Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Cost modelling and optimisation for cloud: a graph-based approach

Akif Quddus Khan[1*], Mihhail Matskin[2], Radu Prodan[3], Christoph Bussler[4], Dumitru Roman[5,6] and Ahmet Soylu[6]

Abstract

Cloud computing has become popular among individuals and enterprises due to its convenience, scalability, and flexibility. However, a major concern for many cloud service users is the rising cost of cloud resources. Since cloud computing uses a pay-per-use model, costs can add up quickly, and unexpected expenses can arise from a lack of visibility and control. The cost structure gets even more complicated when working with multi-cloud or hybrid environments. Businesses may spend much of their IT budget on cloud computing, and any savings can improve their competitiveness and financial stability. Hence, an efficient cloud cost management is crucial. To overcome this difficulty, new approaches and tools are being developed to provide greater oversight and command over cloud a graph-based approach for modelling cost elements and cloud resources and a potential way to solve the resulting constraint problem of cost optimisation. In this context, we primarily consider utilisation, cost, performance, and availability. The proposed approach is evaluated on three different user scenarios, and results indicate that it could be effective in cost modelling, cost optimisation, and scalability. This approach will eventually help organisations make informed decisions about cloud resource placement and manage the costs of software applications and data workflows deployed in single, hybrid, or multi-cloud environments.

Keywords Cloud computing, Cost optimisation, Cost modelling, Graph theory, Resource placement

Introduction

Cloud computing has seen an unprecedented surge in recent years, becoming integral to many organisations' IT strategies. As predicted by Gartner, by 2025, a staggering 85% of enterprises are expected to have adopted a cloud-first approach [46]. The benefits of cloud computing, such as scalability and flexibility, are widely recognized. However, the complexity of managing costs associated with cloud computing continues to pose a significant challenge. Cost optimisation becomes critical as more and more enterprises move their computing workloads to the cloud. The swift accumulation of cloud resource expenses demands proactive management to avert unforeseen and potentially costly fees. Navigating hybrid or multi-cloud environments adds to this complexity [43]. A thorough understanding of resource utilisation and carefully balancing cost, performance, and availability are prerequisites for effective cloud cost management. Accurately projecting future needs and modifying resource allocation appropriately are essential to preventing over- or under-provisioning problems. Successful cost optimisation can improve overall business performance and free up financial resources for other projects despite its difficulties. Significant funding has been allocated to research and development in cloud cost optimisation due to the cloud computing industry's exponential growth [13]. Despite these initiatives, there is still a critical need for workable and efficient solutions that help businesses better control the expenses associated with cloud computing.

*Correspondence:
Akif Quddus Khan
akif.q.khan@ntnu.no
[1] Norwegian University of Science and Technology, Gjøvik, Norway
[2] KTH Royal Institute of Technology, Stockholm, Sweden
[3] University of Klagenfurt, Klagenfurt, Austria
[4] Robert Bosch LLC, Sunnyvale, CA, USA
[5] SINTEF AS, Oslo, Norway
[6] OsloMet – Oslo Metropolitan University, Oslo, Norway

In response to this need, this article proposes an approach for modelling cost elements and cloud resources in the form of a graph. It further explores potential strategies for solving the resulting constraint problem of cost optimisation. This approach considers various factors, including utilisation, cost, performance, and availability. The significance of this approach is underscored by its potential to aid organisations in making informed decisions about cloud resource allocation. Moreover, it offers a practical solution for effectively managing cloud computing costs across a broad spectrum of software applications and data workflows [39, 47], whether in single, hybrid, or multi-cloud environments. This article, therefore, represents a step forward in the field of cloud cost optimisation, moving from a theoretical proposal to practical implementation. The work presented in this article is an extension of our preliminary work on proposing a graph-based solution for cloud resource placement and cost optimisation [25]. More specifically, it significantly extends our earlier work by providing:

- an implementation of the proposed graph-based approach using Google pricing API[1];
- an evaluation of the proposed approach on different deployment scenarios of big data pipelines;
- a review of scientific literature related to cost optimisation using cloud resource placement; and,
- a review of the scientific literature related to graph theory being used for the optimisation problem.

The rest of the article is organized as follows. "Background" section provides the background including various cloud computing cost elements and related terms, while "Related work" section presents the related scientific literature. "Cost optimisation paths" section discusses a set of paths for cloud cost optimisation in general, while "Graph-based cloud cost modelling and optimisation" section presents the proposed graph-based cloud cost modelling and optimisation approach. "Case study" section discusses a detailed case study for a big data infrastructure on which evaluations are performed, and "Evaluation" section describes implementation details and evaluations. Finally, "Discussion & conclusions" section provides a discussion and concludes the article.

Background

The section overviews three fundamental areas underpinning our cloud cost optimisation solution. The first subsection discusses different elements of cloud cost structure, highlighting the complexity of cost structure in cloud environments and its challenges. The second subsection discusses big data pipelines and various elements such as data ingestion, data warehouses, and data marts. The final subsection discusses graph theory and related algorithms. The aim is to help understand the context and theoretical underpinnings of the proposed graph-based cloud cost optimisation approach.

Cloud computing cost

Cloud computing is a model for delivering on-demand computing resources over the Internet. This article considers three high-level cost categories: compute, data transfer, and storage. Compute cost includes the cost of virtual machines, containers, serverless functions, etc. Data transfer cost includes transferring data within the cloud service providers' (CSP) network and to/from an external network. We consider four categories of cloud storage costs: data storage, network usage, transaction, and data replication [27]. Data storage is the cost of storing data in the cloud, which is charged on a GB-per-month basis. Different storage tiers have different pricing, and some CSPs offer block-rate pricing, where the larger the amount of data, the lower the unit costs. Transaction costs are associated with managing, monitoring, and controlling a transaction when reading or writing data to cloud storage. Cloud storage providers charge for the amount of data transferred over the network and the number of operations it takes. Network usage cost is based on the amount of data transferred over the network. Data replication cost refers to the cost of replicating data from on-premises storage to the cloud or from one instance to another. By default, three copies are generally stored for each chunk of uploaded data to achieve high data reliability and better disaster recovery [30]. In addition, there are several optional costs, including data management, data backup, and data security. Users can optimise mandatory cost elements, but they cannot avoid them.

In short, understanding the cost structure of cloud computing can be a difficult and intricate task due to the complex pricing models offered by various CSPs. Comparing costs and selecting the most suitable option for a particular application can be challenging. Researchers have attempted to simplify it to make it easier for users to comprehend the complexity of the cloud cost structure, e.g., Fig. 1 shows a taxonomy of cloud computing cost elements and how they are inter-related and overlapping at the same time, see also [35]. Martens et al. [36] have observed that many cloud cost evaluations lack a systematic approach to cost estimation, which is necessary to understand the varying pricing models of cloud services. When selecting a CSP, the cost is not the only factor to consider. Other quality of service (QoS) elements exist, such as network performance, data availability,

[1] https://cloud.google.com/billing/docs/how-to/get-pricing-information-api

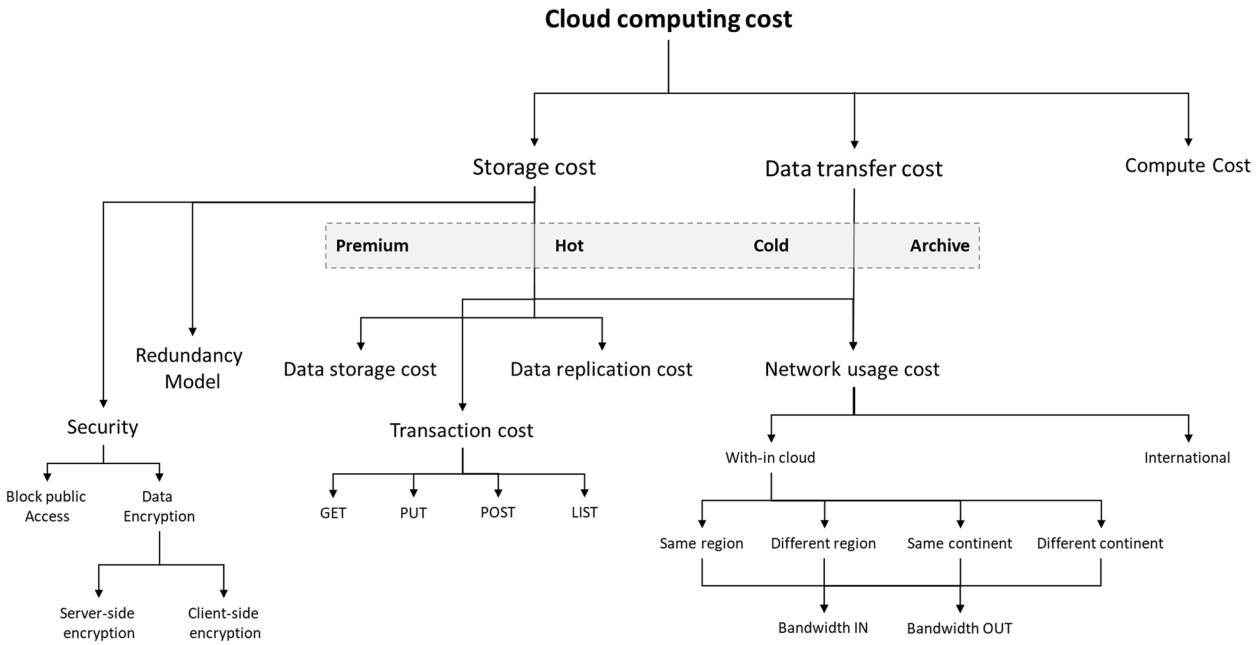

Fig. 1 Cloud computing cost taxonomy [30]

consistency, security, etc. This gives rise to inevitable trade-offs such as storage-computation, storage-cache, storage-network, availability-reliability, and cost-performance [30, 35], which means balancing different factors to make decisions about resource allocation and use. These must be considered when deploying applications and infrastructure to the cloud. A viable solution should be to find the optimal resource placement strategy for performance by quantifying the QoS elements.

Big data pipelines

Nowadays, data has become a valuable asset. It is often compared to oil due to its potential to drive growth and innovation. Data could be bought and sold on data markets, i.e., platforms where data providers and data consumers come together [8, 9, 44]. These markets enable harnessing vast data types in large volumes, enhancing the value of the end products or services. The term "big data" is widely used but lacks a formal definition. However, it is commonly characterized by several "Vs": volume, velocity, and variety [3]. Volume refers to the sheer amount of data generated every second from various sources like social media, business transactions, sensors, and more. Velocity pertains to the speed at which this data is generated and processed. In many cases, to be helpful, the data must be analyzed in near real-time. Variety denotes the different types of data, including structured data (like databases), unstructured data (like text), and semi-structured data (like XML files). Big data pipelines are designed to handle these three features. They are a series of data processing steps where the output of one step is the input of the next. These pipelines are crucial for transforming raw data into insights and can include data collection, cleaning, integration, analysis, and visualisation steps. The design of these pipelines can vary greatly depending on the specific requirements of the big data application. For instance, a pipeline for real-time data analytics might prioritize velocity, while a pipeline for a machine learning model might focus more on volume and variety.

Data ingestion

Data ingestion is the first phase of a big data analytics solution. Data ingestion (acquisition) moves data from multiple sources, such as databases, IoT devices, websites, streaming services, etc., to a target system to be transformed for further processing. Data comes in various forms and can be structured or unstructured [6]. The process of assembling data from multiple heterogeneous sources into a data repository (data lake or data warehouse) is called data ingestion. This process is essential in fields that rely on highly distributed data that needs to be appropriately stored in a data repository, such as artificial intelligence, machine learning, data science, data analytics, and knowledge discovery in databases. Despite its significance, data ingestion is sometimes viewed as a preliminary phase in data analysis and is therefore given less weight. On the other hand, it entails highly

significant access and integration functions with widely distributed data components acquired from various operating systems, applications, and hardware. Bringing disparate data into a single data repository is the aim of data ingestion [21].

Date warehouse

A data warehouse (DW) is a central repository for aggregating data from various sources into a unified, consistent format. From a technical standpoint, a data warehouse is akin to a relational database optimised for reading, aggregating, and querying large volumes of data. Traditionally, DWs primarily contained structured data-information neatly arranged in tables. However, modern data warehouses have also evolved to accommodate unstructured data, including images, PDF files, and audio formats [11]. Data warehouses collect and transform data from various sources, such as transactional databases, spreadsheets, logs, and external APIs. These data sets undergo transformation, cleansing, and integration, often using Extract, Transform, and Load (ETL) techniques to create a cohesive whole. Data warehouses are optimised for analytics, enabling complex analytical queries, aggregations, and reporting, making them invaluable for business intelligence and decision-making. Modern data warehouses can accommodate structured and unstructured data, allowing organisations to extract insights from various data types. They typically follow a three-tier architecture, with the bottom tier housing the data warehouse server, the middle tier containing an online analytical processing server for fast query speeds, and the top tier providing a user interface or reporting tool for ad-hoc data analysis. Cloud-based data warehouses have gained prominence, offering scalability, elasticity, and cost-effectiveness. The benefits of data warehouses include centralized data, improved decision-making, performance optimisation, and data consistency, making them an essential tool for organisations [21].

Data marts

With a typical size of less than 100GB, a data mart is a smaller version of a data warehouse. They become essential when a company's size and volume of data increase to the point where conducting an enterprise data warehouse search takes too long and is no longer productive. Rather, data marts are designed to make it simple and quick for other departments (such as sales, marketing, and the C-suite) to obtain pertinent information. A data mart is a type of storage system that holds extremely specific information targeted at the needs of workers in a particular department or at the main workstream of an organisation. It is a component of the data lake concept, where data are provided in their original format, making analytics and the presentation of aggregated data challenging. This problem is resolved by using data marts [5]. This system represents the environments of stored data. The data mart is a small-sized data warehouse focused on a specific subject. While a data warehouse is meant for an entire enterprise, a data mart is built to address the specific analysis needs of a business unit. Thus, an enterprise usually has many data marts [52].

Data lake

A Data Lake is a relatively new concept in data management and big data. It is a system or repository of data stored in its natural/raw format, usually object blobs or files. The concept of Data Lake was put forward by Dixon to face the challenges of big data and the deficiencies of Data Warehouses [45, 54]. Data Lake has neither a standard definition nor an acknowledged architecture. However, some researchers have proposed a complete definition and a generic and extensible architecture of a data lake [45]. They can quickly process and store data, regardless of format and size, from structured tables to unstructured text such as emails, images, or videos. Data lakes allow various data types and sources to be available in one location, supporting statistical discovery [14]. They are often designed for low-cost storage, so they can house a high volume of data at a relatively low price as they require low-cost hardware, and most technologies used to manage data in a data lake are open source like Hadoop [23]. Moreover, they are highly agile. Data scientists can prepare and analyze data models rapidly [1].

Graph theory

Graph theory, as a mathematical discipline, studies graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context refers to a collection of vertices or nodes and a collection of edges that connect pairs of vertices [55]. This allows us to model and analyze the structure of a network, providing a robust framework for both quantitative and qualitative approaches [10]. In the context of big data application deployment in the cloud environment, graph theory can be beneficial. One or more shortest-path algorithms or graph-neural networks (GNN) can be employed to find the optimal solution [50]. These tools allow us to navigate the graph most efficiently, identifying the shortest or most cost-effective path between two vertices. Shortest path algorithms are specific algorithms designed to solve the shortest path problem. The problem is to find the shortest path or paths from a starting point to a destination, given a graph where each edge has a length or cost. Some of the most popular shortest path algorithms include Dijkstra's algorithm [19], Bellman-Ford algorithm [4], A* algorithm [48], Floyd-Warshall

algorithm [7], and Johnson's algorithm [22]. Each of these algorithms has its strengths and weaknesses, and the choice of algorithm can depend on the specific characteristics of the graph and the problem at hand. On the other hand, GNNs [37] are a type of artificial neural network specifically designed to operate on graph-structured data. They can process the graph's topological information and the features of its nodes and edges. GNNs have shown great promise in various applications, including social network analysis, molecular chemistry, and recommendation systems, to name a few. Graph theory offers an efficient mathematical framework for modelling and analyzing network structures, while shortest path algorithms and GNN provide practical tools for navigating these structures and finding optimal solutions. These techniques can be particularly valuable in big data and cloud computing, where efficient resource allocation and data processing are critical.

