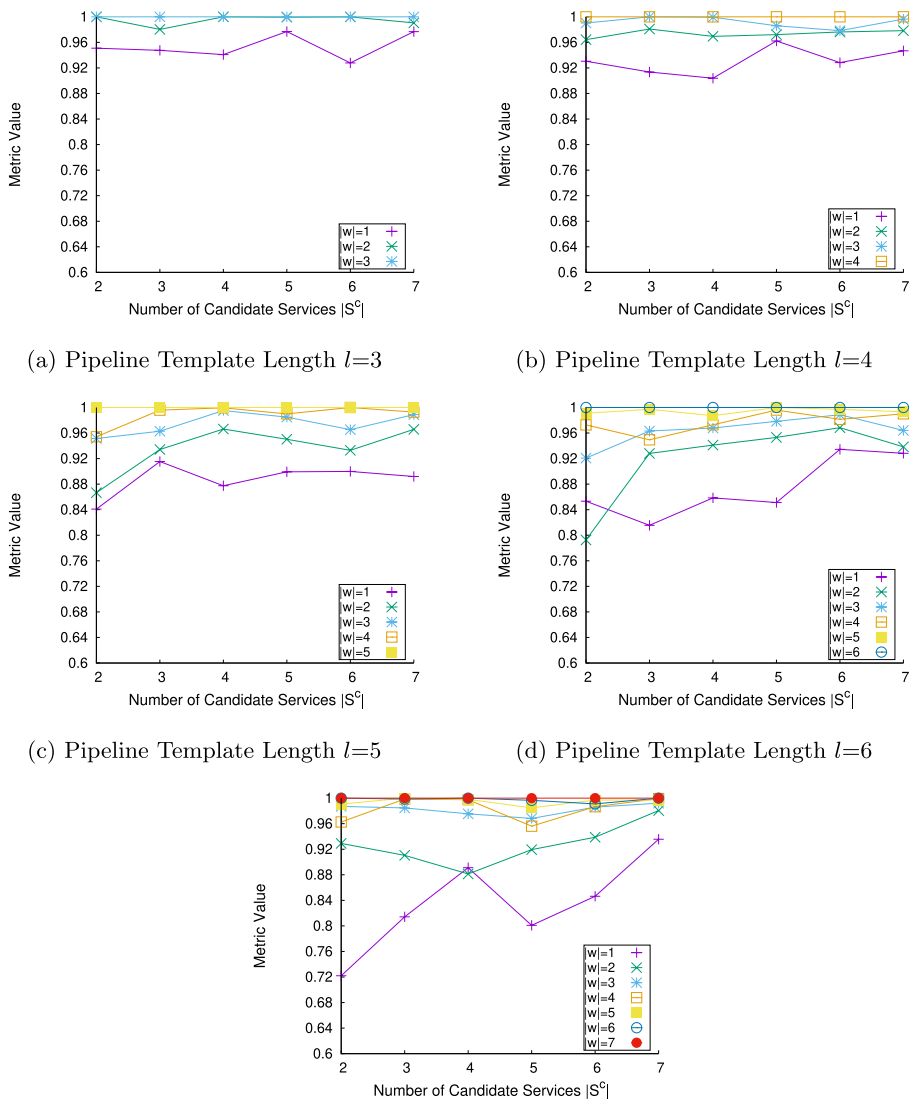
(c) Pipeline Template Length l=5

(d) Pipeline Template Length l=6

(e) Pipeline Template Length l=7

Fig. 7 Evaluation of quality using the *Quantitative* Metric in configuration wide. Each figure shows the metric value as a function of the number of candidate services $|S^c|$, for different values of the window size |w|, varying pipeline template length in: (**a**) l=3, (**b**) l=4, (**c**) l=5, (**d**) l=6, (**e**) l=7

When considering configuration *wide* (Fig. 9), the baseline ($|w|$=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 l=3, 0.941 and 0.988 for l=4, 0.919 and 0.974 for l=5, 0.911 and 0.971 for l=6, 0.877 and 0.924 for l=7. The worst quality results are obtained with the baseline, 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 l=4, 0.981 and 0.998 for l=5, 0.988 and 1.0 for l=6, 0.976 and 0.999 for l=7. When $|w|$=l-1, the quality varies between 0.987 and 0.998 for l=3, 0.993 and 1.0 for l=4, 0.985 and 0.999 for l=5, 0.997 and 1.0 for l=6, 0.995 and 1.0 for l=7.