Related work

The field of cloud cost optimisation has received significant attention in recent years, with numerous studies exploring different approaches to reducing cloud computing costs. The proposed approach is mainly relevant to resource placement and network usage cost optimisation; hence, scientific literature related to these topics is discussed in this section.

Cloud cost optimisation

Cloud cost optimisation is a wide topic and includes several different directions. However, in this section, the related scientific literature discussed concerns cloud cost optimisation techniques for network usage cost reduction, either by efficient resource placement, caching techniques, or optimising the data replication/migration process. Khan et al. [26, 28] developed a ranking method for available storage options based on five key parameters: cost, proximity, network performance, the impact of server-side encryption, and user weights and presented the effectiveness in terms of data transfer performance and the feasibility of dynamic selection of storage option. However, the framework primarily focuses on cloud storage and not other services. For network cost optimisation, Mansouri et al. [34] proposed an approach to minimize the cost of data placement for applications with time-varying workloads. Hence, the scope of work is too narrow and cannot meet diverse application requirements. Zeng et al. [57] proposed a method with a slightly different scope, i.e., a method to deploy edge servers in wireless metropolitan area networks economically. The proposed approach is related to resource placement; however, it is only focused on wireless networks. There is another approach with a similar scope proposed by Shao et al. [51], a data placement strategy for IoT services in wireless networks, which considers user distribution density to determine optimal edge server deployment locations and minimize deployment costs.

Caching techniques can be employed to reduce network usage. Ghoreishi et al. [18] proposed a cost-effective caching as a service (CaaS) framework for virtual video caching in 5G mobile networks. For evaluation, two virtual caching problems were formulated. Results obtained have shown significant performance enhancement of the proposed system regarding return on investment, quality, offloaded traffic, and storage efficiency. This technique focuses solely on using caching to reduce network costs and does not involve anything related to resource placement. Another caching approach is proposed by Kumar et al. [31]. They presented a framework for cost-efficient content placement in a cloud-based content delivery network. The proposed approach uses a cost matrix and a crosslinking data structure to minimize replica dependencies and optimise content delivery. Data replication and migration consume many network resources, resulting in high network usage costs. Mansouri et al. [34] suggest two techniques for dynamic replication and data migration in cloud data centers (i.e., network cost). The first approach uses dynamic and linear programming techniques, assuming that accurate information about the burden on objects is readily available. The second method uses a randomized The "Receding Horizon Control" (RHC) approach uses knowledge about potential future workloads. Dong et al. [12] proposed an online, cost-efficient transmission scheme for cloud users to address the issue of selecting high service levels that result in unnecessary resource waste. The proposed method utilizes an information-agnostic approach, where long-term transmission requests are split into a series of short-term ones, allowing for more efficient utilisation of cloud resources. In short, techniques such as dynamic replication, data migration, and heuristic strategies address the dynamic nature of network costs. Additionally, content delivery networks, caching frequently accessed data, and cost-efficient content placement frameworks contribute to reducing network infrastructure use. As stated earlier, these approaches focus on network usage cost optimisation; since network cost is a sub-element of total storage cost, they impact reducing the overall storage cost in the cloud.

Storage selection is an essential step before any application deployment. The selection could be done to find the cheapest possible option while meeting certain QoS requirements. Following are some storage selection techniques from the scientific literature. Ilieva et al. [20] proposed a new approach for evaluating and ranking

cloud services, which combines multi-criteria and fuzzy approaches to consider various factors. Oki et al. [40] presented selection models for cloud storage to satisfy data availability requirements, and Halimi et al. [38] proposed a QoS-focused approach for storage service allocation that considers various QoS objectives to improve the performance and scalability of cloud storage systems. Liu and Shen [32] proposed a method for efficient storage resource distribution, which includes three enhancement strategies to reduce payment cost and service latency. These proposed algorithms may not work effectively in real-world situations where workloads and prices are more dynamic and complex. The data placement strategies may not always reflect user behavior and usage patterns, and the methods for large-scale deployment may be unfeasible. Since network usage cost is an integral and expensive cost element [24], in practice, one would avoid accessing data remotely and instead move the computation close to the data. An example could be having the same software application step available in different regions, having the data available in different regions, and accessing the data locally in the same region by the software application (assuming the data passed between steps is small compared to the data accessed within a step). The comparison presented in [24, 29] indicates that storing multiple copies of data is more cost-effective than repeatedly transferring it over the network for processing.

Graph-theory for optimisation

Graph theory, a powerful decision-making method, has been used since 1736 and has found applications in various fields to address optimisation problems [49]. Gandhi and Agrawal [16] used graph and matrix approaches to carry out failure modes and effects analysis on both hydraulic and mechanical systems. They achieved this rigor, sensitivity, and impact of imminent failures by systematically identifying potential failure modes, highlighting impact, and prioritizing them for further corrective measures. Similarly, Upadhyay [53] utilized graph theory and matrix algebra to construct and analyze the structure of an intelligent mobile learning environment. This approach gave information about dependency and interconnections in the learning environment, resulting in the design of more effective strategies. In their study, Al-Hakim et al. [2] used graph theory to illustrate a product and connect its parts. It is an illustration of the product's structure, making the identification of weak elements easier and helping with improved resource management. Pishvaee Mir and Rabbani [41] gave a graph-theoretic algorithm that can help design supply chain networks. Their algorithm took both shipments and indirect shipments into consideration and was thus able to devise the most efficient routes that would result in a more efficient supply chain.

Raj et al. [42] designed a digraph and matrix model to investigate the number of obstacles preventing flexible manufacturing systems implementation. Goyal and Grover [49] combined graph-theoretical analysis with fuzzy theory to choose the most suitable advanced manufacturing system from given options. Their approach considered factors such as costs, quality, and system flexibility. This helped companies make informed decisions about manufacturing systems. Although graph theory has not been expressly utilized for cloud resource placement and cost optimisation, its effectiveness in addressing optimisation problems across several fields indicates its potential relevance in this particular subject. Graph theory's capacity to represent intricate systems and determine the most efficient routes makes it a promising instrument for addressing the difficulties of allocating resources and optimising costs in cloud computing settings. This field is very suitable for research and has the potential to produce substantial advantages in the effective exploitation of cloud resources.

Discussion

The related work demonstrates a growing body of research focused on cloud cost optimisation, with various optimisation algorithms proposed and evaluated. However, these algorithms do not address industry-specific requirements, and most are not evaluated in real-world scenarios. Our cost optimisation approach, which employs graph theory, represents a new approach to cloud cost optimisation. We provide a novel approach to cloud cost optimisation by modelling cost elements and cloud resources in a graph and quantifying the impact of QoS elements to address the trade-offs. Graph theory is effective in finding optimised solutions for problems in a wide range of domains, including the cloud environment, such as data replication [33] and caching [56]; therefore, we find it promising for the cloud cost modelling and optimisation as well. The proposed approach should apply to new and existing applications in the cloud, be able to address multiple challenges, such as compute and storage resource placement and network cost optimisation, and incorporate QoS requirements.

Such an optimisation approach is currently missing in the literature, creating a substantial challenge to cloud resources' economical and effective utilisation. It is important to note that implementing the proposed approach may necessitate changes to the system's architecture and behavior, which could be a trade-off depending on the changes made. The proposed approach has several advantages, including increased efficiency and cost savings. According to the proposed approach, a cloud

service instance is conceptualized as a node for processing or storage in each region. The best path between nodes is then found using graph theory, which lowers expenses by placing cloud resources as efficiently as possible. The information flow between the nodes will be pre-defined, and the data transfer cost from one node to another will be specified on the edge connecting the two nodes and treated as the weight to minimize network usage cost. Resources from different regions, such as Europe West, Europe East, and possibly the US East, may be included in the application architecture. However, CSPs offer various options for every location, including EUWest1, EUWest2, EUEast1, EUEast2, and so forth. The algorithm will determine the optimal path between the selected locations based on the resources chosen in the designed architecture that outlines cloud resource deployment and data flow for a software application, considering resource specifications, availability, and geographical limitations.

Cost optimisation paths

Cloud cost structure consists of several elements, and optimisation can be done by simultaneously targeting one or more elements. Hence, we break down the complete process of cloud cost optimisation into six paths, as described in what follows.

1. **Optimisation of the network cost by proposing new locations for cloud resources**: This path focuses on identifying the most suitable locations for compute and storage resources to ensure optimal performance and cost efficiency. By analyzing data access patterns and workload requirements, the solution shall suggest new locations for the resources that can achieve the best possible balance between performance and cost. However, if it is done post-deployment, migration costs will apply.
2. **Optimisation of storage cost by proposing the optimal number of storage instances**: This path expands the optimisation approach to include storage, backup, and archiving. By proposing new locations and the optimal number of storage instances, the system can ensure that data is stored cost-effectively and efficiently. In a hybrid cloud, a storage instance is a server and, for the public, a storage bucket in a zone/region.
3. **Optimisation of compute cost by proposing the optimal number of compute resources (VMs, GPU):** By carefully selecting the number of virtual machines and GPUs needed for a specific workload, the overall cost of compute resources can be minimized, while still ensuring that the application runs smoothly and efficiently.
4. **Optimisation of compute cost by scaling compute resources**: This path focuses on optimising the system's compute resources to ensure they are appropriately scaled to meet the workload requirements. Analyzing the workload patterns and scaling the compute resources accordingly can reduce costs and improve performance.
5. **Optimising storage costs through data migration between storage tiers**: This path involves migrating data between storage tiers to save cost by identifying opportunities to move data to lower-cost storage tiers without compromising performance. It can be done by analyzing data access patterns and workload requirements.
6. **Proposing more efficient and cost-effective resource alternatives**: This path involves identifying alternative solutions that can provide the same or better performance while reducing the cost. By analyzing the system's current configuration and workload requirements, the proposed solution can suggest better and more cost-effective alternatives. These could include different CSP or hardware options like spot VMs instead of regular VMs, containerisation instead of VMs, or serverless computing instead of fixed provisioning. However, this may require changes in the architecture/implementation of the system.

An ideal cost optimisation scenario containing all six paths is exemplified in Fig. 2. The goal is to show how each optimisation path will work when put into action. The proposed architecture is not the actual representation of the optimised scenario. The designed architecture, which could be for a large software application or a multi-step big data workflow, has data storage in five locations, with compute resources placed at two locations near all storage servers. Additionally, a separate data lake is deployed for the data archive. After passing through the cost optimisation techniques, this architecture gets modified based on user requirements, such as required resources and their location, as well as data access patterns (i.e., proposed architecture). This scenario can offer several benefits. Firstly, moving them closer to the data store will significantly reduce data transfer costs, assuming that computing resources only access one data storage. Secondly, centralizing the system status and moving the data archive to the same location as the storage will reduce overhead costs and improve overall efficiency.

Graph-based cloud cost modelling and optimisation

The proposed approach, shown in Fig. 3, aims to find the most suitable placement for cloud service instances in terms of storage and compute resources,

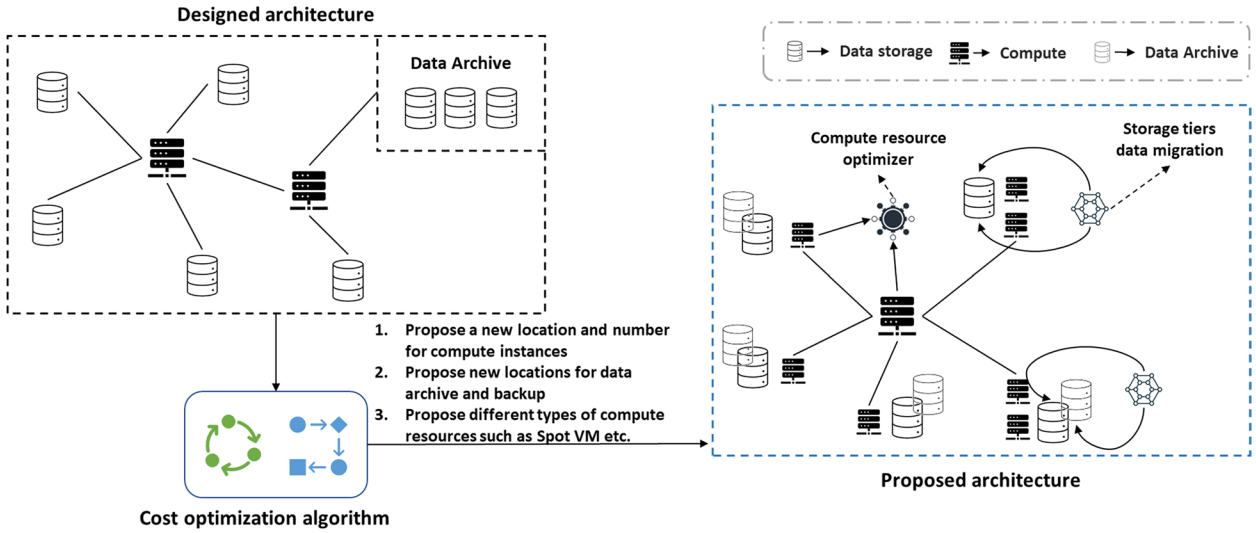

Fig. 2 High-level diagram illustrating an ideal cost optimisation scenario

hence optimising network usage costs (i.e., path 1) and can potentially target paths 2 and 3 as described in "Cost optimisation paths" section to optimise compute and storage costs. This involves the development of a model that considers the number and location of cloud services deployed, the data access patterns, and an algorithm to suggest the most efficient location for cloud service instances. To implement the proposed approach, the following steps are required.

Purpose clarification

The first step is to clarify the purpose of the application. It is an essential first step that involves the identification of both the functional and non-functional requirements that are specific to the industry. Functional requirements include the specific tasks that the application should be able to perform, such as data storage, data processing, data retrieval, and data analysis. On the other hand, non-functional requirements

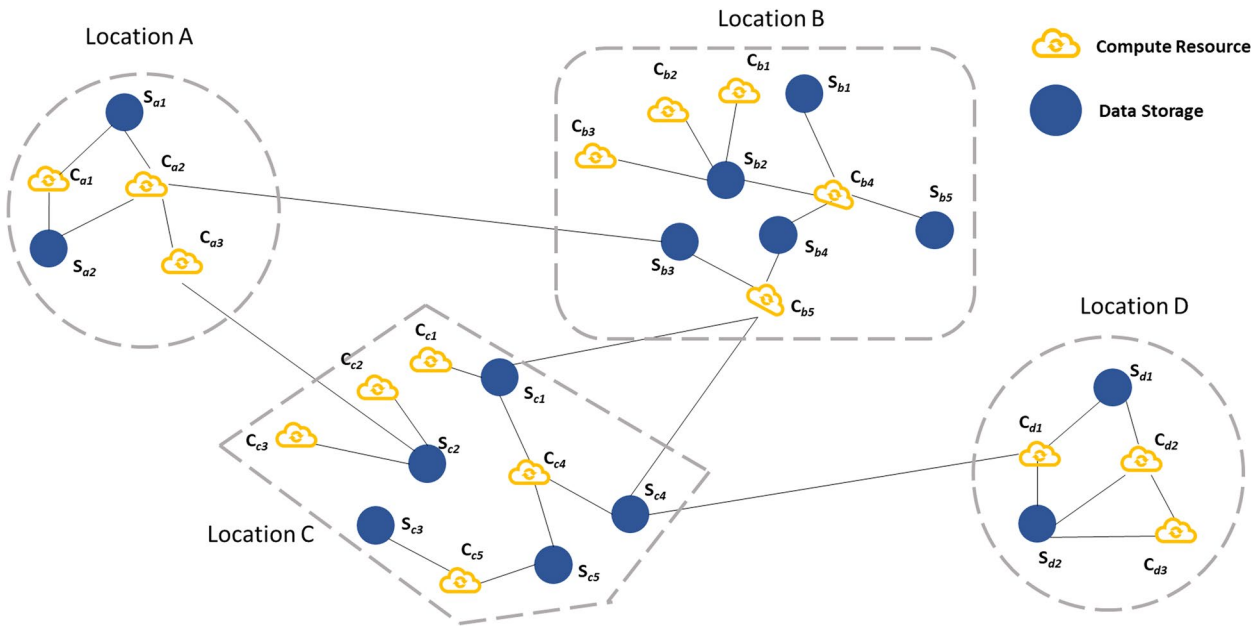

Fig. 3 Illustration of directed graph data structure based on required cloud resources for big data applications deployed in a cloud environment

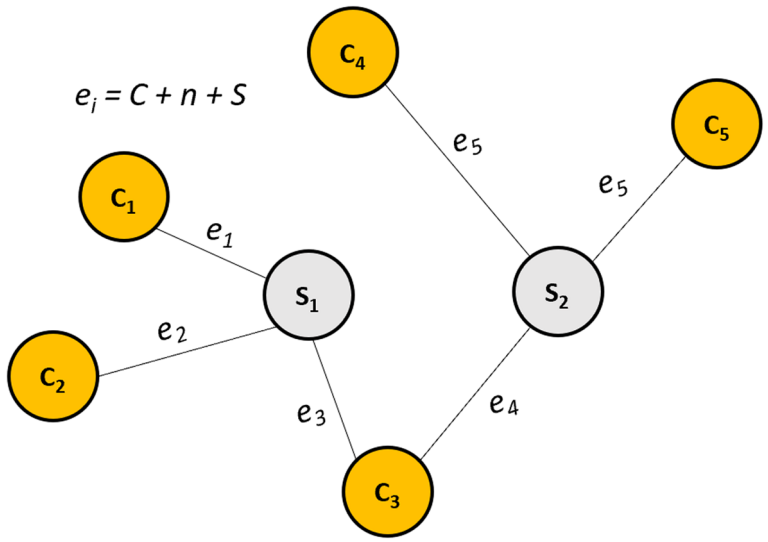

Fig. 4 Elements of graph model. Each compute and storage instance is represented as a node and labeled as *C* and *S*, respectively. *e* represents the cost of these resources as the cost of network usage for data transfer between any two resources

include the quality attributes that the application should possess, such as latency, availability, and durability. Considering these requirements makes it possible to determine the necessary resources, including storage servers and compute resources. The cost-effectiveness of deploying the resources is also influenced by the requirements, as the resources must meet the necessary functional and non-functional requirements while also being cost-efficient.

Designed architecture

An architecture for the software application must be designed according to this proposed approach. The architecture describes the kinds of cloud resources used, where they will be deployed geographically, and how the data will flow within the application. It will specify the limitations on what resources can be moved, how much (within a country, region, or continent), and the specifications for those resources (the quantity of RAM, CPU, and storage space needed). The number and location of cloud resources, data flow, QoS requirements, and other variables must all be considered. The next step's graph creation will be based on the output from this step. Verifying the availability of cloud resources in designated regions and defining the data flow within the application for effective processing and storage are two crucial aspects that must be considered during the design phase to guarantee the viability of the architecture. To optimise the application for effectiveness and cost, we can use the information provided by the designed architecture based on graphs to guide our proposed approach.

Graph creation

Creating a graph based on the information gathered from the previous steps presents a few challenges. Cloud resources can be treated as graph nodes, and the network usage cost can be specified as the edge. However, multiple cost elements, such as storage, computing, security, and other cloud resource costs, must be considered. As seen in Fig. 4, the nodes with labels C_x represent compute resources, while the nodes with labels S_x represent storage resources. Figure 3 presents a more detailed example where $C_{a1}, C_{a2}...$ are the compute resource costs, and $S_{a1}, S_{a2}...$ are the storage resource costs. We denote the edges as e_i, representing the weights of the graph and, in this case, the costs of services. Storage and computing costs are not the only factors to be considered when finding the cost-optimal solution. To ensure that all costs are accounted for, we must also consider the costs of services, such as network usage costs. To solve this problem, we can treat n_{ij} as the network cost between the nodes C_i and S_j, hence, e_{ij} be as follows:[2]

$$e_{ij} = C_i + n_{ij} + S_j \tag{1}$$

To incorporate QoS elements, we assume latency, availability, and durability as l_{ij}, a_{ij} (quantified value based on the SLAs), and d_{ij} (numerical representation of the redundancy model), respectively. Similarly, w_l, w_a, and w_d are weighted in percentage to define the importance of each factor as per requirements and N as the normalizing

[2] Equation 1 refers to a general formula; multiple variations might be used to avoid redundancy in the cost calculation.

constant. Using a cost-effectiveness ratio (CER), we can quantify l_{ij}, a_{ij}, and d_{ij} as:

$$f(CER_l|CER_a|CER_d) = \frac{Cost}{(1-(l_{ij}|a_{ij}|d_{ij}) \times N)} \times w_{(l|a|d)} \quad (2)$$

For example, if the cost of *Server A* is \$150 and its latency performance is 0.3[3], the cost-effectiveness ratio would be: $\frac{150}{(1-0.3)\times 10} = 21.42\frac{\$}{sec.}$. Similarly, if *Server B* has a cost of \$200 and a latency performance of 0.2, then its cost-effectiveness ratio would be: $\frac{200}{(1-0.2)\times 10} = 25\frac{\$}{sec.}$. In this case, *Server A* would be more cost-efficient than *Server B*.

Hence, $f(QoS)$ will be:

$$f(QoS)_{ij} = f(CER_l)_{ij} + f(CER_a)_{ij} + f(CER_d)_{ij} \quad (3)$$

Putting Eqs. 1, 2 and 3 together, we will get:

$$e_{ij} = C_i + n_{ij} + S_j + f(QoS)_{ij} \quad (4)$$

This way, additional costs, such as security and encryption, can also be included in the calculation. Once the weights are specified on the edges, all possible resource combinations in a graph can be formed. For example, as per Fig. 3, *Location A* has three compute and two storage instances; similarly, *Location B* has five compute and four storage instances. Assuming only one storage and one compute instance are required in each location, the number of resource combinations in the graph for just these two locations will be $Num(S)_A \times Num(C)_B + Num(S)_B \times Num(C)_C$, which in this case will be $(2 \times 5) + (3 \times 4) = 22$. The total number of combinations can be calculated using Eq. 5. The complexity and the total number can be increased when more resources are required in each location.

$$Total = \Sigma_{x=1}^{n}(Num(S)_x \times Num(C)_{x+1}) \quad (5)$$

Implementation process

The implementation of the proposed approach begins with the collection and normalisation of input data. The following are the details of the execution steps that are carried out:

1. Obtain a list of regions where the desired services are available within the specified geographic location. This information will be utilized to create the graph nodes.
2. Using the CSPs' "Pricing API", such as the one provided by Google[4], for cost estimations of the requested services, the weights of the edges will be set.
3. Nodes and edges obtained in the previous steps will be combined and transformed into a graph. This data structure will serve as the foundation for the next step.
4. Find the optimal solution that minimizes the cost and satisfies the necessary performance requirements by addressing the constraint problem. These include, for example, trade-offs, a limit on the amount of storage or compute resources available within a specific cloud provider or location, or QoS requirements of the applications running on the cloud resources, which could impact the selection of suitable cloud resources for cost optimisation.

Selected algorithm

Although many algorithms could be used to solve this problem, Dijkstra's algorithm is employed in this article. Dijkstra's algorithm is renowned for finding the shortest paths between nodes in a weighted graph. Conceived by computer scientist Edsger W. Dijkstra in 1956, the algorithm has found extensive application in areas such as network routing protocols [19]. The algorithm works on the heuristic that nodes are labeled and that the node with the shortest label at each stage is greedily selected. It makes the assumption that there is never a negative distance between any two nodes. The labels consist of two elements: the node that comes before the current node in the shortest path and an upper bound on the shortest path length from the source node to the node in question. Until it discovers the shortest routes from the source to all other nodes, the algorithm iteratively changes these labels [17]. Although Dijkstra's algorithm is quite old, it is still being studied; most recently, possibilities have been explored for using it in conjunction with predictions from machine learning [15]. Since the scenarios addressed in this article do not require extensive computation, this algorithm is suitable as it is simple, lightweight, and easy to implement.

User interface

A user interface is provided to demonstrate the implementation of the approach and the corresponding process from the users' perspective. The interface implemented allows users to select an instance type and related parameters, as shown in Fig. 5. It starts with the option to select the resource type; based on the selected option, further parameters appear, such as total storage, storage class, etc. At the bottom of the interface, there is an "Architecture Graph". This graph visually represents the steps added to the pipeline architecture. In this

[3] Latency performance of 300ms divided by 1000.
[4] https://cloud.google.com/billing/docs/how-to/get-pricing-information-api

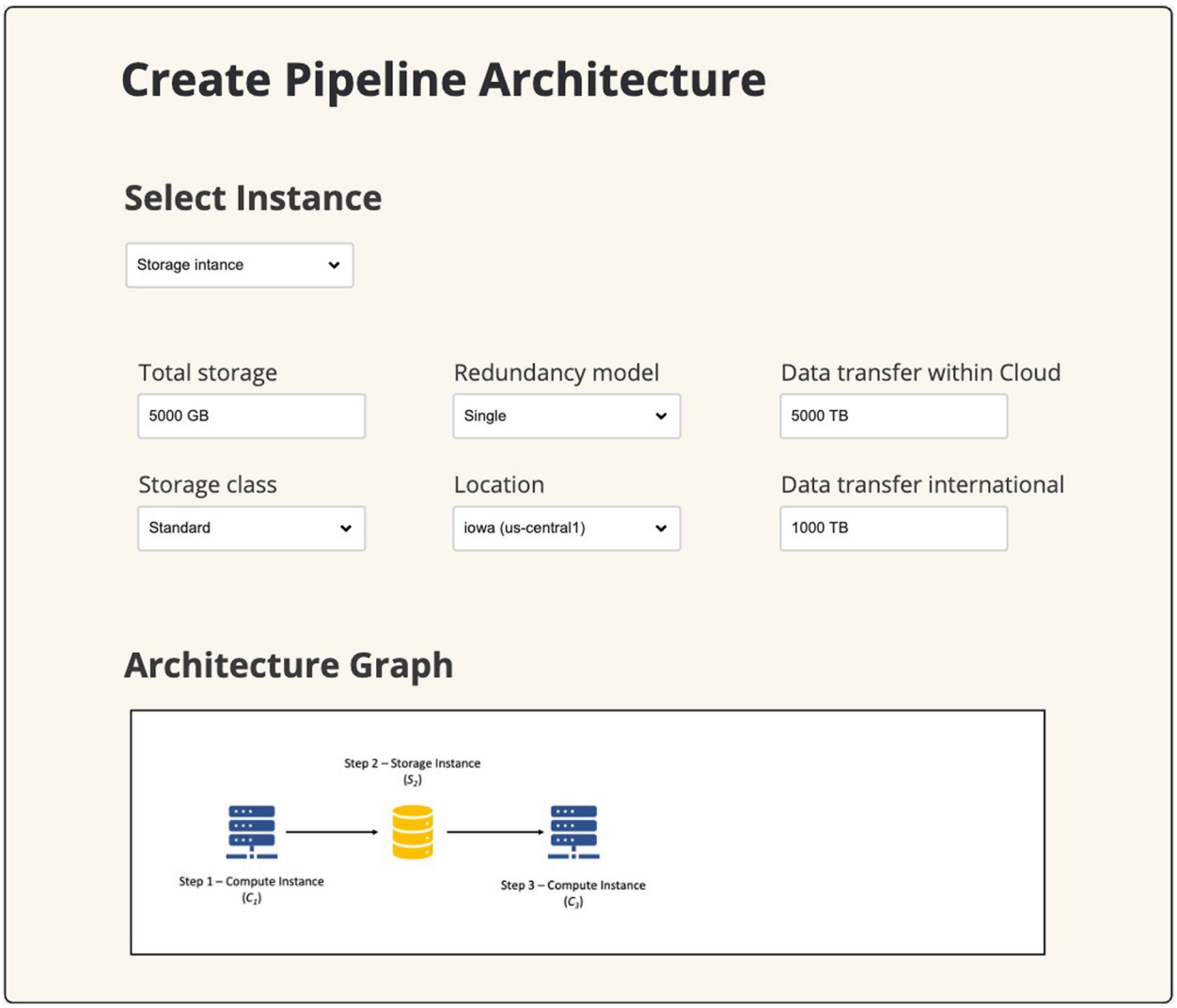

Fig. 5 User interface for creating an architecture to be used as an input for the graph-based cost optimisation model

scenario, three steps are displayed: a compute instance, a storage instance, and an additional compute instance. The interface makes it simple for users to choose and modify parameters to suit their needs. It offers an orderly and transparent method for building a pipeline architecture in a cloud computing setting. The process is made even easier to understand by the architecture's visual representation, which is an important step toward improving the approach's usability and accessibility through this interface.

Case study

To perform evaluations, we provide a case study of the data engineering process in this section, utilizing an ETL pipeline. Figure 6, which illustrates the steps involved, shows the most basic data engineering process. The process begins with the ingestion of data from multiple sources. Among other things, these sources may be software programs, sensors, and IoT devices. Because of the variety of sources used, the dataset is rich and diversified, which can result in more accurate and informative studies. In this article, this step is also called the "Data Source", emphasizing its function as the data source to be processed. The data is sent to a computing step for processing after it has been ingested. This step is referred to as "Data Transformation". The raw data is cleaned, normalized, and essentially ready for analysis at this point in the process. This could entail handling missing values, eliminating outliers, or formatting the data so that it can be analyzed. A storage instance is used to store the transformed data. Depending on what the project requires, this storage might take on a variety of shapes.

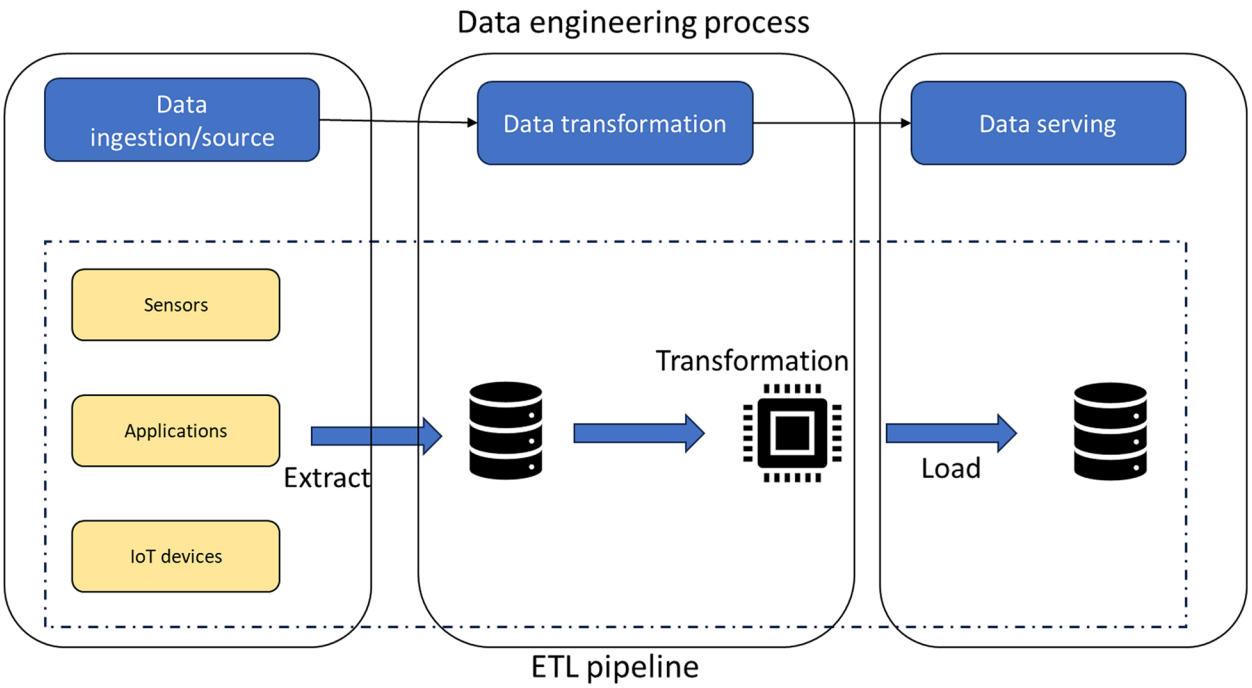

Fig. 6 Illustration of a simple data engineering process and a representation of an ETL pipeline to be deployed in one single geographic region

A data lake is a more contemporary option than a data warehouse or a conventional database. Whatever shape it takes, the storage instance acts as a storehouse for the processed data, ready for additional processing to access and examine.