(a) Pipeline Template Length l=3

(b) Pipeline Template Length l=4

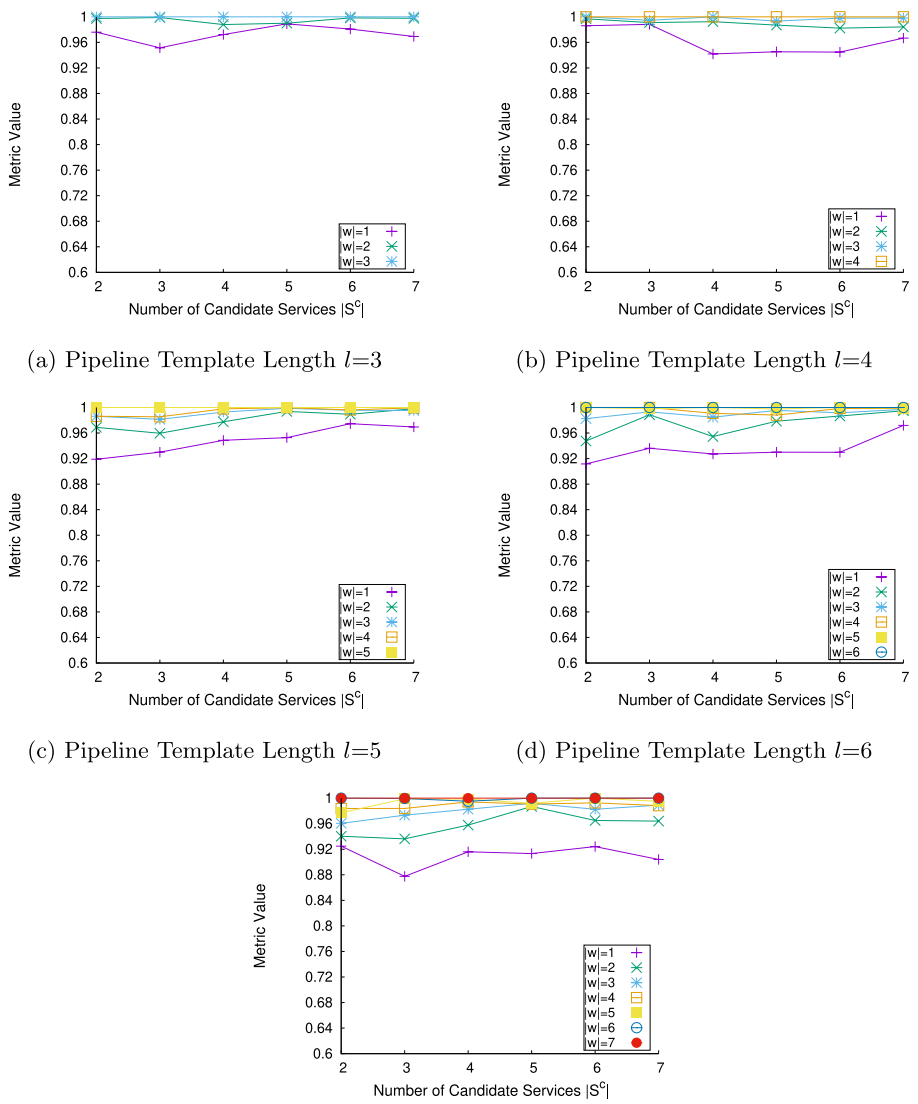
(c) Pipeline Template Length l=5

(d) Pipeline Template Length l=6

(e) Pipeline Template Length l=7

Fig. 8 Evaluation of quality using the *Quantitative* metric in configuration average. Each figure shows the metric value as a function of the number of candidate services $|S^c|$, for different values of the window size |w|, varying pipeline template length in: (**a**) l=3, (**b**) l=4, (**c**) l=5, (**d**) l=6, (**e**) l=7

When considering configuration *average* (Fig. 10), the baseline ($|w|$=1) provides results similar to configuration *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 l=3, 0.942 and 0.988 for l=4, 0.919 and 0.975 for l=5, 0.912 and 0.972 for l=6, 0.878 and 0.925 for l=7. The *average* configuration provides even tighter quality oscillations than the *wide* configuration. Notably, the poorest quality outcomes are observed with the baseline. 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