A thorough representation of all the steps in an ETL pipeline, combined in one place, is shown in Fig. 6. Data collection from a wide range of different sources is the first step in the process. After that, the data is kept in a specific storage instance and may include unstructured text files or organized databases. Acting as a central repository, this instance ensures the data is easily available for the next steps. The transformation of this raw data comes next after the data collection. Data transformation is done using a "compute" step. In this stage, the data is put through several procedures to transform it into a format better suited for analysis. These processes can include normalisation, which involves adjusting data to meet a standard scale; aggregation, which involves compiling data into a summary form; and cleaning, which involves eliminating inaccurate or unnecessary data. The data is loaded into a storage instance after it has been transformed. This instance is made especially to make effective data retrieval easier, guaranteeing that the processed data is easily accessible for additional use. The transformed data is now ready for consumption, whether it is being used for reporting, machine learning techniques, or data analysis.

Figure 7 displays several ETL pipelines tailored to distinct regions that gather data from diverse sources. This data is subsequently put into a central data warehouse for processing. The hub-like representation of the data warehouse highlights its function as the primary hub for data processing and collection. The processed data is sent to several data marts from the data warehouse. These data marts provide customized information to business groups and end customers in different regions. The illustration demonstrates the effectiveness and structure of this infrastructure, demonstrating how information may be gathered from many sources, analyzed centrally, and then dispersed in a targeted way. The flow of data in a big data infrastructure from the point of origin to the end user is depicted by this visualisation. It highlights how data marts help provide customized information to various business sectors and how ETL pipelines and data warehouses process data.

Using graph nodes and edges, Fig. 8 conceptualizes the complete big data infrastructure. This conceptualisation is important to comprehend the complex relationships and connections inside the infrastructure. Several directed graphs are involved in the implementation and deployment of this big data infrastructure, requiring the identification of several paths. The data flow across the infrastructure is represented by these paths, which go from data ingestion and transformation to data warehouse storage and, ultimately, data marts for user access.

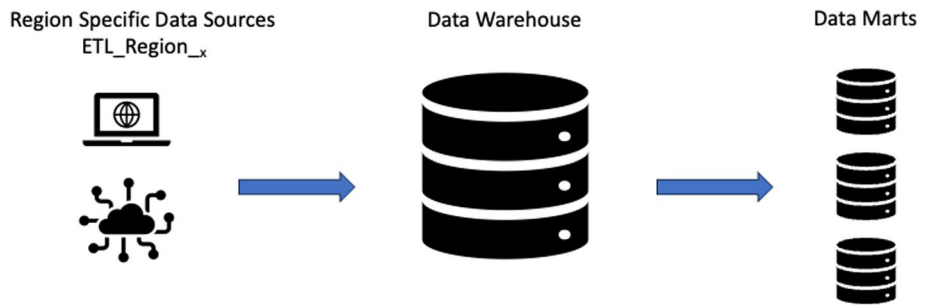

Fig. 7 An illustration of a big data infrastructure including an ETL pipeline, a data warehouse, and multiple data marts

In this context, since there are three regions, hence three starting points, the following paths for data flow are identified for each region:

For ETL_Region_a:

- Path 1: $1 \rightarrow 8$
- Path 2: $1 \rightarrow 7 \rightarrow 9$
- Path 3: $1 \rightarrow 7 \rightarrow 10$

For ETL_Region_b:

- Path 4: $15 \rightarrow 18 \rightarrow 5 \rightarrow 8$
- Path 5: $15 \rightarrow 18 \rightarrow 5 \rightarrow 7 \rightarrow 9$
- Path 6: $15 \rightarrow 18 \rightarrow 5 \rightarrow 7 \rightarrow 10$

For ETL_Region_c:

- Path 7: $11 \rightarrow 14 \rightarrow 5 \rightarrow 8$
- Path 8: $11 \rightarrow 14 \rightarrow 5 \rightarrow 7 \rightarrow 9$
- Path 9: $11 \rightarrow 14 \rightarrow 5 \rightarrow 7 \rightarrow 10$

In Fig. 8, ETL_Region_a, ETL_Region_b, and ETL_Region_c represent data pipelines designed for data ingestion and transformation. These pipelines receive raw data, process it into a format that can be used for further processing, and then forward it to the next step that

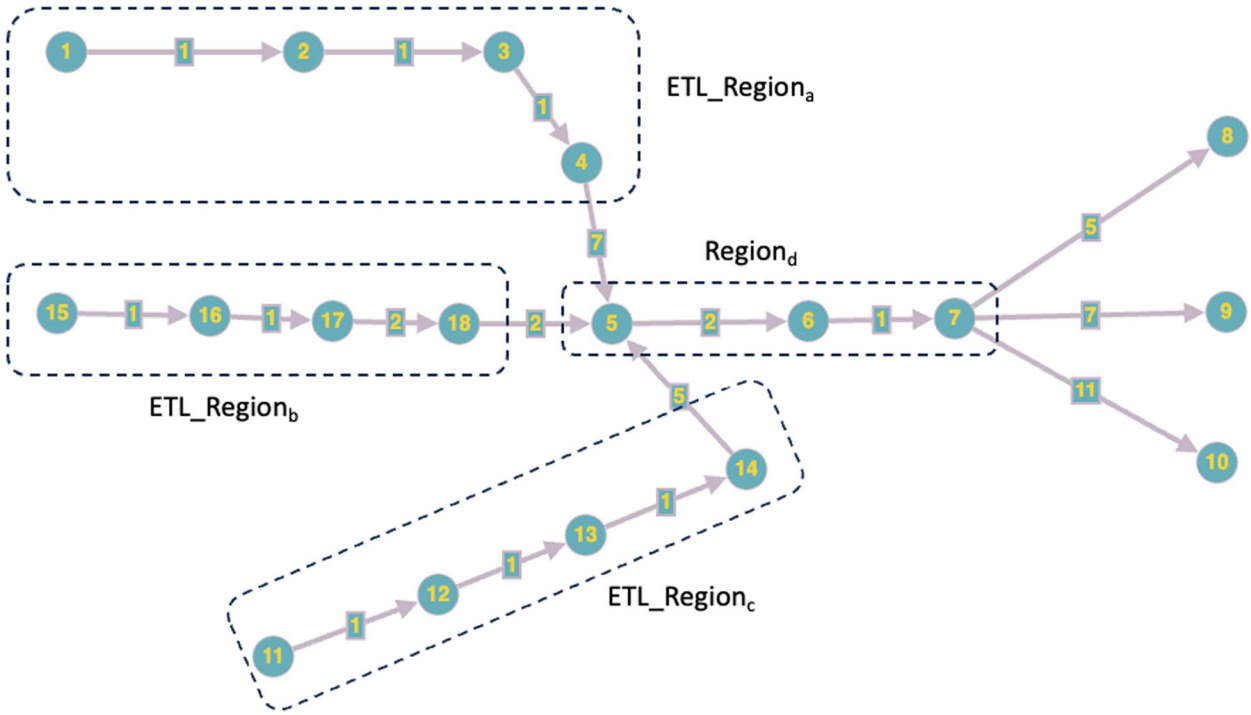

Fig. 8 Directed graph illustrating the flow and transformation of data through ETL pipelines, central data warehouse, and data marts in a big data infrastructure

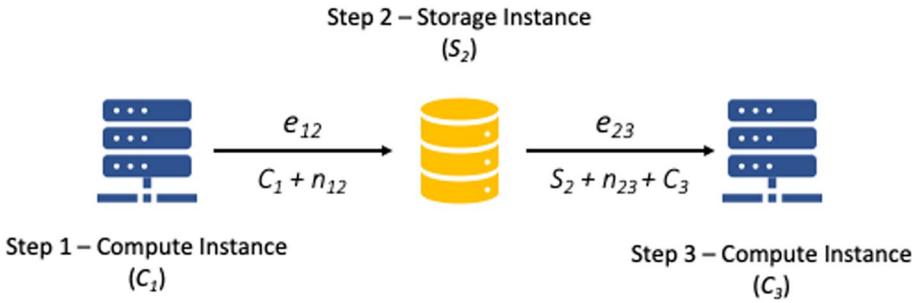

Fig. 9 A schematic representation of the three-step data pipeline, illustrating the unidirectional flow of data from computation in the US Central geographic region, through storage in the US East, to further computation in the US West geographic region

follows. $Region_d$ refers to a data warehouse, a substantial storage facility containing transformed and unprocessed data from numerous sources. It acts as a central hub for data that has been cleaned, combined, and prepared for analysis. Data marts, subsets of the data warehouse, are represented by nodes 8, 9, and 10. They are made to meet the requirements of particular teams or business divisions. Data marts let users access a portion of the data warehouse, enabling them to retrieve and examine information pertinent to their particular requirements. The visual representation in Fig. 8 makes the process and resources involved easier to understand with this graph-based representation of the big data infrastructure, which offers a clear and succinct overview of the data flow.

Evaluation

This section presents the evaluations of the proposed approach. Due to the involvement of multiple parameters and the complexity, the approach is evaluated gradually in multiple steps.

It evaluates the efficacy and performance of the suggested methodology in a range of scenarios based on the case study presented. These scenarios are designed to mimic situations expected to occur in real-world contexts. This strategy seeks to ensure that the applicability and practicality of the approach are not merely theoretical but also have practical implications. The results of this analysis will give a clear picture of the approach's strengths as well as any potential weaknesses. Note that the term "region" refers to a specific area within a cloud service, for instance, useast1 and useast2. On the other hand, "geographic region" pertains to a broader geographical area like Europe West, which encompasses further regions such as euwest1 and euwest2.

Each of the following subsections contains "Requirements" section that outlines the specifications and regions for the steps involving compute and storage instances. It provides detailed information on the number of CPUs, memory, and boot disk size[5], and other specifications for each step. This information is used as input to the approach based on the estimated prices retrieved from the pricing API. When choosing one, the total cost of a compute instance can be changed by several factors. These include the operating system, the provisioning model, the boot disk type, the inclusion of a local SSD, and the machine type. In the same way, there are other choices for storage instances as well. In addition, latency can be measured in real-time for each instance, and its impact on the cost (recall Eq. 2) can be included in the total cost.

Scenario: three-step data pipeline

In this scenario, we consider a data pipeline with only three steps: compute, storage, and compute. Data moves from step one to step three in a unidirectional manner, i.e., in a single way. The first computation step needs to be deployed in the US Central. After the data is processed in the first step, it is moved to the geographic region of US East for storage. The data is then extracted and moved to the US West area for the final computation step. Regarding QoS, the redundancy model is one of the QoS considerations. The approach ensures the integrity of the data pipeline, ensuring that the data is not lost during transfer and that it is still accurate and available. An example of this data pipeline may be seen in Fig. 9, which shows how data moves from the first computation in the US Central geographic region to storage in the US East and then to the second computation in the US West geographic region. This illustration makes it easier to understand the composition and operation of the data pipeline.

Requirements: Table 1 outlines the specifications and regions for three steps involving two compute and one

[5] A boot disk, or startup disk, is a storage device from which a computer can "boot" or "start up1". This disk contains files required by the boot sequence and the operating system, loaded at the end of the startup process.

Table 1 Requirement specifications for a three-step data pipeline, presenting a detailed overview of two compute and a storage instance

Step	Instance	Specifications	Geographical region
Step 1	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US Central
Step2	Storage	Storage: 100 TB Redundancy model: Single Network: 500 TB	US East
Step3	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US West

storage instance. A single redundancy model is selected for Step 2. A graph data structure is constructed after retrieving potential locations and their corresponding estimated prices, as shown in Fig. 10, which are determined based on the input requirements. This graph, depicted in Fig. 10, represents the interconnections between the different steps and their respective regions. The primary objective is to identify the shortest path from any given region for Step 1 to any region for Step 3, which necessarily passes through Step 2. Consequently, there are three potential starting points in Step 1 and three potential ending points in Step 3, thereby offering a variety of paths for consideration in the cost optimisation approach. This approach ensures a comprehensive analysis of all possible routes, thereby facilitating the identification of the most cost-effective path.

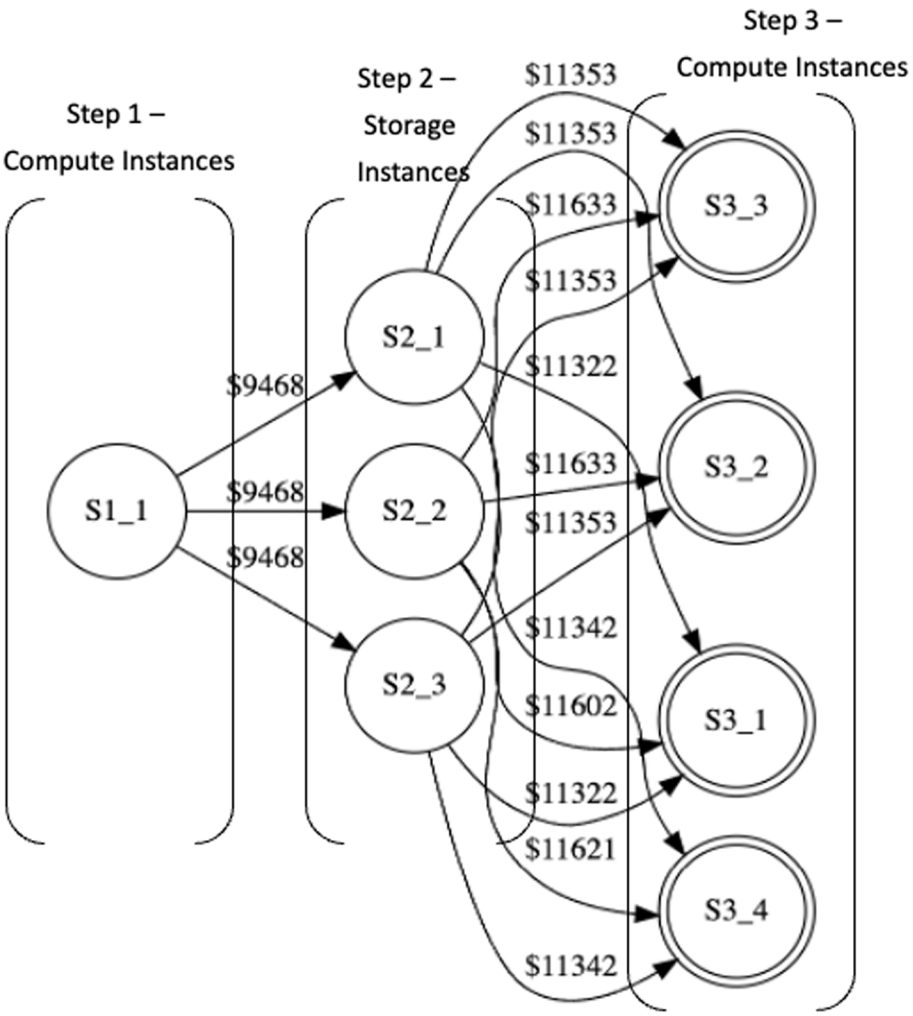

Fig. 10 A visual representation of the graph data structure for the three-step data pipeline for cost optimisation by mapping all possible paths across steps and regions. *Sx_y* represents a cloud resource and arrows show the direction of data flow. The cost on the edges is the total estimated cost of cloud resources on the left side of the edge, as well as the cost of data transfer from this resource to the next resource

Results Table 2 is created based on the information retrieved from the Google pricing API. There is one possible deployment region for Step 1, whereas there are three possibilities for Step 2 and four for Step 3. It is also interesting to note how prices vary for the same compute instance with similar specifications across different regions. Data transfer cost is estimated based on the assumption that the data pipeline is deployed using the Google Cloud Platform and that data transfer will occur in North American regions and within the Google Cloud. The graph is implemented, and the Dijkstra algorithm is used to find the cheapest deployment model. After setting the weights and running the algorithm, the following are the proposed regions for the resource deployment:

Step 1 (US Central 1) → Step 2 (US East 1) → Step 3 (US West 1).

The proposed deployment approach could lead to significant cost savings as more attributes are involved. Although, in this case, the focus is on cost optimisation, by including different QoS factors, the impact on performance can also be studied and optimised, such as latency, throughput, and the overall performance of cloud services.

Table 2 Cost estimation for three-step data pipeline for steps in different regions. All prices are in US Dollars

Steps (Instance)	Cost		Total
	Compute/Storage	Network	
Step 1 (Compute)	US Central: 148.85	9311.23	9460.08
Step 2 (Storage)	US East1: 1862.55	9311.23	11173.78
	US East4: 2142.04		11453.27
	US East5: 1862.65		11173.78
Step 3 (Compute)	US West1: 148.85		148.85
	US West2: 179.79		179.79
	US West3: 179.79		179.79
	US West4: 168.56		168.56

Scenario: multi-step data pipeline - two locations

This scenario represents a slightly extended version of the first scenario, in which an ETL pipeline is deployed in the same geographic region, but the data after transformation is stored in a warehouse deployed in a different geographic region. Hence, we are considering a data pipeline that consists of three steps: 1) compute, 2) storage, and 3) compute. Additionally, a dataware where the data would be stored. The data flows unidirectionally from the first to the third step. All steps are deployed in the same geographic region, which in this case is the US West. However, the data warehouse is deployed in the EU West. Moreover, we consider only one resource in the data warehouse. It is a slightly unlikely scenario, but it is part of the larger big data infrastructure, and it is set up like this primarily for assessment purposes. QoS considerations involve the redundancy model in this scenario. This model ensures that the data is not lost during transfer and remains accessible and accurate, thereby maintaining the integrity of the data pipeline. Figure 11 presents a visual representation of this data pipeline, illustrating the flow of data from the initial computation in the US West, through storage in the same region, and then to the subsequent computation again in the US West. From there, the data is transferred to the storage in the EU West. This visualisation aids in understanding the structure and operation of the data pipeline.

Requirements: Table 3 outlines the specifications and regions for three steps involving compute and storage instances. A graph data structure is constructed after retrieving potential locations and their corresponding estimated prices, shown in Fig. 12, which are determined based on the input requirements. This graph, depicted in Fig. 12, represents the interconnections between the

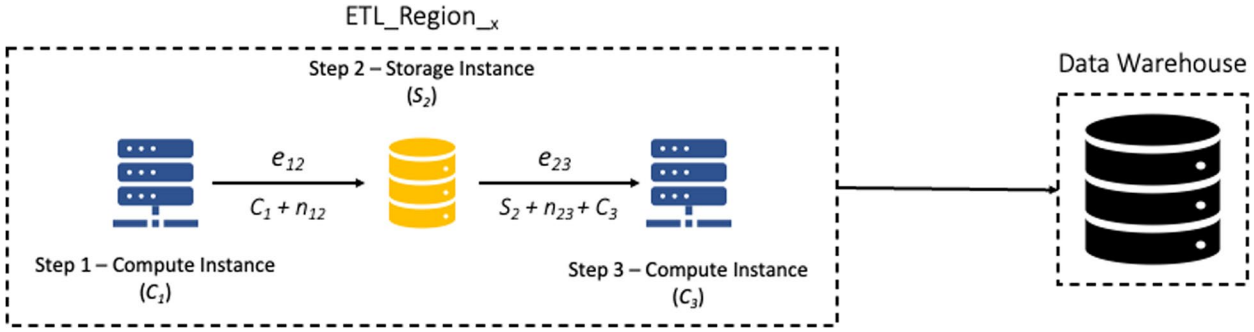

Fig. 11 A schematic representation of the ETL data pipeline, i.e., multi-step data pipeline - two locations, illustrating the unidirectional flow of data from computation in the US West into a data warehouse in the EU West

Table 3 Requirement specifications for each step in the big data pipeline, i.e., multi-step data pipeline - two locations, which is deployed in two different locations presenting a detailed overview of two compute and two storage instances

Step	Instance	Specifications	Geographic region
Step 1	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US West
Step2	Storage	Storage: 100 TB Redundancy model: Single Network: 500 TB	US West
Step3	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US West
Step4	Storage	Storage: 1000 TB Redundancy model: Single Network: 1000 TB	EU West

different steps and their respective regions. The primary objective is to identify the shortest path from any given region in Step 1 to any region in Step 4, which necessarily passes through Steps 2 and 3. Consequently, there are four potential starting points in Step 1 and eight potential ending points in Step 4, thereby offering a variety of paths for consideration in the cost optimisation approach. This approach ensures a comprehensive analysis of all possible routes, thereby facilitating the identification of the most cost-effective path.

Results Table 4 is created based on the information retrieved from the Google pricing API. There are four possible deployment regions for Steps 1, 2, and 3, whereas there are 8 possibilities for Step 4. It is also interesting to note how prices vary for the same compute instance with similar specifications across different regions. Data transfer cost is estimated based on the assumption that the data pipeline is deployed using the Google Cloud Platform and that data transfer will occur in the North American region and within the Google Cloud. For the data transfer between Steps 3 and 4, the estimated cost is retrieved for the transfer between the North American region and Europe. For 2000TB of data transfer between North America, it is approximately 37,250 US Dollars, whereas if it is between North America and Europe, it is 93,295 US Dollars. The graph is implemented, and the Dijkstra algorithm is used to find the cheapest deployment model. After setting the weights and running the algorithm, the following are the proposed regions for the resource deployment:

Step 1 (US West 1) → Step 2 (US West 1) → Step 3 (US West 1) → Step 4 (EU West 1 or EU West 4).

The deployment model suggested by the algorithm accumulates an estimate of 132,538 US Dollars compared to the most expensive, 137,536. This reduction mounts to approximately 60,000 US Dollars annually, and it occurs when the data pipeline is deployed in only two regions with a single redundancy model and does not consider different storage tiers. The larger the data pipeline with more detailed input parameters, the higher the cost reduction. Figure 12 shows how the complexity of the graph increased by adding just one more step to the data pipeline. Moreover, many more options can be included. For example, for each node of Steps 2 and 4, there could be 4 different variations based on the redundancy model selected. In addition, for Steps 1 and 3, committed use options can also be selected, such as none, 1 year, and 3 years, and then the price can be calculated. Last but not least, QoS elements such as latency and throughput can be measured and, after calculating their impact on cost, can be included in the graph.

Scenario: big data infrastructure

In this scenario, we are considering three different ETL data pipelines deployed in three different regions that consist of three steps each: 1) compute, 2) storage, and 3) compute. The data flows unidirectionally from the first to the third step. We consider a scenario in which the first ETL pipeline needs to be deployed in the US West geographic region, the second in the US East, and the third in the US South geographic region. From each ETL data pipeline, the data is transferred and stored in the data warehouse, which is deployed in the US Central geographic region. Once the data is accumulated and processed, it is transferred back to data marts deployed in the same regions as the ETL pipelines. QoS considerations involve the redundancy model in this scenario. This model ensures that the data is not lost during transfer and remains accessible and accurate, thereby maintaining the integrity of the data pipeline.

Requirements: Table 5 outlines the specifications and regions for big data infrastructure involving three ETL data pipelines, a data warehouse, and three data marts. A graph data structure is constructed after retrieving potential locations and their corresponding estimated prices, as shown in Fig. 13, which are determined based on the input requirements. This graph, depicted in Fig. 13, represents the interconnections between the different steps and their respective regions. The primary objective is to identify the shortest path from any given region in each data pipeline to any region in the respective data mart, which necessarily passes through Steps

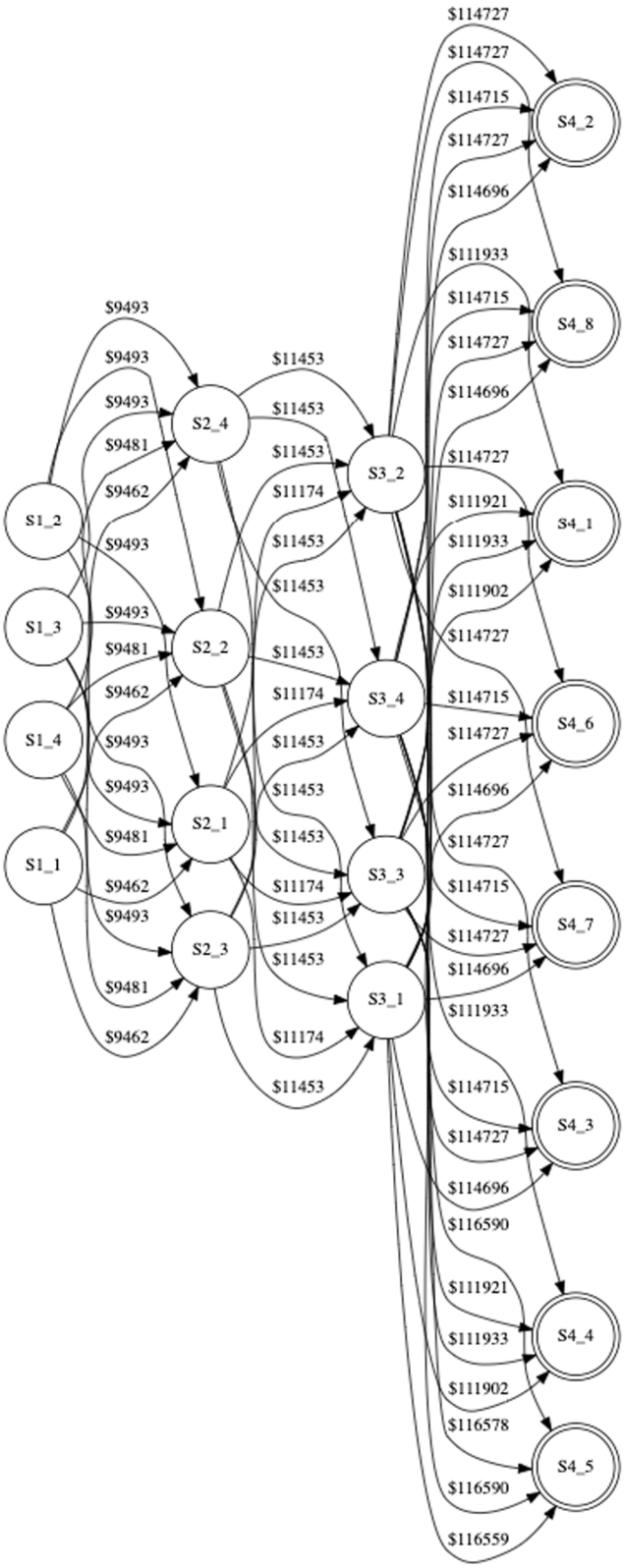

Fig. 12 A visual representation of the graph data structure for cost optimisation by mapping all possible paths across steps and regions for the multi-step data pipeline - two locations

Table 4 Cost estimation for data pipeline steps in different regions for the multi-step data pipeline - two locations

Steps (Instance)	Cost		Total
	Compute/Storage	Network	
Step 1 (Compute)	US West1: 148.85	9313.37	9462.22
	US West2: 179.79		9493.16
	US West3: 179.79		9493.16
	US West4: 168.56		9481.93
Step 2 (Storage)	US West1: 1862.55	9313.37	11173.77
	US West2: 2142.04		11453.27
	US West3: 2142.04		11453.27
	US West4: 2142.04		11453.27
Step 3 (Compute)	US West1: 148.85	93127.26	93276.11
	US West2: 179.79		93307.05
	US West3: 179.79		93307.05
	US West4: 168.56		93295.82
Step 4 (Storage)	EU West1: 18626.45		18626.45
	EU West2: 21420.42		21420.42
	EU West3: 21420.42		21420.42
	EU West4: 18626.45		18626.45
	EU West6: 23283.06		23283.06
	EU West8: 21420.42		21420.42
	EU West9: 21420.42		21420.42
	EU West12: 21420.42		21420.42

2, 3, 4, 5, and 6. Consequently, there are three potential starting points for the first data pipeline in the US East region, four for the second data pipeline in the US West region, and one for the third data pipeline in the US South region. Regarding potential ending points, the US East region has three possible ending points, the US West has four, and the US South has only one possible ending point. Thus offering a variety of paths for consideration in the cost optimisation approach. This approach ensures a comprehensive analysis of all possible routes, thereby facilitating the identification of the most cost-effective path.

Results Data is gathered based on the information retrieved from the Google Pricing API. There are three possible deployment regions for the ETL data pipeline in the US East region, four for the ETL data pipeline in the US West region, and one for the ETL data pipeline in the US South region. Moreover, there is only one possible deployment region for the data warehouse in the US central region, but two different redundancy models are considered for the storage in the data warehouse. It is also interesting to note the exponential increase in the complexity of graphs as more elements of big data infrastructure are added. More complexity means more possible deployment models, making it challenging to do a cost-benefit analysis of different deployment options without a standard framework and software tool.

Furthermore, data transfer cost is estimated based on the assumption that the data pipeline is deployed using the Google Cloud Platform and that data transfer will occur in the North American region and within the Google Cloud. The graph is implemented, and the Dijkstra algorithm is used to find the cheapest deployment model. After setting the weights and running the algorithm, the following are the proposed regions for the resource deployment:

- ETL_Region_e: Step 1 (US East 1) → Step 2 (US East 1) → Step 3 (US East 1) → Step 4 (US Central 1) → Step 5 (US East 1).
- ETL_Region_w: Step 1 (US West 1) → Step 2 (US West 1) → Step 3 (US West 1) → Step 4 (US Central 1) → Step 5 (US West 1).
- ETL_Region_s: Step 1 (US South 1) → Step 2 (US South 1) → Step 3 (US South 1) → Step 4 (US Central 1) → Step 5 (US South 1).

This shows that the proposed approach has the capability, to handle complex multi-region ETL data pipelines effectively by leveraging graph theory to map the data flow and identify cost-efficient paths for data transfer and storage. Moreover, it is scalable and flexible enough to adapt to various configurations of data pipelines and storage options. This adaptability ensures that it can be tailored to different organisational needs and infrastructure setups, further enhancing its utility in diverse real-world applications. By incorporating QoS considerations, such as redundancy models, the approach ensured data integrity and accessibility. This is critical for maintaining a reliable and accurate data pipeline.

Scenario: multi-regional big data pipeline

In this case, we present a big data pipeline that involves multiple steps, each requiring specific compute and storage resources. The deployment spans across three geographic regions: US West, EU West, and Asia East. The first step in the pipeline is data ingestion. This is handled by two compute instances located in the US West region. Each instance is equipped with 8 CPUs, 32 GB of memory, and a boot disk size of 50 GB. Following data ingestion, the data is stored in the US West region. The storage instance has a capacity of 200 TB and operates on a single-region redundancy model. The third step in the pipeline is data processing. This is carried out by four compute instances located in the EU West region. Each

Table 5 Requirement specifications for each step in the big data infrastructure presenting a detailed overview of three big data pipelines, a data warehouse, and several data marts

	Steps	Instance	Specifications	Geographic region
ETL_DataPipeline_w	Step 1	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US West
	Step 2	Storage	Storage: 100 TB Redundancy model: Single Network: 500 TB	US West
	Step 3	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US West
ETL_DataPipeline_e	Step 1	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US East
	Step 2	Storage	Storage: 100 TB Redundancy model: Single Network: 500 TB	US East
	Step 3	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US East
ETL_DataPipeline_s	Step 1	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US South
	Step 2	Storage	Storage: 100 TB Redundancy model: Single Network: 500 TB	US South
	Step 3	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US South
Data Warehouse	Step 4	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US Central
	Step 5	Storage	Storage: 100 TB Redundancy model: Single Network: 500 TB	US Central
	Step 6	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US Central
Data Marts	Step 7	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US West

Table 5 (continued)

Steps	Instance	Specifications	Geographic region
Step 8	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US East
Step 9	Compute	No. of instances: 1 No. of CPUs: 4 Memory: 16 GB Boot disk size: 20 GB	US South

of these instances has 16 CPUs, 64 GB of memory, and a boot disk size of 100 GB. Finally, the data is archived in the Asia East geographic region. The storage instance used for this purpose has a capacity of 500 TB and operates on a multi-regional redundancy model. This comprehensive and robust pipeline ensures efficient and secure handling of big data.