(a) Pipeline Template Length l=3

(b) Pipeline Template Length l=4

(c) Pipeline Template Length l=5

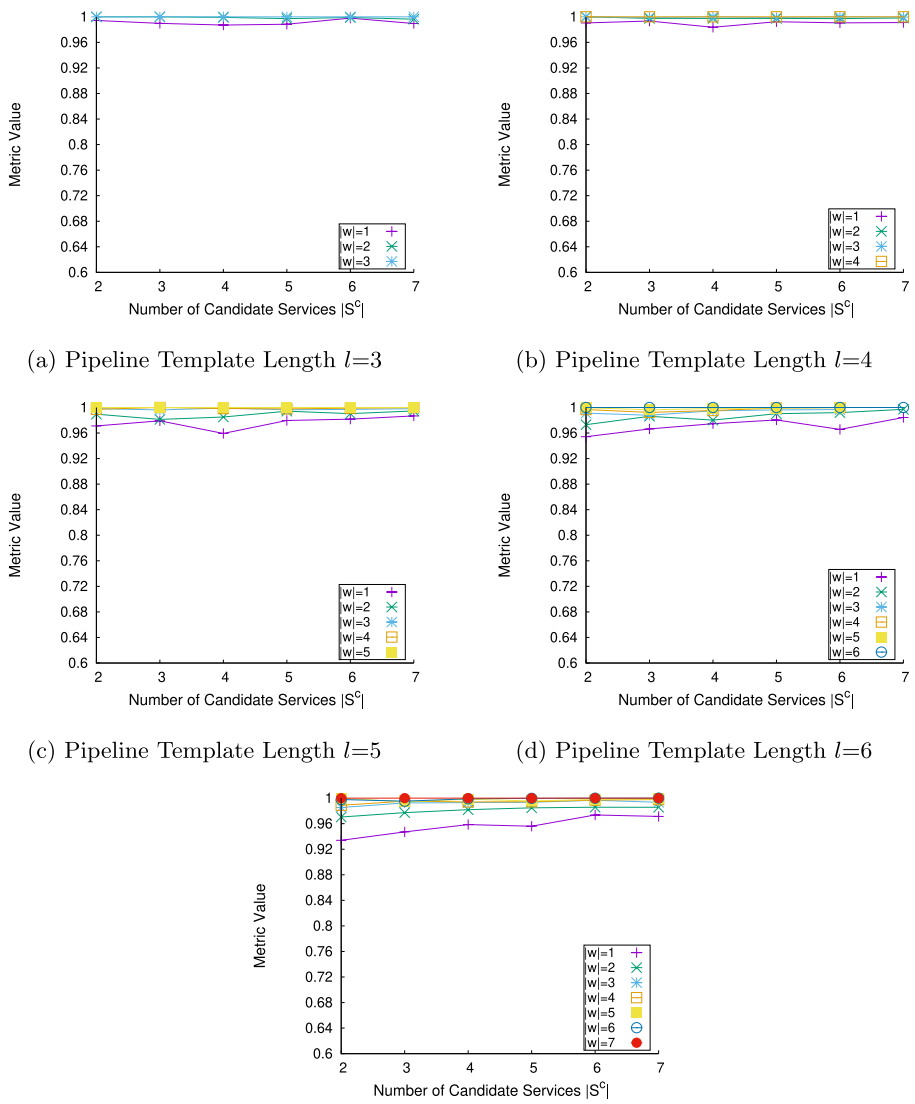
(d) Pipeline Template Length l=6

(e) Pipeline Template Length l=7

Fig. 9 Evaluation of quality using the *Qualitative* metric in configuration wide. Each figure shows the metric value as a function of the number of candidate services $|S^c|$, for different values of the window size |w|, varying pipeline template length in: (**a**) l=3, (**b**) l=4, (**c**) l=5, (**d**) l=6, (**e**) l=7

quality varies between 0.993 and 1 for l=4, 0.981 and 0.998 for l=5, 0.982 and 997 for l=6, 0.960 and 0.991 for l=7.