Requirements Table 6 presents the requirements for this multi-regional big data pipeline. A graph data structure is constructed after retrieving potential locations and their corresponding estimated costs, as shown in Fig. 14, which are determined based on the input requirements. This graph represents the interconnections between the different steps and their respective regions. The primary objective is to identify the shortest path from any given region in Step 1 to any region in Step 4, which necessarily passes through Steps 2, and 3. Consequently, there are four potential starting points for the first step in the US West geographic region, four for the second step in the data pipeline in the US West geographic region, and again four for the third step in the EU West geographic region. Regarding potential ending points, the Asia East geographic region has two possible deployment regions. Thus offering a variety of paths for consideration in the cost optimisation approach. This approach ensures a comprehensive analysis of all possible routes, thereby facilitating the identification of the most cost-effective path.

Results Table 7 is created based on the information retrieved from the Google pricing API. The "Steps" column lists the different steps involved in the data pipeline. These steps are categorized into Compute, Network, and Storage. The "Compute/Storage" column provides cost estimations for compute and storage resources used in each step of the data pipeline for different regions. The costs are based on the amount of resources (like CPU, RAM, and disk space) consumed during each step, which are specified in Table 6. This "Network" provides the cost estimations for network resources used in each step of the data pipeline for different regions. The costs are based on the amount of data transferred over the network from the given step to the next step in the data pipeline. The last column provides the total cost estimation for each step of the data pipeline for different regions. It is the sum of the "Compute/Storage" and "Network" costs. The graph is implemented, and the Dijkstra algorithm is used to find the cheapest deployment model. After setting the weights and running the algorithm, the following are the proposed regions for the resource deployment:

Step 1 (US West 1 or US West 2) → Step 2 (US West 1) → Step 3 (EU West 1) → Step 4 (Asia East Coldline).

The deployment model suggested by the algorithm accumulates an estimate of 225,052 US Dollars compared to the most expensive, 239,066. This reduction amounts to approximately 168,000 US Dollars annually, and it occurs when the data pipeline is deployed in three different geographic regions with a single redundancy model and considers only one different storage tier for the data archiving step. In usual circumstances, for data archival, Archive tier would be first choice, but based on data retrieval and storage characteristics, Coldline is suggested to be the most suitable. The larger the data pipeline with more detailed input parameters, the higher the cost reduction.

Cost comparison with baseline and robustness analysis

We define the baseline as a deployment executed without utilizing any specific cost models. Figure 15 shows the annual cost difference between graph-based and baseline deployments for the above-discussed four scenarios. This difference grows with increasing complexity and resource demands. Efficiency can be further enhanced by factoring in the impact of QoS on cost, as shown in Eq. 3. It can be seen that even seemingly a

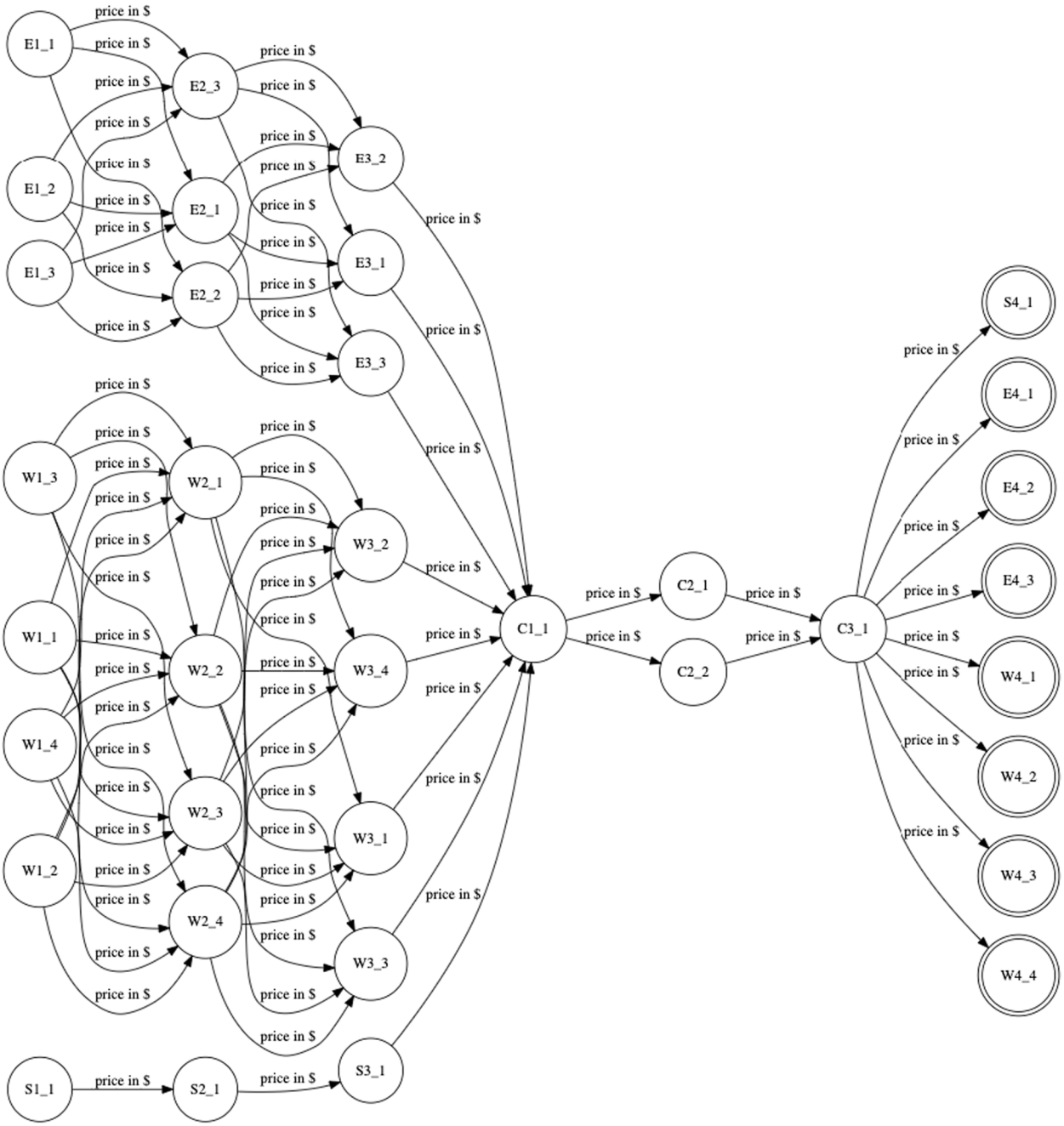

Fig. 13 A visual representation of the graph data structure for cost optimisation by mapping all possible paths across steps and regions for the big data infrastructure (actual cost estimates are not mentioned in the graph due to the limited dimensions of the figure)

small difference in total cost (~7-9%) amounts to large amounts of sum. For example, for a multi-step data pipeline, that contains a very limited number of cloud resources, the cost difference is almost 60,000 US Dollars. Similarly, this difference is around 120,000 US Dollars for a big data infrastructure scenario and 170,000 US Dollars for a multi-regional data pipeline.

We further present a sensitivity analysis to demonstrate the robustness of the proposed approach. By examining the complexities of basic cloud resource configurations, specifically a storage instance and a compute instance, and by diving into the operational specifics of these instances, we aim to highlight the strengths and potential scalability of the proposed approach.

Table 6 Requirement specifications for each step in the multi-regional big data pipeline presenting a detailed overview of two compute and two storage instances

Step	Instance	Specifications	Geographic region
Step 1: Data Ingestion	Compute	No. of instances: 2 Number of CPUs: 8 Memory: 32 GB Boot disk size: 50 GB	US West
Step 2: Data Storage	Storage	Storage: 200 TB Redundancy model: Single Network: 500 TB	US West
Step 3: Data Processing	Compute	No. of instances: 4 Number of CPUs: 16 Memory: 64 GB Boot disk size: 100 GB	EU West
Step 4: Data Archiving	Storage	Storage: 500 TB Redundancy model: Multi Tier: Archive, Coldline, Nearline Data retrieval: 300 TB Data transfer: 700 TB	Asia East

Table 8 outlines the parameters to consider when selecting a simple storage instance in Google Cloud. For instance, a storage instance with a capacity of 1000 TB should be deployed in the USWest region. Key considerations include storage tiers (Table 8 lists Standard and Nearline due to their similar characteristics), additional data retrieval costs, redundancy models, available regional options, and data transfer sources and destinations.

The impact of these parameters on the cost is illustrated in the following examples:

- Storage cost, shown in Fig. 16.
  - Size: 1000TB; tier: **standard**; redundancy model: single - USWest1 = $20,479
  - Size: 1000TB; tier: **nearline**; redundancy model: single - USWest1 = $10,240
  - Size: 1000TB: tier: **standard**; redundancy model: single - USWest2 = $23,552
  - Size: 1000TB; tier: **nearline**; redundancy model: single - USWest2 = $16,384
- Data transfer cost within Google Cloud, shown in Fig. 17.
  - Size: 1000TB; tier: standard; source: North America; destination: **North America** = $20,478
  - Size: 1000TB; tier: standard; source: North America; destination: **Europe** = $51,195
  - Size: 1000TB; tier: standard; source: North America; destination: **Asia** = $81,912
  - Size: 1000TB; tier: standard; source: North America; destination: **Indonesia** = $102,390
  - Size: 1000TB; tier: standard; source: North America; destination: **Middle East** = $112,640
  - Size: 1000TB; tier: standard; source: North America; destination: **Latin America** = $143,346

Additionally, Table 9 lists the key parameters to consider when selecting a single compute instance in Google Cloud. Some parameters are related to resource requirements, such as the number of CPUs, memory size, boot disk size, and local SSD capacity (if needed). Other parameters are qualitative, including the provisioning model, machine type, extended memory requirements, boot disk type, and committed use discount options. These parameters affect either cost, performance or both. Hence, each of the options needs to be selected carefully.

The impact of these parameters on the cost is illustrated in the following examples, also shown in Fig. 18.

- Cost of a single compute instance in us-central1 region with different provisioning models:
  - No. of vCPUs: 12; memory: 36 GiB; provisioning model: **regular**; committed use option: none = $407.60

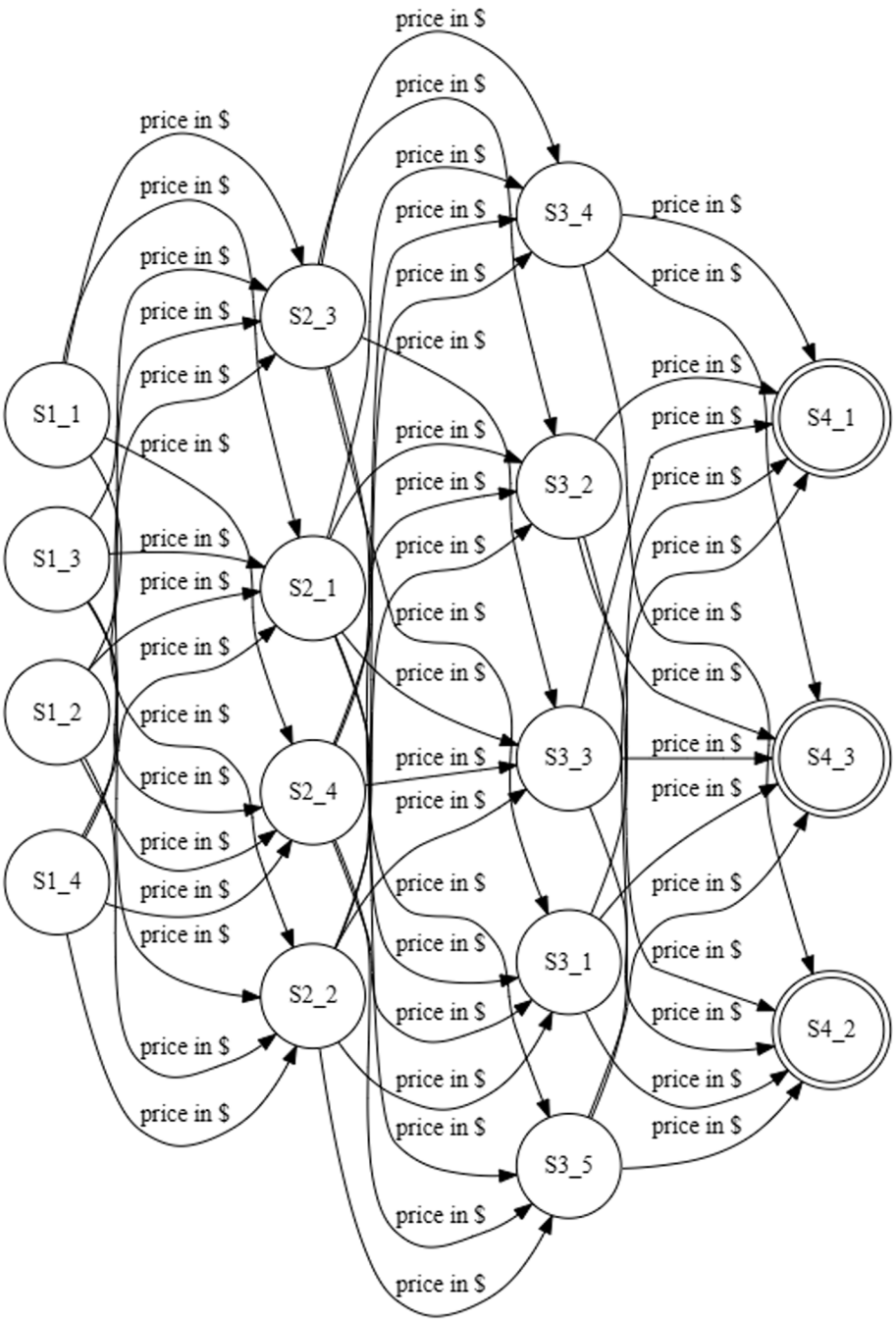

Fig. 14 A visual representation of the graph data structure for cost optimisation by mapping all possible paths across steps and regions for the multi-regional big data pipeline (actual cost estimates are not mentioned in the graph due to the limited dimensions of the figure)

  - No. of vCPUs: 12; memory: 36 GiB; provisioning model: **spot**; committed use option: none = $101.86

- Cost of a single compute instance in us-central1 region with different committed use options:
  - No. of vCPUs: 12; memory: 36 GiB; provisioning model: regular; committed use option: **none** = $407.60
  - No. of vCPUs: 12; memory: 36 GiB; provisioning model: regular; committed use option: **1 year** = $253.33

Table 7 Cost estimation for data pipeline steps in different regions for the multi-regional big data pipeline

Steps (Instance)	Cost		Total
	Compute/Storage	Network	
Step 1 (Compute)	USWest1: 598.19	10238	10836.19
	USWest2: 598.19		10836.19
	USWest3: 720.11		10958.11
	USWest4: 675.11		10913.11
Step 2 (Storage)	USWest1: 3725.19	25595	29320.19
	USWest2: 4284.08		29879.08
	USWest3: 4284.08		29879.08
	USWest4: 4284.08		29879.08
Step 3 (Compute)	EUWest1: 1317.01	81920	83237.01
	EUWest2: 1544.12		83464.12
	EUWest3: 1544.12		83464.12
	EUWest4: 1320.02		83240.02
	EUWest6: 1676.91		83596.91
Step 4 (Storage)	Archive: 41327.44	71313.84	112641.28
	Coldline: 38649.89	62931.94	101581.83
	Nearline: 44237.82	71313.84	115551.66

– No. of vCPUs: 12; memory: 36 GiB; provisioning model: regular; committed use option: **3 years** = $183.46

It can be seen that by changing the provisioning model, the cost of a compute instance can be significantly reduced. For example, switching from a regular provisioning model to a spot instance reduces the cost to one-fourth, leveraging idle capacity. Additionally, committed use options play a crucial role in cost management. Opting for a 1-year or 3-year commitment can further decrease expenses. These examples highlight the importance of careful planning and architectural design in optimizing application costs. In addition to these two parameters, there are other parameters that need to be considered before creating a cost-effective compute instance. The proposed approach has the ability to incorporate and process these additional parameters in a very minimal amount of time.