To conclude, our results (on average) show that the impact of the number of services on the retrieved quality is positive in all settings as the number of services increase, though negligible with respect to the impact provided by the pipeline template length and window size.



(a) Pipeline Template Length l=3

(b) Pipeline Template Length l=4

(c) Pipeline Template Length l=5

(d) Pipeline Template Length l=6

(e) Pipeline Template Length l=7

Fig. 10 Evaluation of quality using the *Qualitative* metric in configuration average. Each figure shows the metric value as a function of the number of candidate services $|S^c|$, for different values of the window size |w|, varying pipeline template length in: (**a**) l=3, (**b**) l=4, (**c**) l=5, (**d**) l=6, (**e**) l=7

Discussion

The experimental results we obtained yield several valuable insights that merit further discussion. Three key observations emerged as follows.

Trade-off between execution time and quality As expected, the execution time improvement provided by our heuristic introduces a loss of quality with respect to the exhaustive approach. This loss causes an increase in the quality variance, especially when the window size ($|w|$) is small compared to the vertex count. A fine-grained tuning of heuristics is needed to balance computational efficiency and data quality.

Impact of parameters on quality Our experiments show that the parameters listed in Table 3 significantly impact the quality of the pipeline, with the pipeline template length and the window size standing out as critical factors influencing both quality and performance.

Specifically, larger window sizes generally improve quality; however, there exists a point where the trade-off between computational cost and quality gain becomes suboptimal. Beyond this threshold, additional computational resources do not proportionately enhance data quality, as modeled by our metrics. We also note that lower window sizes exhibit higher instability, particularly under the *wide* configuration, where data quality varies significantly across different setups. This variation diminishes when larger window sizes, approximately half the length of the pipeline (e.g., $|w| = l/2$), are used, leading to more stable and consistent results.

We also note that the number of candidate services and service nodes increases the computation cost (performance) with a minor impact on quality (on average).

In conclusion, our results demonstrate a significant reduction in the computational cost, while maintaining high data quality. Further analysis is needed to explore the impact of additional parameters, particularly in terms of datasets modeling additional real-world domains, to understand their broader effects on quality. Investigating alternative quality metrics could also provide new insights and opportunities for improvement. Future experiments, as outlined in Section 9, will aim to address these aspects to provide a step further in the evaluation of the soundness and applicability of our framework on a larger scale.

Sliding window approach versus global awareness The intrinsic nature of our sliding window heuristic can sometimes lead to a local optimum, as the window size limits the candidate services for each pipeline stage to a restricted subset, which may prevent reaching the global optimum. This aspect is maximized when using the baseline representing the state of the art, where the sliding window heuristic is configured with a window size of $|w|$=1. Additionally, as dependencies between services increase, the likelihood of finding a sub-optimal solution rises. Our experiments show that *i)* increasing the window size helps mitigate this issue and *ii)* a broader decision-making scope becomes essential as service dependencies grow more complex.

Related work

We present an overview of the related work and a comparison with existing relevant tools and solutions in literature.

Data quality and data protection

Data quality is a widely studied research topic studied across various communities and perspectives. In the context of (big) data pipelines, 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 are 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 need of metrics to assess the quality of data resulting from anonymization processes. Differential privacy [14], *k*-anonymity [15], and *l*-diversity [16] are three distinct techniques used to provide data anonymization, with different protection levels and results on data quality. For example, differential privacy is highly effective in maintaining confidentiality, but the noise added can reduce data precision, impacting analytical accuracy, whereas *k*-anonymity and *l*-diversity generally maintain higher data quality than differential privacy, but they might still be unable to protect against sophisticated attacks. 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 [17–19]. 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 [20, 21]. 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 (i.e., data completeness)—one quantitative and one qualitative. Nonetheless, our framework and heuristic are designed to be modular and flexible, accommodating the chosen metric.

While existing techniques have provided sound and effective solutions that guarantee data quality and data protection, they often unsuit to scenarios aiming to maximize data quality while ensuring data protection, have limited expressiveness (e.g., the definition of *k* when *k*-anonymity is used to protect data), are not applicable to pipelines orchestrating services owned by different providers. Our solution fills in the above gaps, by providing a framework for service-based data pipelines that support the selection of data processing services that maximize data quality while upholding privacy and security requirements. Service selection is driven by high-expressive policies, where data transformations built on data protection techniques (e.g., *k*-anonymity) are applied to data before they are used in the pipeline.