This analysis highlights the complexity of the cost structure, demonstrating that costs are influenced by a

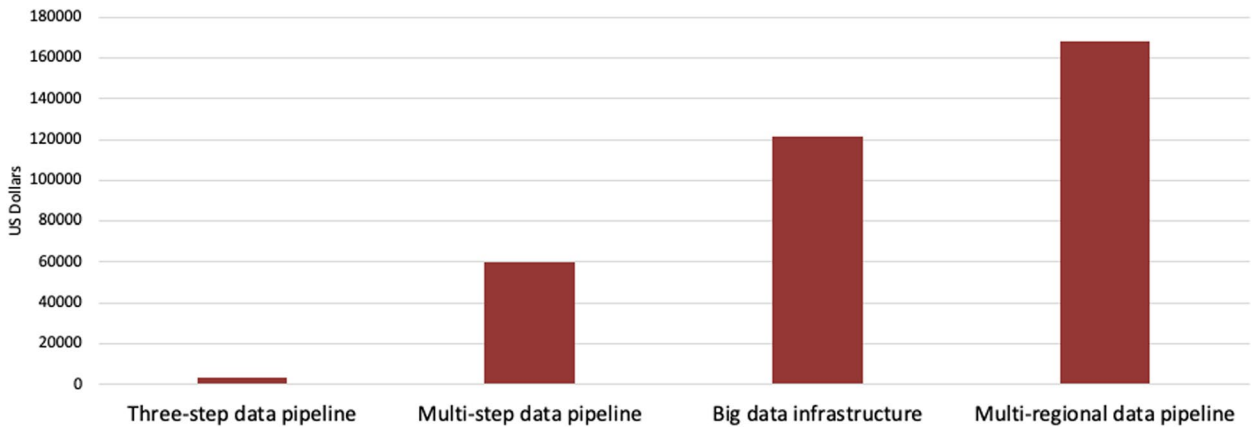

Fig. 15 Comparative analysis of graph-based approach against the baseline cost

Table 8 Range of parameters for selecting a simple storage instance in Google Cloud

Size	Storage tiers	Additional data retrieval cost	Redundancy models	Regions (USWest)	Source region	Destination region
1000 TB	Standard	No	Single	USWest1	North America	North America
	Nearline	Yes	Dual	USWest2	Europe	Europe
			Multi	USWest3	Asia	Asia
				USWest4	Indonesia	Indonesia
					Oceania	Oceania
					Middle East	Middle East
					Latin America	Latin America

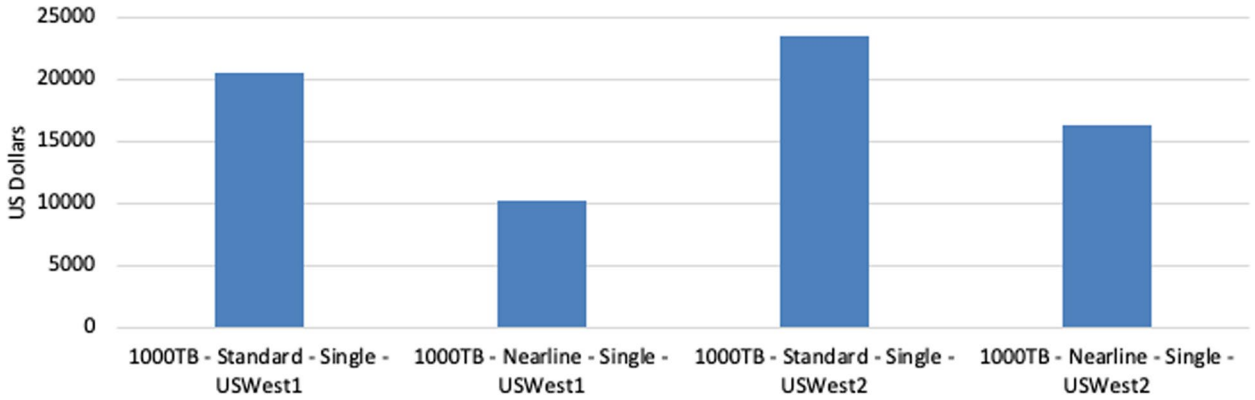

Fig. 16 Difference in storage cost for 1000TB of data based on region and storage tier

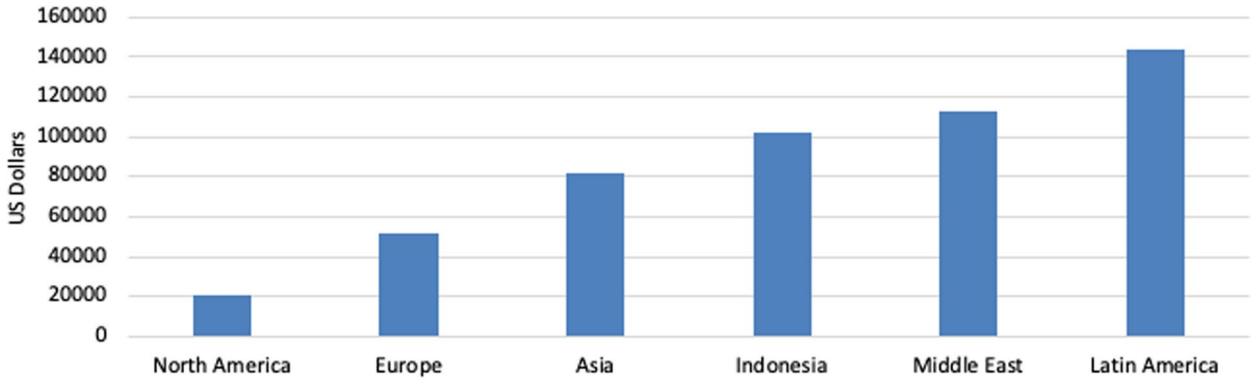

Fig. 17 Difference in data transfer cost from destination "North America" and source listed on the x-axes

Table 9 Range of parameters for selecting a simple compute instance with no. of instances set to 1 in Google Cloud

Operation system	Provisioning model	Machine type	No. of CPUs	Amount of memory	Extended memory	Boot disk type	Boot disk size (GiB)	Local SSD	Committed use discount options
Free	Regular	General purpose	Min: 1 vCPUs Max: 96 vCPUs	Min: 0.6 GiB Max: 624 GiB	Yes	Standard persistent disk	Min: 0 Max: >= 65,536	0	None
Paid	Spot (Preemptible VM)	Compute-optimized			No	Balanced persistent disk		1 x 375 GB	1 year
		Memory-optimized				SSD persistent disk		2 x 375 GB	3 years
		Accelerator-optimized						3 x 375 GB	
		Storage-optimized						4 x 375 GB	
								5 x 375 GB	
								6 x 375 GB	
								7 x 375 GB	
								8 x 375 GB	
								16 x 375 GB	
								24 x 375 GB	

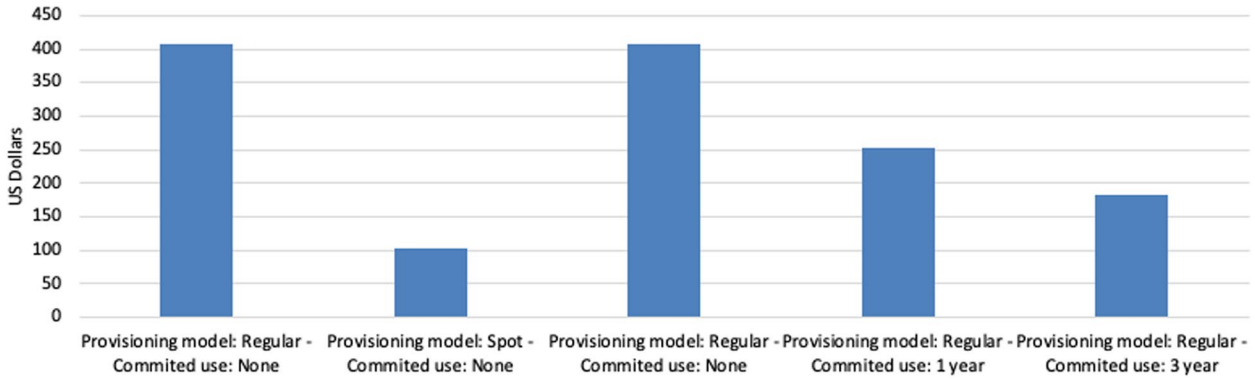

Fig. 18 Difference in cost of a single compute instance with 12 vCPUs, 36 GiB memory based on provisioning model and committed use discount options

Table 10 A comparison of graph-based approach with existing literature in terms of strategy, cost elements, scalability potential, technology focus and innovation

	Strategy	Cost optimisation	Scalability	Technology focus	Innovation
Grahp-based approach	Data placement using graph-theory	Advanced cost optimisation techniques	Enhanced scalability for diverse applications	Platform- & industry independent	Use of graph-theory to incorporate multiple parameters and address existing gaps
[26, 28]	Smart data placement using storage-as-a-service model	Not explicitly focused	High scalability for big data pipelines	Storage-as-a-service	MCDA-based storage selection model
[34]	Dynamic replication and migration	For replication and migration	Scalable replication and migration	Cloud data centers	Advanced cost optimisation
[57]	Edge server placement	Cost-effective placement	Scalable edge server placement	Wireless metropolitan area networks	Novel edge server placement
[51]	Edge server placement for IoT services	Cost-aware placement optimisation	Scalable IoT services	Wireless metropolitan area networks	Novel IoT service placement
[18]	Caching-as-a-Service	Cost-driven caching	Scalable caching solutions	Cloud-based 5G mobile networks	Innovative caching approach
[31]	Aggregation networks for streaming analytics	Cost-aware aggregation	Scalable aggregation networks	Geo-distributed streaming analytics	Advanced aggregation techniques
[12]	Transmission scheme for inter-datacenter networks	Cost-efficient transmission	Scalable transmission schemes	Inter-datacenter networks	Novel transmission scheme
[20]	Cloud service selection	Not explicitly focused	Scalable cloud service selection	Cloud services	Unique fuzzy multi-criteria approach
[40]	Cloud provider selection	Availability requirements	Scalable cloud provider selection	Cloud storage services	Unique provider selection model
[38]	Intelligent service selection for IoT	Cost-effective service selection	Scalable IoT service selection	Cloud providers for IoT	Intelligent multi-dimensional selection

combination of multiple factors rather than a single element. It also showcases the model's robustness and its ability to integrate these factors and generate a model almost instantaneously (see the next subsection). Additionally, performance metrics can be incorporated. For instance, latency is crucial for the application's performance. Real-time latency can be monitored using services like AWS Latency Monitoring[6]. The cost-effectiveness ratio can be calculated (refer to Eq. 3), ensuring efficient deployment.

Qualitative comparison with the state of the art

In this section, a qualitative comparison with the state-of-the-art is presented. Table 10 presents a comparative analysis of a graph-based approach against existing literature in terms of strategy, cost optimization, scalability potential, technology focus, and innovation. The

[6] https://www.cloudping.co/grid

graph-based approach leverages graph-theory for data placement, offering advanced cost optimisation techniques and enhanced scalability across diverse applications. It is platform and industry-independent, making it applicable to a wide range of scenarios and applications. It also incorporates multiple parameters, as compared to approaches focused solely on data placement, edge server placement, and caching techniques.

Computational cost

In this section, we present the computational cost of the graph-based approach in terms of processing time. Dijkstra's algorithm has many variants, but in this case, we have used the one to find the shortest paths from the source node to all other nodes in the graph. The time complexity of Dijkstra's algorithm is $O(V^2)$ where V is the number of nodes or vertices. However, since we have multiple possible starting nodes, the computational time will be higher based on the number of possible starting nodes. The proposed approach is implemented using the Java programming language and executed on a machine running MacOS, 32GB of RAM, and an Intel Core i7 processor. Figure 19 shows the number of nodes on the x-axis, computational time in milliseconds, and number of starting nodes on the y-axis. The time taken to retrieve data from the pricing API is not included, as that can vary based on the network speed; however, the time for creating and loading graph data structures in memory is included along with the traversal time. The computational cost of the graph-based approach using Dijkstra's algorithm is very minimal and it increases as the number of nodes and starting nodes increase. Moreover, the graph in Fig. 19 clearly illustrates this trend, showing a steeper increase in computational time as the number of nodes increases. While Dijkstra's algorithm is efficient for smaller graphs or when the number of nodes is limited, its computational cost can become significant for larger graphs or when there are large numbers of nodes with many potential starting nodes. For such scenarios, alternative approaches may be necessary to ensure efficient computation within reasonable time frames.

Discussion & conclusions

The graph-based approach proposed in this article presents a novel and efficient approach to modelling cloud cost elements. Its platform independence is a significant advantage, allowing for its application across various environments and scenarios. This versatility is further demonstrated through the evaluations performed on generic scenarios. The approach's applicability is not confined to any industry, making it a universally adaptable solution for various user scenarios. During the evaluations, the approach effectively depicted the

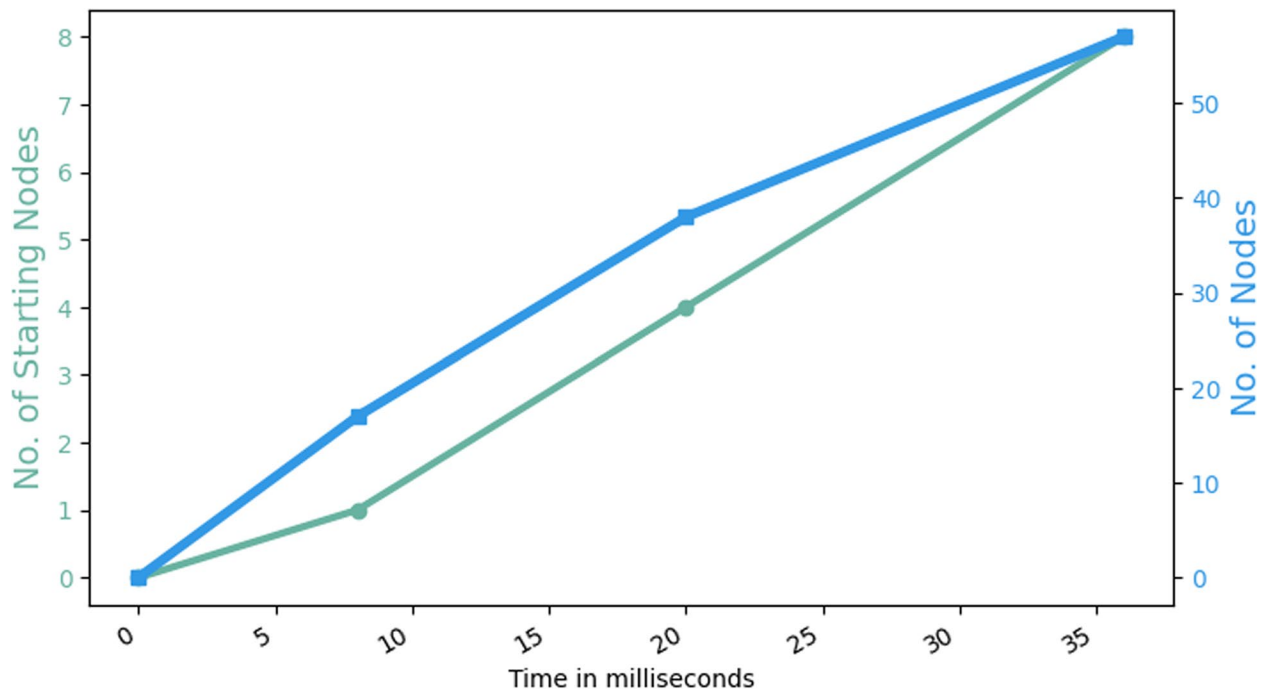

Fig. 19 Comparison of starting nodes and computation time vs. total number of nodes. This graph illustrates the relationship between the number of nodes in a network, the number of starting nodes, and the computation time required in milliseconds

complexity and variety of cost offerings from a major CSP, i.e., Google Cloud. This demonstration highlights the approach's capability to handle diverse cost structures and indicates that it can be used with other CSPs offering similar patterns in their cost structures. This observation suggests the proposed approach would be equally effective when applied to other CSPs.