Data quality and data protection in service-based pipelines

As organizations increasingly realize the practical benefits and significant value of big data, they also acknowledge the limitations of current big data ecosystems, especially in terms of data governance. In this context, the need for privacy-aware systems enforcing sensitive data protection without compromising data quality throughout the entire data lifecycle arises. Recently, both industry and academia have begun to investigate the issue, recognizing the need of new security requirements [22] and the importance of addressing the conflict between the need to share and the need to protect information [23–27], from a data governance perspective [28, 29], and, more in general, to ensure compliance of (big) data pipelines with generic non-functional requirements [30, 31].

The pipeline template proposed in this work addresses these challenges by enabling 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 [32]. Table 4 provides a comparative analysis with relevant existing approaches, highlighting how few industrial solutions compare to our framework according to the following critical features:

- **F1 — Service-Based Pipeline Support in the Cloud-Edge Continuum:** The ability to effectively operate within distributed environments spanning cloud and edge infrastructure.
- **F2 — Quality-Aware Service Selection Ensuring Data Protection:** The capacity to optimize service selection processes, maintaining data quality across the pipeline and ensuring robust data protection measures.
- **F3 — Framework-Agnostic Data Protection:** The degree to which each solution is bound to specific data protection techniques.
- **F4 — Policy Expressiveness:** The degree to which each solution supports fine-grained specification of policies or privacy measures.

According to Table 4, most competitor solutions have full support for F3, while no solution has full support for F2. All solutions provide partial or full support for F1 and F4, with F4 fully supported by just two of the competitors. Microsoft Presidio aligns

Table 4 Comparative analysis with relevant existing approaches

Solution	F1	F2	F3	F4
Microsoft Presidio [33]	✓, can integrate within cloud-edge pipelines	~, focuses on data redaction	✓, compatible with diverse techniques	~, pre-built PII detectors with configurable policies
Apache Ranger [34]	~, mostly limited to cloud settings	✗, provides access control rather than service optimization	✓, integrates with various techniques	✓; high expressiveness with fine-grained policy control
Google Cloud DLP [35]	✓, primarily within Google Cloud	~, focuses on redaction and anonymization	✓, works across data types	~, flexible templates for data masking and redaction policies
AWS Macie [36]	~, suited for AWS cloud infrastructure	~, prioritizes data protection	✓, AWS-centric	~, supports predefined PII types but less customizable
IBM Guardium [37]	✓, supports hybrid cloud and on-prem setups	✗, focuses on monitoring and access control	✓, adaptable to multiple frameworks	✓, extensive policy-based access control and monitoring
Apache Sentry [38]	~, Hadoop ecosystems	✗, static access control	✗, closely tied to Hadoop	~, supports column and row-level access control
Our paper	✓, suitable for cloud-edge environments	✓, selection of services that optimize quality while ensuring protection	✓, data-protection techniques agnostic	✓, high expressiveness with fine-grained policy control

Feature support is classified according to ✓(fully supported), ~(partially supported or limited in scope), ✗ (not supported)

most closely with our approach, as it supports cloud-edge integration, offers compatibility with diverse techniques, and includes configurable policies for PII detection. However, our tool uniquely supports the optimization of data quality alongside privacy through a service selection feature and across the entire pipeline lifecycle. Additionally, unlike other solutions that are cloud-specific, our tool maintains compatibility across hybrid environments, addressing both cloud-edge and on-premise scenarios.

Additional solutions address individual aspects of these requirements. For example, several 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 [39–43]. Other approaches focus on specific databases, such as NoSQL or graph databases, or specific types of analytical pipelines [44–46]. 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 [47, 48]. The most similar to our approach are platform-independent solutions that adopt Attribute-Based Access Control (ABAC) [49] as a common underlying model, given its ability to support highly flexible and dynamic forms of data protection for business-critical data. For instance, Hu et al. [50] 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.

To conclude, the selection and composition of services, originally discussed in the Web service scenario, face additional challenges in the era of big data due to the volume and velocity of data, as well as the heterogeneity of services, domains, and hosting infrastructures. Despite its critical nature, security is often one of the least considered metrics in service selection [51]. Even when security is taken into account, it is not always coupled with data quality. Related work in this area includes approaches (e.g., [52]) 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 security metrics. Another relevant study [53] 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 relevant solution is [51], where the authors address the challenges of big service composition, with reference to 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. To address security issues, in their model, each service is assigned an L-Severity level [54] 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 also minimizes L-Severity (LS), data sources, and communication costs.