The effectiveness of the proposed approach is further evidenced by the different paths of graph traversal. These paths validate the approach's effectiveness and open up many opportunities for its extension. The approach's adaptability and extensibility make it a promising tool for future research and practical applications in cloud cost modelling. Moreover, scalability is a crucial aspect of any cloud-related solution, and it is relevant in the context of the proposed graph-based cloud cost modelling and optimisation approach. Computational cost and sensitivity analysis is performed to depict the computational viability and the robustness and scalability of the proposed approach. It allows for dynamic resource allocation. As resource demand changes, it can adapt by reallocating resources across different regions and instances. This flexibility can help maintain optimal performance and cost-efficiency. Moreover, cloud service costs can vary over time due to changes in pricing models or fluctuations in demand. The proposed approach can adapt to these changes by recalculating the optimal deployment model, ensuring that resource allocation remains cost-effective.

Regarding future work, there is a limitation in the current approach's applicability, which might be addressed by including a wider range of resource aspects for additional testing. Furthermore, the evaluation of the approach relies on a robust big data infrastructure, which is a relatively straightforward situation. The practical usefulness of this technology may not be fully presented due to the lack of testing in more sophisticated, real-world user settings. Another constraint exists in the integration of QoS elements. Currently, the approach incorporates a pre-selected redundancy model for QoS. Expanding the number of QoS factors and considering their potential fluctuations might improve the approach's resilience. In order to enhance the usability, improvements and studies on user interface could be considered, while extending its functionality. This would improve the approach's usability and applicability, increasing its accessibility to a broader spectrum of users and scenarios. These constraints offer further investigation and innovation opportunities to improve the approach's efficiency and relevance.

The proposed approach can be integrated with the current cloud management systems, enabling the efficient and scalable administration of resources across various platforms and services. The graph-based approach possesses the fundamental characteristic of being extendable, which implies that it may be effortlessly enlarged to incorporate additional categories of resources or cost-related aspects. This feature allows for easy expansion and adaptation to future developments and modifications in the cloud services industry, making it a flexible and adaptable solution. Additionally, the proposed approach can potentially can be integrated with container orchestration systems like Kubernetes. For example, in terms of resource allocation and optimisation, node selection can be done for containers as per the output of the proposed approach to minimise resource usage costs like CPU and memory. In terms of network cost optimisation, Kubernetes clusters often span multiple nodes, regions, or even cloud providers. The graph-based approach could be used to model the network topology and data flows within a Kubernetes cluster, thereby optimising data transfer paths.

Acknowledgements
We thank Nikolay Nikolov for the initial discussions around the topic of cloud storage cost that helped to shape the direction of the work in this article.

Authors' contributions
A. Q. K. is a PhD student and undertaken conceptualisation, methodology, experiements, analysis, and writing & review. M. M., R. P., C. B., D. R., and A. S. provided supervision and contributed to conceptualisation and writing & review.

Funding
Open access funding provided by NTNU Norwegian University of Science and Technology (incl St. Olavs Hospital - Trondheim University Hospital) This work was partly funded through the EC-funded projects DataCloud (H2020 101016835), enRichMyData (HE 101070284), Graph-Massivizer (HE 101093202), UPCAST (HE 101093216), and INTEND (HE 101135576).

Availability of data and materials
No research data outside the submitted manuscript file.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 24 June 2024 Accepted: 18 September 2024
Published online: 30 September 2024

References

1. (2021) What Is a Data Lake? Pros and Cons of Data Lakes. https://www.masterclass.com/articles/what-is-a-data-lake. Accessed 9 May 2024
2. Al-Hakim L, Kusiak A, Mathew J (2000) A graph-theoretic approach to conceptual design with functional perspectives. Comput Aided Des 32(14):867–875. https://doi.org/10.1016/S0010-4485(00)00075-0

3. Ashabi A, Sahibuddin SB, Haghighi MS (2020) Big data: current challenges and future scope. In: Proceedings of 10th Symposium on Computer Applications Industrial Electronics (ISCAIE 2020), IEEE, pp 131–134. https://doi.org/10.1109/ISCAIE47305.2020.9108826
4. Bang-Jensen J, Gutin GZ (2008) Digraphs: theory, algorithms and applications. Springer Science & Business Media
5. Belov V, Kosenkov AN, Nikulchev E (2021) Experimental Characteristics Study of Data Storage Formats for Data Marts Development within Data Lakes. Appl Sci 11(18):8651. https://doi.org/10.3390/app11188651
6. Chawla H, Khattar P (2020) Data Ingestion, Apress, Berkeley, pp 43–85. https://doi.org/10.1007/978-1-4842-6252-8_4
7. Cormen T, Leiserson C, Rivest R, Stein C (2009) Introduction to algorithms, 3rd ed. MIT Press, p 693
8. Corodescu AA, Nikolov N, Khan AQ, Soylu A, Matskin M, Payberah AH, Roman D (2021a) Locality-aware workflow orchestration for big data. In: Proceedings of the 13th International Conference on Management of Digital EcoSystems (MEDES 2021), Springer, pp 62–70. https://doi.org/10.1145/3444757.348510
9. Corodescu AA, Nikolov N, Khan AQ, Soylu A, Matskin M, Payberah AH, Roman D (2021) Big data workflows: Locality-aware orchestration using software containers. Sensors 21(24):8212. https://doi.org/10.3390/s21248212
10. Dauphiné A (2017) 8 - Models of Basic Structures: Networks. In: Dauphiné A (ed) Geographical Models with Mathematica, Elsevier, pp 199–224. https://doi.org/10.1016/B978-1-78548-225-0.50011-7
11. Dedić N, Stanier C (2016) An Evaluation of the Challenges of Multilingualism in Data Warehouse Development. In: Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016), SciTePress, pp 196–206. https://doi.org/10.5220/0005858401960206
12. Dong X, Zhao L, Zhou X, Li K, Guo D, Qiu T (2019) An Online Cost-Efficient Transmission Scheme for Information-Agnostic Traffic in Inter-Datacenter Networks. IEEE Trans Cloud Comput 10(1):202–215. https://doi.org/10.1109/TCC.2019.2941688
13. Donida Labati R, Genovese A, Piuri V, Scotti F, Vishwakarma S (2020) Computational Intelligence in Cloud Computing, Springer, pp 111–127. https://doi.org/10.1007/978-3-030-14350-3_6
14. Dowsett C (2023) What Is a Data Lake? https://builtin.com/data-science/data-lake. Accessed 9 May 2024
15. Feijen W, Schäfer G (2021) Dijkstras algorithm with predictions to solve the single-source many-targets shortest-path problem. CoRR 1–28. https://doi.org/10.48550/arXiv.2112.11927
16. Gandhi O, Agrawal V (1992) FMEA–A diagraph and matrix approach. Reliab Eng Syst Saf 35(2):147–158. https://doi.org/10.1016/0951-8320(92)90034-I
17. Gass SI, Fu MC (eds) (2013) Dijkstra's Algorithm, Springer US, Boston, p 428. https://doi.org/10.1007/978-1-4419-1153-7_200148
18. Ghoreishi SE, Karamshuk D, Friderikos V, Sastry N, Dohler M, Aghvami AH (2019) A Cost-Driven Approach to Caching-as-a-Service in Cloud-Based 5G Mobile Networks. IEEE Trans Mob Comput 19(5):997–1009. https://doi.org/10.1109/TMC.2019.2904061
19. Goldberg A, Radzik T (1993) A heuristic improvement of the bellman-ford algorithm. Stanford University - Computer Science Department, Tech. rep
20. Ilieva G, Yankova T, Hadjieva V, et al (2020) Cloud Service Selection as a Fuzzy Multi-criteria Problem. TEM J 9(2):484. https://doi.org/10.18421/TEM92-09
21. Irfan M, George JP (2022) A Systematic Review of Challenges, Tools, and Myths of Big Data Ingestion. In: Proceedings of the International Conference on Data Science for Computational Security (IDSCS 2022), Springer, LNNS, vol 462, pp 481–494. https://doi.org/10.1007/978-981-19-2211-4_43
22. Johnson DB (1977) Efficient Algorithms for Shortest Paths in Sparse Networks. J ACM 24(1):1–13. https://doi.org/10.1145/321992.321993
23. Karatas G (2024) Data Lake: What It Is, Benefits & Challenges in 2024. https://research.aimultiple.com/data-lake/. Accessed 9 May2024
24. Khan AQ, Nikolov N, Matskin M, Prodan R, Bussler C, Roman D, Soylu A (2023) Towards Cloud Storage Tier Optimization with Rule-Based Classification. In: Proceedings of the 10th IFIP WG 6.12 European Conference on Service-Oriented and Cloud Computing (ESOCC 2023), Springer, LNCS, vol 14183, pp 205–216. https://doi.org/10.1007/978-3-031-46235-1_13
25. Khan AQ, Nikolov N, Matskin M, Prodan R, Bussler C, Roman D, Soylu A (2023) Towards Graph-based Cloud Cost Modelling and Optimisation. In: Proceedings of 47th Annual Computers, Software, and Applications Conference (COMPSAC 2023), IEEE, pp 1337–1342. https://doi.org/10.1109/COMPSAC57700.2023.00203
26. Khan AQ, Nikolov N, Matskin M, Prodan R, Song H, Roman D, Soylu A (2022) Smart Data Placement for Big Data Pipelines: An Approach based on the Storage-as-a-Service Model. In: Proceedings of 15th International Conference on Utility and Cloud Computing (UCC 2022), IEEE, pp 317–320. https://doi.org/10.1109/UCC56403.2022.00056
27. Khan AQ, Nikolov N, Matskin M, Prodan R, Song H, Roman D, Soylu A (2023) A Taxonomy for Cloud Storage Cost. In: Proceedings of 15th International Conference on Management of Digital Ecosystems (MEDES 2023), Springer, CCIS, vol 2022, pp 317–330. https://doi.org/10.1007/978-3-031-51643-6_23
28. Khan AQ, Nikolov N, Matskin M et al (2023) Smart Data Placement Using Storage-as-a-Service Model for Big Data Pipelines. Sensors 23(2):564. https://doi.org/10.3390/s23020564
29. Khan AQ, Matskin M, Prodan R, Bussler C, Roman D, Soylu A (2024) Cloud storage tier optimization through storage object classification. Computing 1–30. https://doi.org/10.1007/s00607-024-01281-2
30. Khan AQ, Matskin M, Prodan R, Bussler C, Roman D, Soylu A (2024) Cloud storage cost: a taxonomy and survey. World Wide Web 27(4):36
31. Kumar D, Ahmad S, Chandra A, Sitaraman RK (2021) AggNet: Cost-Aware Aggregation Networks for Geo-distributed Streaming Analytics. In: Proceedings of the IEEE/ACM Symposium on Edge Computing (SEC 2021), IEEE, pp 297–311. https://doi.org/10.1145/3453142.3491276
32. Liu G, Shen H (2017) Minimum-cost cloud storage service across multiple cloud providers. IEEE/ACM Trans Networking 25(4):2498–2513. https://doi.org/10.1109/TNET.2017.2693222
33. Liu J, Shen H, Chi H et al (2020) A Low-Cost Multi-Failure Resilient Replication Scheme for High-Data Availability in Cloud Storage. IEEE/ACM Trans Networking 29(4):1436–1451. https://doi.org/10.1109/TNET.2020.3027814
34. Mansouri Y, Toosi AN, Buyya R (2017) Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centers. IEEE Trans Cloud Comput 7(3):705–718. https://doi.org/10.1109/TCC.2017.2659728
35. Mansouri Y, Toosi AN, Buyya R (2017) Data Storage Management in Cloud Environments: Taxonomy, Survey, and Future Directions. ACM Comput Surv 50(6):1–51. https://doi.org/10.1145/3136623
36. Martens B, Walterbusch M, Teuteberg F (2012) Costing of Cloud Computing Services: A Total Cost of Ownership Approach. In: Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS 2012), IEEE, pp 1563–1572. https://doi.org/10.1109/HICSS.2012.186
37. Micheli A (2009) Neural Network for Graphs: A Contextual Constructive Approach. IEEE Trans Neural Netw 20(3):498–511. https://doi.org/10.1109/TNN.2008.2010350
38. Milani OH, Motamedi SA, Sharifian S et al (2021) Intelligent Service Selection in a Multi-Dimensional Environment of Cloud Providers for Internet of Things Stream Data through Cloudlets. Energies 14(24):8601. https://doi.org/10.3390/en14248601
39. Nikolov N, Dessalk YD, Khan AQ et al (2021) Conceptualization and scalable execution of big data workflows using domain-specific languages and software containers. Internet Things 16:100440. https://doi.org/10.1016/j.iot.2021.100440
40. Oki E, Kaneko R, Kitsuwan N, et al (2017) Cloud provider selection models for cloud storage services to satisfy availability requirements. IEICE Trans Commun E100-B(8):1406–1418. https://doi.org/10.1587/transcom.2016EBP3403
41. Pishvaee MS, Rabbani M (2011) A graph theoretic-based heuristic algorithm for responsive supply chain network design with direct and indirect shipment. Adv Eng Softw 42(3):57–63. https://doi.org/10.1016/j.advengsoft.2010.11.001
42. Raj T, Shankar R, Suhaib M, Khan R (2010) A graph-theoretic approach to evaluate the intensity of barriers in the implementation of fmss. Int J Serv Oper Manag 7(1):24–52. https://doi.org/10.1504/IJSOM.2010.033142
43. Rajeshwari BS, Dakshayini M, Guruprasad HS (2022) Workload balancing in a multi-cloud environment: challenges and research directions, Springer, pp 129–144. https://doi.org/10.1007/978-3-030-74402-1_7
44. Ramachandran GS, Radhakrishnan R, Krishnamachari B (2018) Towards a Decentralized Data Marketplace for Smart Cities. In: Proceedings of International Smart Cities Conference (ISC 2 2018), IEEE, pp 1–8. https://doi.org/10.1109/ISC2.2018.8656952
45. Ravat F, Zhao Y (2019) Data Lakes: Trends and Perspectives. In: Hartmann S, Küng J, Chakravarthy S, Anderst-Kotsis G, Tjoa AM, Khalil I (eds)

Database and Expert Systems Applications, Springer, pp 304–313. https://doi.org/10.1007/978-3-030-27615-7_23
46. Robinson K (2021) Why companies are flocking to the cloud more than ever. https://www.businessinsider.com/cloud-technology-trend-software-enterprise-2021-2. Accessed 20 Feb 2023
47. Roman D, Prodan R, Nikolov N et al (2022) Big Data Pipelines on the Computing Continuum: Tapping the Dark Data. Computer 55(11):74–84. https://doi.org/10.1109/MC.2022.3154148
48. Russell SJ (2010) Artificial intelligence a modern approach. Pearson Education, Inc
49. Sabharwal S, Garg S (2013) Determining cost effectiveness index of remanufacturing: A graph theoretic approach. Int J Prod Econ 144(2):521–532. https://doi.org/10.1016/j.ijpe.2013.04.003
50. Scarselli F, Gori M, Tsoi AC et al (2008) The Graph Neural Network Model. IEEE Trans Neural Netw 20(1):61–80. https://doi.org/10.1109/TNN.2008.2005605
51. Shao Y, Shen Z, Gong S et al (2022) Cost-Aware Placement Optimization of Edge Servers for IoT Services in Wireless Metropolitan Area Networks. Wirel Commun Mob Comput 2022. https://doi.org/10.1155/2022/8936576
52. Song IY (2009) Data Mart, Springer US, Boston, p 594. https://doi.org/10.1007/978-0-387-39940-9_883
53. Upadhyay N, Agarwal VP (2007) Structural Identification and Comparison of Intelligent Mobile Learning Environment. J Appl Quant Methods 2(4):363–374
54. Vargas-Solar G, Darmont J, Adorjan A, Espinosa-Oviedo JA, Hara C, Loudcher S, Motz R, Musicante M, Zechinelli-Martini JL (2024) Dataversifying Natural Sciences: Pioneering a Data Lake Architecture for Curated Data-Centric Experiments in Life & Earth Sciences. arXiv:2403.20063
55. West DB, et al (2001) Introduction to graph theory, vol 2. Prentice Hall Upper Saddle River
56. Xia X, Chen F, He Q et al (2020) Graph-based data caching optimization for edge computing. Futur Gener Comput Syst 113:228–239. https://doi.org/10.1016/j.future.2020.07.016
57. Zeng F, Ren Y, Deng X et al (2018) Cost-effective edge server placement in wireless metropolitan area networks. Sensors 19(1):32. https://doi.org/10.3390/s19010032

Publisher's Note