Conclusions and future work

In distributed service pipelines, ensuring both data quality and protection presents numerous challenges. This paper proposed a framework specifically designed to address this dual concern. Our data governance model integrates robust policies and continuous monitoring mechanisms to effectively address data security and privacy challenges. Simultaneously, it ensures the maximization of data quality to support efficient and reliable service pipeline generation. The key point of the framework lies in its ability to annotate each element of the pipeline with specific data protection requirements and functional specifications, and then drive service pipeline construction. This method enhances compliance with regulatory standards and improves data quality by preserving maximum information across the 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. Using 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. The paper leaves space for future work. First, we will extend our methodology with a taxonomy of possible quality dimensions and metrics supporting the definition of multidimensional data quality. Multiple dimensions and metrics will be adopted and weighted according to user priorities or task-specific requirements to better address the inherent multidimensional nature of data quality. This extension will enable more sophisticated monitoring and optimization mechanisms throughout the entire pipeline lifecycle. Second, we will evaluate the impact of different datasets and larger sets of services and configurations on our methodology. The primary objective is to identify generalizable patterns and recurring schemes that transcend specific experimental settings, thereby enhancing the broader applicability of our findings. Third, we will evaluate our methodology in different real-world production scenarios with the scope of evaluating its practical usability and utility, bridging the gap between theoretical and practical efficiency. Moreover, we plan to explore adaptive techniques based on machine learning for dynamic service selection, to increase the stability of data quality and privacy in varying operational conditions. Finally, we will extend our methodology to consider service quality assessment as a means to complement data quality evaluation, thus enabling the development of hybrid scenarios.

Author contributions

Marco Anisetti (M.A.) and Claudio A. Ardagna (C.A.A.) jointly conceived the original idea and guided 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 the whole 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.

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", v) Università Degli Studi di Milano through the APC initiative. 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.

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

Declarations

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable

Competing interests
The authors declare that they have no competing interests.

Received: 13 June 2024 Accepted: 25 February 2025
Published online: 10 March 2025

References

1. European Parliament, Council of the European Union: Regulation (EU) 2016/679 of the European Parliament and of the Council. https://data.europa.eu/eli/reg/2016/679/oj Accessed 2023-04-13.
2. U.S. Congress: Health Insurance Portability and Accountability Act of 1996. https://www.govinfo.gov/content/pkg/PLAW-104publ191/html/PLAW-104publ191.htm. Public Law 104-191, 110 Stat. 1936 (1996). https://www.govinfo.gov/content/pkg/PLAW-104publ191/html/PLAW-104publ191.htm.
3. Wang J, Liu Y, Li P, Lin Z, Sindakis S, Aggarwal S. Overview of data quality: examining the dimensions, antecedents, and impacts of data quality. J Knowl Econ. 2023;15(1):1159–78. https://doi.org/10.1007/s13132-022-01096-6.
4. Zhou Y, Tu F, Sha K, Ding J, Chen H. A survey on data quality dimensions and tools for machine learning. arXiv 2024. https://doi.org/10.48550/ARXIV.2406.19614 . https://arxiv.org/abs/2406.19614.
5. Palanisamy S, SuvithaVani P. A survey on rdbms and nosql databases mysql vs mongodb. In: 2020 International Conference on Computer Communication and Informatics (ICCCI), 2020;1–7. IEEE.
6. Thambiraja E, Ramesh G, Umarani DR. A survey on various most common encryption techniques. Int J Adv Res Comput Sci Softw Eng. 2012;2(7).
7. Woodruff A, Stonebraker M. Supporting fine-grained data lineage in a database visualization environment. In: Proceedings 13th International Conference on Data Engineering, 1997;91–102. IEEE.
8. 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). https://eur-lex.europa.eu/eli/reg/2016/679/oj Accessed 2022-08.
9. Fung BCM, Wang K, Chen R, Yu PS. Privacy-preserving data publishing: a survey of recent developments. ACM Comput Surv. 2010. https://doi.org/10.1145/1749603.1749605.
10. Aslanyan Z, Vasilikos P. Privacy-Preserving Machine Learning - A Practical Guide. Technical report, The Alexandra Institute 2020. https://www.alexandra.dk/wp-content/uploads/2020/10/Alexandra-Instituttet-whitepaper-Privacy-Preserving-Machine-Learning-A-Practical-Guide.pdf.
11. Rahman M, Hassan MR, Buyya R. Jaccard index based availability prediction in enterprise grids. Procedia Comput Sci. 2010;1(1):2707–16. https://doi.org/10.1016/j.procs.2010.04.304. (**ICCS 2010**).
12. 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.
13. 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.
14. Dwork C. Differential privacy: a survey of results. In: International Conference on Theory and Applications of Models of Computation, 2008;1–19. Springer.
15. Samarati P. Protecting respondents identities in microdata release. IEEE Trans Knowl Data Eng. 2001;13(6):1010–27.
16. Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. L-diversity: privacy beyond k-anonymity. ACM Trans Knowl Discov Data. 2007;1(1):3. https://doi.org/10.1145/1217299.1217302.
17. Majeed A, Lee S. Anonymization techniques for privacy preserving data publishing: a comprehensive survey. IEEE Access. 2021;9:8512–45. https://doi.org/10.1109/ACCESS.2020.3045700.
18. Fung BCM, Wang K, Fu AW-C, Yu PS. Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques, 1st edn. Chapman & Hall/CRC, New York 2010. https://doi.org/10.1201/9781420091502.

19. Sharma A, Singh G, Rehman S. A review of big data challenges and preserving privacy in big data. In: Kolhe ML, Tiwari S, Trivedi MC, Mishra KK, editors. Advances in data and information sciences. Singapore: Springer; 2020. p. 57–65. https://doi.org/10.1007/978-981-15-0694-9_7.
20. Wang RY, Strong DM. Beyond accuracy: what data quality means to data consumers. J Manag Inf Syst. 1996;12(4):5–33. https://doi.org/10.1080/07421222.1996.11518099.
21. Tayi GK, Ballou DP. Examining data quality. Commun ACM. 1998;41(2):54–7. https://doi.org/10.1145/269012.269021.
22. Colombo P, Ferrari E. Access control technologies for big data management systems: literature review and future trends. Cybersecurity. 2019;2(1):3. https://doi.org/10.1186/s42400-018-0020-9.
23. Geetha P, Naikodi C, Setty SLN. Design of big data privacy framework–a balancing act. In: Jain V, Chaudhary G, Taplamacioglu MC, Agarwal MS, editors. Advances in data sciences, security and applications. Singapore: Springer; 2020. p. 253–65. https://doi.org/10.1007/978-981-15-0372-6_19.
24. van den Broek T, van Veenstra AF. Governance of big data collaborations: how to balance regulatory compliance and disruptive innovation. Technol Forecasting Social Change. 2018;129:330–8. https://doi.org/10.1016/j.techfore.2017.09.040.
25. Ahlbrandt J, Brammen D, Majeed R, Lefering R, Semler S, Thun S, Walcher F, Rohrig R. Balancing the need for big data and patient data privacy-an it infrastructure for a decentralized emergency care research database. Stud Health Technol Inf. 2014;205:750–4. https://doi.org/10.3233/978-1-61499-432-9-750.
26. 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. Proc Natl Acad Sci. 2022. https://doi.org/10.1073/pnas.2104906119.
27. Creese S, Hopkins P, Pearson S, Shen Y. Data protection-aware design for cloud services. In: Jaatun MG, Zhao G, Rong C, editors. Cloud Comput. Berlin, Heidelberg: Springer; 2009. p. 119–30. https://doi.org/10.1007/978-3-642-10665-1_11.
28. Al-Badi A, Tarhini A, Khan AI. Exploring big data governance frameworks. Procedia Comput Sci. 2018;141:271–7. https://doi.org/10.1016/j.procs.2018.10.181.
29. Aissa MMB, 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;63–78. https://doi.org/10.1007/978-3-030-63396-7_5. Springer.
30. 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.
31. Ardagna CA, Bena N, Hebert C, Krotsiani M, Kloukinas C, Spanoudakis G. Big data assurance: an approach based on service- level agreements. Big Data. 2023. https://doi.org/10.1089/big.2021.0369.
32. Marco A, A, AC, 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.
33. Microsoft: Microsoft Presidio: Open-source tools for differential privacy and PII detection. https://github.com/microsoft/presidio. Accessed: 2024-10-30.
34. Apache Software Foundation: Apache Ranger: Framework to enable, monitor, and manage comprehensive data security across the Hadoop platform. https://ranger.apache.org/. Accessed: 2024-10-30.
35. Google Cloud: Google Cloud Data Loss Prevention (DLP). Available at: https://cloud.google.com/dlp 2023.
36. Amazon Web Services: AWS Macie. Available at: https://aws.amazon.com/macie/ 2023.
37. IBM: IBM Security Guardium Data Protection. Available at: https://www.ibm.com/products/guardium 2023.
38. Apache Software Foundation: Apache Sentry. Available at: https://sentry.apache.org/ 2023.
39. Rathore MM, Paul A, Ahmad A, Anisetti M, Jeon G. Hadoop-based intelligent care system (hics) analytical approach for big data in iot. ACM Trans Internet Technol. 2017;18(1):8–1824. https://doi.org/10.1145/3108936.
40. 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. Sustain Cities Soc. 2018;39:68–77. https://doi.org/10.1016/j.scs.2017.12.019.
41. Awaysheh FM, Alazab M, Gupta M, Pena TF, Cabaleiro JC. Next-generation big data federation access control: a reference model. Futur Gener Comput Syst. 2020;108:726–41. https://doi.org/10.1016/j.future.2020.02.052.
42. 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.
43. Gupta M, Patwa F, Sandhu R. Object-tagged RBAC model for the Hadoop ecosystem. In: Livraga G, Zhu S, editors. Data and applications security and privacy XXXI. Cham: Springer; 2017. p. 63–81. https://doi.org/10.1007/978-3-319-61176-1_4.
44. Chabin J, Ciferri CDA, Halfeld-Ferrari M, Hara CS, Penteado RRM. Role-based access control on graph databases. In: Bureš T, Dondi R, Gamper J, Guerrini G, Jurdziński T, Pahl C, Sikora F, Wong PWH, editors. Proc. of SOFSEM. Cham: Springer; 2021. p. 519–34. https://doi.org/10.1007/978-3-030-67731-2_38.
45. Gupta E, Sural S, Vaidya J, Atluri V. Enabling attribute-based access control in NoSQL Databases. IEEE Transactions on Emerging Topics in Computing, 2022;1–15. https://doi.org/10.1109/TETC.2022.3193577.
46. 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 AM, Khalil I, editors. Database and expert systems applications. Cham: Springer; 2019. p. 44–59. https://doi.org/10.1007/978-3-030-27615-7_4.
47. Preuveneers D, Joosen W. Towards Multi-party Policy-based Access Control in Federations of Cloud and Edge Microservices. In: Proc. of EuroS &PW, 2019;29–38.https://doi.org/10.1109/EuroSPW.2019.00010.
48. Matos E, Tiburski RT, Amaral LA, Hessel F. Providing Context-Aware Security for IoT Environments Through Context Sharing Feature. In: Proc. of IEEE TrustCom/BigDataSE, 2018;1711–1715. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00257.
49. Hill RC, 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.
50. 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.

51. Sellami M, Mezni H, Hacid MS. On the use of big data frameworks for big service composition. J Netw Comput Appl. 2020;166: 102732. https://doi.org/10.1016/j.jnca.2020.102732.
52. Carminati B, Ferrari E, K Hung, PC. Security conscious web service composition. In: 2006 IEEE International Conference on Web Services (ICWS'06), 2006;489–496. https://doi.org/10.1109/ICWS.2006.115.
53. Anisetti M, Ardagna CA, Bena N. Multi-dimensional certification of modern distributed systems. IEEE Trans Serv Comput. 2023;16(3):1999–2012. https://doi.org/10.1109/TSC.2022.3195071.
54. Vavilis S, Petković M, Zannone N. Data leakage quantification. In: Atluri V, Pernul G, editors. Data and applications security and privacy XXVIII. Berlin, Heidelberg: Springer; 2014. p. 98–113. https://doi.org/10.1007/978-3-662-43936-4_7.

Publisher's Note